Centro de Ciências Exatas, Ambientais e de Tecnologias

Faculdade de Engenharia de Computação

Paradigmas de Linguagens de Programação II

2º Semestre de 2003
(Volume 1 de 2)

Prof. André Luís dos R.G. de Carvalho

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS
INSTITUTO DE INFORMÁTICA
PLANO DE ENSINO

Faculdade:	Disciplina:	Docente:	C.h. Semanal	Período Letivo:
Engenharia de Computação	Paradigmas de Linguagens de Programação	André Luís dos Reis Gomes de Carvalho	06 (seis horas aula)	2º Semestre de 2003

Objetivo Geral da Disciplina:

- Solidificar e aprofundar os conhecimentos de programação orientada a objetos.

Avaliação:

- 70% da nota será dada pela média aritmética de duas provas aplicadas pelo professor da parte teórca da discipilna.
- 30% da nota será dada por uma série, eventualmente unitária, de projetos a serem desenvolvidos por solicitação do professor da parte prática desta disciplina.

Freqüência:

- Nenhum abono de falta será concedido pelos professores. Qualquer problema neste sentido deverá ser encaminhada ao PA que tomará todas as providências no sentido de encaminhar a solução do mesmo.
- Para fins de controle de freqüência, não será permitido que alunos desta turma assistam aulas em outras turma, e nem o contrário.

Bibliografia Utilizada :

- Concepts of Programming Languages
  Sebesta, R.W.
- Conceitos de Linguagens de Programação
  Ghesi, C.; e Jazayeri, M.
- Programming Languages: Design and Implementation
  Pratt, T.W.
- Object Orientede System Analysis: Modelling the World in Data
  Shlaer, S.; e Mellor, S.J.
- Object Lifecycles: Modelling the World in States
  Shlaer, S.; e Mellor, S.J.
- Introdução à Programação Orientada a Objetos
  Takahashi, T.
- Programação Orientada a Objetos
  Takahashi T.; e Liesemberg, H.K.
- Programação Orientada para Objeto
  Cox, B.J.
- The TAO of Objects: A Beginner's Guide to Object Oriented Programming
  Entsminger, G.
- A Complete Object Oriented Design Example
  Richardson, J.E.; Schulz, R.C.; e Berard, E.V.
- Java Unleashed
  Morrison, M.; December, J.; et alii
- Dominando o Java
  Naughton, P.
- C Completo e Total
  Schildt, H.
- The C Programming Language
  Kernighan, B.W; e Ritchie, D.M.
- C++: Manual de Referência Comentado (Documento Oficial do ANSI)
  Ellis, M.A., Stroustrup, B.
- Programação Orientada para Objeto e C++
  Wiener, R.S.; e Pinson, L.J.
- Object-Oriented Programming using C++
  Pohl, I.
- Programação Orientada para Objeto com Turbo C++
  Perry, G.
- Programação Orientada para Objeto com Turbo C++
  Weiskamp, K.; Heiny, S.; e Flaming, B.
- Programação Orientada para Objeto com Turbo C++
  Perry, G.
- Visual C++ - Guia de Desenvolvimento Avançado
  Barbakati, N.

Nº aulas	Conteúdos selecionados

22	ASPECTOS AVANÇADOS DE PROGRAMAÇÃO ORIENTADA A OBJETOS EM LINGUAGEM JAVA:
	
	- RESISTÊNCIA A FALHAS
	  - Tratamento de Excessões → (1) programação em alto nível de abstração; (2) programação em baixo nível de abstração; (3) limitações do programador; (4) a cláusula finally.
	
	- AS BIBLIOTECAS DE CLASSES DA LINGUAGEM JAVA
	  - Visão Geral das Bibliotecas de Classes → (1) o pacote Language; (2) o pacote Utilities; (3) o pacote I/O; (4) classes relacionadas com threads; (10) classes relacionadas com tratamento de erros; (5) classes relacionadas com processos.
	  - O Pacote Language → (1) a classe Object; (2) classes relacionadas com os tipos de dados: a classe Boolean / a classe character / classes relacionadas com inteiros / classes relacionadas com reais; (3) a classe Math; (4) classes relacionadas com Strings: a classe String / a classe StringBuffer; (5) a classe System; (6) a classe Runtime; (7) a classe Class; (8) a classe ClassLoader.
	  - O Pacote Utilities → (1) interfaces: enumeration / observer; (2) classes: BitSet / Date / Random / StringTokenizer / Vector / Stack / Dictionary / Hashtable / Properties / Observable.
	  - O Pacote I/O → (1) classes de entrada: a classe InputSream / o objeto System.in / a classe BufferedInputStream / a classe DataInputStream / a classe FileInputStream / a classe StringBufferInputStream; (2) classes de saída: a classe OutputStream / a classe PrintSream / o objeto System.out / a classe BufferedOutputStream / a classe DataOutputSream / a classe FileOutputStream; (3) classes relacionadas com arquivos: a classe File / a classe RandomAccessFile.
	
	- PROGRAMAÇÃO DE APPLETS
	  - Visão Geral de Programação de Applets → (1) o que é uma applet: applets e a WWW / diferença entre applets e aplicações; (2) os limites das applets: limites funcionais / limites impostos pelo navegador; (3) noções básicas sobre applets: herança da classe Applet / HTML; (4) exemplos básicos de applets.
	  - O Pacote AWT (abstract window toolkit) → (1) uma applet AWT simples; (2) tratamento de eventos: tratamento de eventos em detalhes / handleEvent () ou action () / geração de eventos; (3) componentes: componentes de interface / containers / métodos comuns a todos os componentes; (4) como projetar uma interface de usuário.
	  - O Pacote *Applets e Gráficos* → (1) características das applets; o ciclo de vida de uma applet: init () / start () / stop () / destroy (); (2) como explorar o navegador: como localizar arquivos / imagens / como usar o MediaTracker / audio; (3) contextos de uma applet: showDocument () /

showStatus () / como obter parâmetros; (4) gráficos; (5) uma applet simples.
- Como programar Applets → (1) projeto básico de applets: interface do usuário / projeto das classes; (2) applets no munto real: applets devem ser pequenas / como responder ao usuário.

- MULTIPROGRAMAÇÃO
  - Threads e Multithreading → (1) o que são threads e para que elas servem; (2) como escrever applets com threads; (3) o problema do conceito de paralelismo; (4) como pensar em termos de multithreads; (5) como criar e usar threads; (6) como saber que uma thread parou; (7) thread scheduling: preemptivo X não preemptivo; como testar.

- ACESSO A BASES DE DADOS
  - O Pacote SQL → (1) drivers; (2) classes: Connection, Statement, ResultSet.

- PROGRAMAÇÃO DISTRIBUÍDA
  - ServerSockets e Sockets.

PROGRAMAÇÃO ORIENTADA A OBJETOS EM LINGUAGEM C++:

- ASPECTOS BÁSICOS
  - Classes, membros, e tipos de membros
  - Funções sobrecarregadas
  - Construtores e destrutores
  - Classes derivadas
  - Classes abstratas

- AMIZADE
  - Funções amigas
  - Classes amigas

- OPERADORES
  - Operadores como funções
  - Sobrecarga de operadores

- HERANÇA MÚLTIPLA
  - Derivação múltipla
  - Classes base virtuais
- STREAMS
  - E/S estreams
  - Formatação
  - Dispositivos padrão de E/S e streams
  - Arquivos e streams
  - Strings e streams
  - E/S em streams de tipos do usuário
- TEMPLATES
  - Templates de função
  - Templates de classe
- EXCEÇÕES
  - Tratamento de exceções
  - Discriminação e nomeação de exceções
  - Exceções que não são erros
  - Especificação de interface
  - Exceções não tratadas
  - Alternativas para tratamento de erros

.

Índice

Exceções

O autor de uma biblioteca pode detectar erros com os quais não consegue lidar. O usuário de uma biblioteca pode querer tratar adequadamente erros que não é capaz de detectar. O conceito de exceção vem para prover meios para resolver este tipo de situação.
A idéia fundamental é que uma função que detecta um problema que não é capaz de tratar lança uma exceção que pode ser pega por uma função que quer tratar o problema mas que não é capaz de detectar.

Uma exceção é um objeto que é uma instância da classe Throwable (ou de uma de suas subclasses).

Métodos que podem lançar exceções devem deixar este fato claro através do acréscimo ao final de seu cabeçalho da palavra chave throws seguida pelos nomes de todas as exceções possíveis de serem lançadas pelo método separadas por virgulas (,).

Quando um método chama outro que, por sua vez pode lançar uma exceção, o primeiro deve:

1. Tratar a referida exceção; ou
2. Avisar através da palavra chave throws o possível lançamento da referida exceção.

Exceções da classe Error ou RunTimeException (e derivadas) não precisam ser mencionados em um throws porque podem ocorrer em qualquer momento, em qualquer ponto do programa, por qualquer razão.

Sendo Exceção o identificador de uma exceção, veja abaixo a forma geral do comando que lança uma exceção.

```
throw new Excecao ();
```

A chamada de um métodos que lançam exceções deve ocorrer da seguinte maneira: primeiro deve vir a palavra chave try; em seguida deve vir, entre chaves ({}) as chamadas dos métodos que podem causar o lançamento de exceções.

O processo de lançar e pegar exceções envolve uma busca retrógrada por um tratador na cadeia de ativações a partir do ponto onde a exceção foi lançada.

A partir do momento que um tratador pega uma exceção já considera-se a exceção tratada, e qualquer outro tratador que possa ser posteriormente encontrado se torna neutralizado.

Exceções são pegas pelo comando catch. Um try pode ser sucedido por um número arbitrariamente grande de catches, cada qual com o objetivo de tratar uma das várias exceções que podem ser lançadas dentro do try.

O comando catch tem a seguinte forma geral: primeiro vem a palavra chave catch e, em seguida, entre parênteses (()), o nome da exceção que deve ser tratada. Logo após deve vir, entre chaves ({}), o código que trata a referida exceção.

Após todos os catches pode vir um finally. O comando finally tem a seguinte forma geral: primeiro vem a palavra chave finally e, em seguida, vem, entre chaves, o código a ser executado, em qualquer caso, no final do tratamento das exceções (independentemente do fato de em algum tratamento que o antecedesse executar comandos que quebram o fluxo de controle, e.g., return, break ou continue).

Uma classe pode lançar várias exceções, e as funções usuárias podem ou não pegá-las a todas. Exceções não pegas em um certo nível podem ser pegas em um nível mais alto.

Tratadores de exceção podem pegar uma exceção mas não conseguir tratá-la completamente.

Neste caso pode ser necessário um novo lançamento de exceção para provocar um tratamento complementar em um nível mais alto.

Tratadores mais sofisticados podem precisar receber dados. Como exceções nada mais são objetos de um certo tipo, pode-se perfeitamente implementar exceções que carreguem dados consigo.

Sendo Exceção o identificador de uma exceção e dd_i os dados que devem acompanhar a excessão, veja abaixo a forma geral do comando que lança uma exceção com dados:

$$\text{throw new Excecao } (dd_1, dd_2, ..., dd_n);$$

Podem existir conjuntos de exceções cuja natureza e tratamento sejam semelhantes. Podemos implementar exceções deste tipo como uma hierarquia de exceções .

Eventualmente pode-se desejar construir um tratador que pegue e trate um grupo (ou ate mesmo qualquer) de exceções lançado pelo bloco try que o precede.

Para tanto basta escrever um catch que trata uma exceção de alto nível na hierarquia de exceções (talvez a raiz). Tais tratadores podem funcionar como uma espécie de "caso contrário" em um seqüência de tratadores.

Posto que construtores são chamados automaticamente, temos que construtores não retornam resultado. Exceções podem ser uma forma interessante e única de retornar a informação de uma falha no processo de construção.

[C:\ExplsJava\Expl_01\BibAgenda\ExcecaoDeAgenda.java]

```
package BibAgenda;

public class ExcecaoDeAgenda extends Exception
{
    public static final int CAPACIDADE_INVALIDA   = 0,
                            AGENDA_CHEIA          = 1,
                            AGENDA_VAZIA          = 2,
                            NOME_INEXISTENTE      = 3,
                            NOME_JA_REGISTRADO    = 4,
                            FONE_FORA_DOS_LIMITES = 5,
                            NOME_FORA_DOS_LIMITES = 6,
                            EXCECAO_INVALIDA      = 7;

    private static final int MENOR_ID = CAPACIDADE_INVALIDA,
                             MAIOR_ID = NOME_FORA_DOS_LIMITES;

    public static final String Mensagem [] =
    {
         "Capacidade deve ser um numero natural positivo",
         "Esgotou-se a capacidade da agenda",
         "Nao ha contatos registrados na agenda",
         "O nome deste contato nao consta na agenda",
         "O nome deste contato ja consta na agenda",
         "So solicite telefones entre zero e a quantidade existente",
         "So solicite nomes entre zero e a quantidade existente",
         "Indicacao de excecao invalida"
    };


    private int Tipo;

    private ExcecaoDeAgenda (int T)
    {
        super (ExcecaoDeAgenda.Mensagem [T]);
        this.Tipo = T;
    }

    public static void indique (int T) throws ExcecaoDeAgenda
    {
        if (T<ExcecaoDeAgenda.MENOR_ID || T>ExcecaoDeAgenda.MAIOR_ID)
```

```
            throw new ExcecaoDeAgenda (ExcecaoDeAgenda.EXCECAO_INVALIDA);

        throw new ExcecaoDeAgenda (T);

    }

    public int seuTipo ()
    {
        return this.Tipo;
    }
}
```

[C:\ExplsJava\Expl_01\BibAgenda\Agenda.java]

```
package BibAgenda;

public class Agenda
{
    private int QtosContatos = 0;

    private String  Nome    [],
                    Telefone [];

    private int ondeEsta (String N)
    {
        int Inicio = 0, Final = this.QtosContatos - 1, Onde;

        while (Inicio <= Final)
        {
            Onde = (Inicio + Final) / 2;

            int Comparacao = N.compareTo (this.Nome [Onde]);

            if (Comparacao == 0)
                return Onde;
            else
                if (Comparacao < 0)
                    Final = Onde - 1;
                else
                    Inicio = Onde + 1;
        }

        return -1;
    }

    private int ondeGuardar (String N)
    {
        int Inicio     = 0,
            Final      = this.QtosContatos - 1,
            Onde       = 0,
            Comparacao = -1;

        while (Inicio <= Final)
        {
            Onde       = (Inicio + Final) / 2;
            Comparacao = N.compareTo (this.Nome [Onde]);

            if (Comparacao == 0)
                return -1;
            else
```

```
                if (Comparacao < 0)
                    Final = Onde - 1;
                else
                    Inicio = Onde + 1;
            }

            return Comparacao < 0? Onde: Onde + 1;
        }

        public Agenda (int C) throws ExcecaoDeAgenda
        {
            if (C <= 0)
                ExcecaoDeAgenda.indique (ExcecaoDeAgenda.CAPACIDADE_INVALIDA);

            this.Nome     = new String [C];
            this.Telefone = new String [C];
        }

        public int capacidade ()
        {
            return this.Nome.length;
        }

        public int quantosContatos ()
        {
            return this.QtosContatos;
        }

        public boolean haRegistroDoContato (String N)
        {
            return this.ondeEsta (N) != -1;
        }

        public void registreOContato (String N, String T) throws ExcecaoDeAgenda
        {
            if (this.QtosContatos == this.Nome.length)
                ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_CHEIA);

            int Posicao = this.ondeGuardar (N);

            if (Posicao == -1)
                ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_JA_REGISTRADO);

            for (int I = this.QtosContatos - 1; I >= Posicao; I--)
            {
                this.Nome     [I+1] = this.Nome     [I];
                this.Telefone [I+1] = this.Telefone [I];
            }

            this.Nome     [Posicao] = N;
            this.Telefone [Posicao] = T;

            this.QtosContatos++;
        }

        public String digaMeONome (int P) throws ExcecaoDeAgenda

        {
            if (this.QtosContatos == 0)
                ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_VAZIA);

            if (P < 0 || P >= this.QtosContatos)
                ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_FORA_DOS_LIMITES);
```

```
        return this.Nome [P];
    }

    public String digaMeOTelefone (int P) throws ExcecaoDeAgenda
    {
        if (this.QtosContatos == 0)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_VAZIA);

        if (P < 0 || P >= this.QtosContatos)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.FONE_FORA_DOS_LIMITES);

        return this.Telefone [P];
    }

    public String digaMeOTelefoneDe (String N) throws ExcecaoDeAgenda
    {
        if (this.QtosContatos == 0)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_VAZIA);

        int Posicao = this.ondeEsta (N);

        if (Posicao == -1)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_INEXISTENTE);

        return this.Telefone [Posicao];
    }

    public void descarteContato (String N) throws ExcecaoDeAgenda
    {
        if (this.QtosContatos == 0)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_VAZIA);

        int Posicao = this.ondeEsta (N);

        if (Posicao == -1)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_INEXISTENTE);

        int I;

        for (I = Posicao; I < this.QtosContatos - 1; I++)
        {
            this.Nome    [I] = this.Nome     [I+1];
            this.Telefone [I] = this.Telefone [I+1];
        }

        this.Nome     [I] = null;
        this.Telefone [I] = null;

        this.QtosContatos--;
    }
}
```

[C:\ExplsJava\Expl_01\TesteDeAgenda.java]

```
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import BibAgenda.*;
```

```
class TesteDeAgenda
{
    public static void main (String Args [])
    {
        BufferedReader Entrada = new BufferedReader
                                 (new InputStreamReader
                                 (System.in));

        Agenda agenda = null;

        for (;;)
        {
            System.out.println ();

            System.out.print ("Capacidade desejada para a Agenda: ");
            try
            {
                int C  = Integer.parseInt (Entrada.readLine ());
                agenda = new Agenda (C);
            }
            catch (IOException E)
            {}
            catch (NumberFormatException E)
            {
                System.err.println ("Nao foi digitado um numero inteiro");
                System.err.println ();
                continue;
            }
            catch (ExcecaoDeAgenda E)
            {
                System.err.println (E.getMessage());
                System.err.println ();
                continue;
            }

            break;
        }

        System.out.println ();

        String Nome = null, Telefone = null;

        char Opcao = ' ';

        do
        {
            System.out.println ();

            System.out.print ("Digite sua Opcao (" +
                              "I=Incluir/" +
                              "C=Consultar/" +
                              "L=Listar/" +
                              "E=Excluir/" +
                              "S=Sair)" +
                              ": ");

            try
            {
                Opcao = (Entrada.readLine ()).charAt (0);
            }
            catch (IOException E)
            {}
```

```
    Opcoes: switch (Opcao)
            {
                case 'i':
                case 'I':
                    if (agenda.quantosContatos () == agenda.capacidade ())
                    {
                        System.err.println

(ExcecaoDeAgenda.Mensagem[ExcecaoDeAgenda.AGENDA_CHEIA]);
                        System.err.println ();
                        break;
                    }

                    try
                    {
                        System.out.print ("Nome....: ");
                        Nome = Entrada.readLine ();
                    }
                    catch (IOException E)
                    {}

                    if (agenda.haRegistroDoContato (Nome))
                    {
                        System.err.println

(ExcecaoDeAgenda.Mensagem[ExcecaoDeAgenda.NOME_JA_REGISTRADO]);
                        System.err.println ();
                        break;
                    }

                    try
                    {
                        System.out.print ("Telefone: ");
                        Telefone = Entrada.readLine ();
                    }
                    catch (IOException E)
                    {}

                    try
                    {
                        agenda.registreOContato (Nome, Telefone);
                    }
                    catch (ExcecaoDeAgenda E)
                    {}

                    System.out.println ();
                    break;


                case 'c':
                case 'C':
                    if (agenda.quantosContatos () == 0)
                    {
                        System.err.println

(ExcecaoDeAgenda.Mensagem[ExcecaoDeAgenda.AGENDA_VAZIA]);
                        System.err.println ();
                        break;
                    }

                    try
                    {
                        System.out.print ("Nome....: ");
```

```
                        Nome = Entrada.readLine ();
                    }
                    catch (IOException E)
                    {}

                    try
                    {
                        Telefone = agenda.digaMeOTelefoneDe (Nome);
                    }
                    catch (ExcecaoDeAgenda E)
                    {
                        System.err.println

(ExcecaoDeAgenda.Mensagem[ExcecaoDeAgenda.NOME_INEXISTENTE]);
                        System.err.println ();
                        break;
                    }

                    System.out.println ("Telefone: " + Telefone);
                    System.out.println ();
                    break;


                case 'l':
                case 'L':
                    String Tecla;

        Exibicao: for (int I = 0;; I++)
                    {
                        try
                        {
                            System.out.println ("Nome: " +
                                                agenda.digaMeONome (I) +
                                                " - Telefone: " +
                                                agenda.digaMeOTelefone (I));
                        }
                        catch (ExcecaoDeAgenda E)
                        {
                            switch (E.seuTipo ())
                            {
                                case ExcecaoDeAgenda.AGENDA_VAZIA:
                                    System.err.println (E.getMessage ());
                                    System.err.println ();
                                    break Opcoes;

                                case ExcecaoDeAgenda.NOME_FORA_DOS_LIMITES:
                                    break Exibicao;
                            }
                        }

                        if (I%23 == 22 && I < agenda.quantosContatos()-1)
                        {
                            System.out.println ();
                            System.out.print ("Tecle ENTER... ");

                            try
                            {
                                Tecla = Entrada.readLine ();
                            }
                            catch (IOException E)
                            {}
```

```
                            System.out.println ();
                        }
                    }

                    System.out.println ();
                    break;


                case 'e':
                case 'E':
                    if (agenda.quantosContatos () == 0)
                        System.err.println

(ExcecaoDeAgenda.Mensagem[ExcecaoDeAgenda.AGENDA_VAZIA]);

                    else
                    {
                          System.out.print ("Nome....: ");
                          try
                          {
                              Nome = Entrada.readLine ();
                          }
                          catch (IOException E)
                          {}

                          try
                          {
                              agenda.descarteContato (Nome);
                          }
                          catch (ExcecaoDeAgenda E)
                          {
                              // certamente sera
                              // ExcecaoDeAgenda.NOME_INEXISTENTE

                              System.err.println (E.getMessage ());
                          }
                    }

                    System.out.println ();
                    break;


                case 's':
                case 'S':
                    break;


                default :
                    System.err.println ("Opcao invalida");
                    System.err.println ();
            }
        }
        while ((Opcao != 's') && (Opcao != 'S'));
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_01> javac -classpath . TesteDeAgenda.java
C:\ExplsJava\Expl_01> java -classpath . TesteDeAgenda
```

Isto poderá produzir no console a seguinte saída:

```
Capacidade desejada para a Agenda: 100

Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Jose
Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Jose
Esse nome ja consta na agenda!


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): e
Nome....: Joao
Nao consta na agenda!


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): c
Nome....: Jose
Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Joao
Telefone: 202.9955


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): l
Nome: Joao - Telefone: 202.9955
Nome: Jose - Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): s
```

E/S em Dispositivos não Padrão (O Pacote java.io)

Entende-se por dispositivo padrão de E/S estão normalmente associados ao teclado (dispositivo padrão de entrada) e à tela (dispositivo padrão de saída). Entende-se por dispositivo não padrão qualquer outro dispositivo que não esses dois, e.g., arquivos, strings, etc.

A Classe java.io.File

A classe java.io.File implementa métodos que possibilitam a manutenção de arquivos e diretórios em um sistema de arquivos. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**
  - **static String pathSeparator:**
    Representa o caractere de separação na variável de ambiente PATH (como String);
  - **static char pathSeparatorChar:**
    Representa o caractere de separação na variável de ambiente PATH (como char);
  - **static String separator:**
    Representa o caractere de separação no nome completo de arquivos (como String);
  - **static char separatorChar:**
    Representa o caractere de separação no nome completo de arquivos (como String);
- **Construtores:**
  - **File (String NC):**
    Constroi uma instância da classe File; NC especifica o nome completo do arquivo (incluindo path);
  - **File (String P, String N):**
    Constroi uma instância da classe File; P especifica o path do arquivo, ao passo que N especifica o nome do arquivo;
  - **File (File D, String N):**
    Constroi uma instância da classe File; funciona de forma semelhante ao construtor anterior, exceto pelo fato de que temos outro objeto da classe File (D) especificando o diretório onde se encontra o arquivo de nome N;
- **Métodos:**
  - **boolean canRead ():**
    Verifica se este File pode ser lido;

- **boolean canWrite ():**
  Verifica se este File pode ser escrito;

- **int compareTo (File F):**
  Compara lexicograficamente o nome deste File com o do File F; retorna um valor negativo, no caso do primeiro ser menor que o primeiro; retorna 0, no caso deles serem iguais; retorna um número positivo no caso do primeiro ser maior do que o segundo;

- **boolean createNewFile ():**
  No caso do arquivo representado por este File não existir, cria um novo arquivo vazio;

- **static File createTempFile (String P, String S):**
  Cria no diretório temporário padrão um arquivo temporário com prefixo P e sufixo S, retornando uma instância da classe File que o representa;

- **static File createTempFile (String P, String S, File D):**
  Cria no diretório representado pelo File D um arquivo temporário com prefixo P e sufixo S, retornando uma instância da classe File que o representa;

- **boolean delete ():**
  Remove o arquivo ou diretório representado por este File, retornando true, em caso de sucesso**,** e false, em caso de insucesso;

- **void deleteOnExit ():**
  Solicita a remoção do arquivo ou diretório representado por este File por ocasião da parada da Java Virtual Machine;

- **boolean exists ():**
  Verifica se o arquivo representado por este File existe;

- **File getAbsoluteFile () ou File getCanonicalFile ():**
  Retorna uma instância da classe File que representa o nome absoluto do arquivo ou diretório representado por este File;

- **String getAbsolutePath () ou String getCanonicalPath ():**
  Retorna uma instância da classe String que representa o nome absoluto do arquivo ou diretório representado por este File;

- **String getName ():**

  Retorna uma instância da classe String que representa o nome do arquivo ou diretório representado por este File;

- **String getParent ():**

  Retorna o nome do diretório pai deste File; retorna null se nenhum diretorio pai for encontrado;

- **File getParentFile ():**

  Retorna uma instância da classe File que representa o nome do diretório pai deste File; retorna null se nenhum diretorio pai for encontrado;

- **String getPath ():**

  Retorna uma instância da classe String que representa o path do diretório representado por este File;

- **boolean isAbsolute ():**

  Verifica se o nome do arquivo ou diretório que este File representa foi especificado de forma absoluta;

- **boolean isDirectory ():**

  Verifica se este File representa um diretório;

- **boolean isFile ():**

  Verifica se este File representa um arquivo;

- **boolean isHidden ():**

  Verifica se este File representa um arquivo ou diretório oculto;

- **long lastModified ():**

  Retorna o valor UTC do instante quando, pela última vez, este File foi atualizado;

- **long length ():**

  Retorna o tamanho em bytes deste File;

- **String [] list ():**

  Este método é aplicável somente se este File representar um diretório; retorna um

vetor de instâncias da classe String contendo os nomes dos arquivos e diretórios que existem no diretório representado por este File;

- **File [] listFiles ():**
  Este método é aplicável somente se este File representar um diretório; retorna um vetor de instâncias da classe File que representam os arquivos e diretórios que existem no diretório representado por este File;

- **static File [] listRoots ():**
  Retorna um vetor de instâncias da classe File que representam as raízes dos sistemas de arquivo existentes;

- **boolean mkdir ():**
  Cria o diretório especificado por este File (os diretórios dentro dos quais ele se encontrará devem ter sido previamente criados); retorna true, em caso de sucesso, e false, em caso de insucesso;

- **boolean mkdirs ():**
  Cria o diretório especificado pelo File (os diretórios dentro dos quais ele se encontrará podem não existir e serão também criados); retorna true, em caso de sucesso, e false, em caso de insucesso;

- **boolean renameTo (File F):**
  Troca o nome do arquivo ou diretório representado por este File para o nome especificado pelo File F; retorna true, em caso de sucesso, e false, em caso de insucesso;

- **boolean setLastModified (long T):**
  Ajusta a data e o horário da última modificação do arquivo ou diretório representado por este File para o instante de tempo representado pela coordenada UTC T; retorna true, em caso de sucesso, e false, em caso de insucesso;

- **boolean setReadOnly ():**
  Torna o arquivo ou diretório representado por este File um arquivo para leitura apenas; retorna true, em caso de sucesso, e false, em caso de insucesso;

- **URL toURL ():**
  Retorna uma URL que equivale ao nome do arquivo ou diretório representado por este File.

E/S de Texto

A Classe java.io.InputStreamReader

Derivada da classe **Reader**, esta classe funciona como uma ponte entre streams de bytes e streams de caracteres. Ela lê bytes e os traduz para caracteres. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **InputStreamReader (InputStream S):**
    Constroi uma instância da classe InputStreamReader; S, em geral é System.in.

A Classe java.io.FileReader

Derivada da classe **Reader**, esta classe se trata de uma classe de conveniência para a leitura de arquivos texto. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **FileReader (String N):**
    Constroi uma instância da classe FileReader associada ao arquivo de nome N;
  - **FileReader (File F):**
    Constroi uma instância da classe FileReader associada ao File F.

A Classe java.io.StringReader

Derivada da classe **Reader**, esta classe se trata de uma classe de conveniência para a leitura de texto a partir de String's. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

- **StringReader (String S):**

  Constroi uma instância da classe StringReader associada ao String S.

A Classe java.io.BufferedReader

Esta classe lê caracteres de um stream de entrada de caracteres. Para tanto, faz uso de um buffer a fim de prover um acesso eficiente ao dispositivo de entrada. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

  - **BufferedReader (Reader R):**

    Constroi uma instância da classe BufferedReader que usa um buffer de entrada com tamanho padrão.

  - **BufferedReader (Reader R, int T):**

    Constroi uma instância da classe BufferedReader que usa um buffer de entrada com o tamanho T dado.

- **Métodos:**

  - **void close ():**

    Fecha o BuffererReader;

  - **void mark (int Q):**

    Marca a posição atual deste BufferedReader (esta marca permanecerá válida enquanto o limite de Q caracteres lidos após o emprego deste método não tiver sido atingido);

  - **boolean markSupported ():**

    Verifica se este BufferedReader suporta void mark (int Q);

  - **int read ():**

    Lê um caractere deste BufferedReader;

  - **int read (char V []):**

    Lê caracteres deste BufferedReader e os posiciona em V; este método bloqueia o fluxo de execução até a disponibilidade de dados a serem lidos, a ocorrência de um erro de

I/O, ou a detecção de EOF; retorna a quantidade de caracteres lidos (ou –1, no caso de EOF);

- **int read (char V [], int P, int Q):**
  Lê no máximo Q caracteres deste BufferedReader e os posiciona em V a partir da posição P; este método bloqueia o fluxo de execução até a disponibilidade de dados a serem lidos, a ocorrência de um erro de I/O, ou a detecção de EOF; retorna a quantidade de caracteres lidos (ou –1, no caso de EOF);

- **String readLine ():**
  Retorna uma instância da classe String contendo uma linha lida deste BufferedReader;

- **boolean ready ():**
  Verifica se este BufferedReader está pronto para uma leitura;

- **void reset ():**
  Reposiciona este BufferedReader na posição marcada pelo último uso do método void mark (int Q);

- **long skip (long Q):**
  Despreza no máximo Q caracteres deste BufferedReader; retorna a quantidade de caracteres efetivamente desprezada.

A Classe java.io.OutputStream

A classe OutputStream é útil por permitir operações simples de saída de dados. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **OutputStream ():**
    Constroi uma instância da classe OutputStream para promover saída de dados.

- **Métodos:**
  - **void close ():**
    Fecha este OutputStream;

- **void flush ():**
  Força o esvaziamento do buffer de saída deste OutpuStream, fazendo-o efetivar toda e qualquer saída pendente;
- **void write (byte V []):**
  Escreve múltiplos bytes; espera dados em formato binário, não em formato ASCII;
- **void write (byte V [], int P, int Q):**
  Escreve no máximo Q bytes do vetor V a partir da posição P; escreve dados em formato binário, não em formato ASCII;
- **void write (int B):**
  Escreve o byte menos significativos de B; escreve dados em formato binário, não em formato ASCII.

A Classe java.io.FileOutputStream

Derivada de **OutputStream**, a classe FileOutputStream é útil por permitir operações simples de saída de dados em arquivos. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **FileOutputStream (String N):**
    Constroi uma instância da classe FileOutputStream para promover saída de dados no arquivo de nome N;
  - **FileOutputStream (File F):**
    Constroi uma instância da classe FileOutputStream para promover saída de dados no arquivo representado pelo objeto F da classe File.

A Classe java.io.StringOutputStream

Derivada de **OutputStream**, a classe StringOutputStream é útil por permitir operações simples de saída de dados em strings. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

- **StringOutputStream ():**
  Constroi uma instância da classe StringOutputStream para promover saída de dados em um String.

- **Métodos:**

- **String toString ():**
  Retorna um String com tudo o que foi escrito no StringOutputStream.

A Classe java.io.PrintStream

A classe PrintStream foi projetada primariamente para a produção de escrita de dados sob a forma de texto. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

- **PrintStream (OutputStream OS, boolean FA):**
  Constroi uma instância da classe PrintStream; OS representa uma instância da classe OutputStream; FA=true indica que desejamos flush automático do *stream* a cada linha escrita, e FA=false indica que desejamos deixar a encargo do objeto criado a decisão de quando esvaziar o buffer e efetivamente escrever os caracteres nele escritos;

- **PrintStream (OutputStream OS):**
  Constroi uma instância da classe PrintStream que opera como se a houvéssemos criado através do construtor anterior sendo o parâmetro FA=false;

- **Métodos:**

- **boolean checkError ():**
  Força o esvaziamento do buffer de saída deste PrintStream, fazendo-o efetivar toda e qualquer saída pendente, e verifica se ocorreu algum erro durante essa escrita;

- **void close ():**
  Fecha este PrintStream;

- **void flush ():**

  Força o esvaziamento do buffer de saída deste PrintStream, fazendo-o efetivar toda e qualquer saída pendente;

- **void print (boolean B):**

  Escreve um valor lógico (true ou false) neste PrintStream;

- **void print (byte B):**

  Escreve um byte neste PrintStream;

- **void print (char C):**

  Escreve um caractere neste PrintStream;

- **void print (char V []):**

  Escreve um vetor de caracteres neste PrintStream;

- **void print (double D):**

  Escreve um real de dupla precisão neste PrintStream;

- **void print (float F):**

  Escreve um real neste PrintStream;

- **void print (int I):**

  Escreve um inteiro neste PrintStream;

- **void print (long L):**

  Escreve um inteiro longo neste PrintStream;

- **void print (Object O):**

  Escreve neste PrintStream o retorno do método O.toString ();

- **void print (String S):**

  Escreve um String neste PrintStream;

- **void println ():**

  Escreve um newline neste PrintStream;

- **void println (boolean B):**

  Escreve um valor lógico (true ou false) seguido por um caractere '\n' neste PrintStream;

- **void println (byte B):**

  Escreve um byte seguido por um caractere '\n' neste PrintStream;

- **void println (char C):**

  Escreve um caractere seguido por um caractere '\n' neste PrintStream;

- **void println (char V []):**

  Escreve um vetor de caracteres seguido por um caractere '\n' neste PrintStream;

- **void println (double D):**

  Escreve um real de dupla precisão seguido por um caractere '\n' neste PrintStream;

- **void println (float F):**

  Escreve um real seguido por um caractere '\n' neste PrintStream;

- **void println (int I):**

  Escreve um inteiro seguido por um caractere '\n' neste PrintStream;

- **void println (long L):**

  Escreve um inteiro longo seguido por um caractere '\n' neste PrintStream;

- **void println (Object O):**

  Escreve neste PrintStream o retorno do método O.toString () seguido por um caractere '\n';

- **void println (String S):**

  Escreve um String seguido por um caractere '\n' neste PrintStream.

[C:\ExplsJava\Expl_02\TudoMaiusculo.java]

```
import java.io.*;

class TudoMaiusculo
{
    public static void main (String Args [])
```

```
    {
        BufferedReader Entrada = null;

        if (Args.length != 2)
        {
            System.err.println ();
            System.err.println ("USO CORRETO: " +
                                "java TudoMaiusculo ArqEntrada ArqSaida");
            return;
        }

        try
        {
            Entrada = new BufferedReader (new FileReader (Args [0]));
        }
        catch (IOException E)
        {
            System.err.println ();
            System.err.println ("O arquivo de entrada (" +
                                Args [0] +
                                ") nao existe!");
            System.err.println ();
            return;
        }

        try
        {
            if (!Entrada.ready ())
                return;
        }
        catch (Exception E)
        {}

        PrintStream Saida = null;

        try
        {
            Saida = new PrintStream (new FileOutputStream (Args [1]));

        }
        catch (IOException E)
        {
            System.err.println ();
            System.err.println ("O arquivo de saida (" +
                                Args [1] +
                                ") nao pode ser criado!");
            System.err.println ();
            return;
        }

        for (;;)
        {
            String Linha = null;

            try
            {
                Linha = Entrada.readLine ();
            }
            catch (IOException E)
            {
                System.err.println ();
                System.err.println ("Problemas na leitura " +
                                    "no arquivo de entrada (" +
```

```
                                    Args [0] +
                                    ")!");
                System.err.println ();
                return;
            }

            Linha = Linha.toUpperCase ();

            Saida.print (Linha);

            try
            {
                if (Entrada.ready ())
                    Saida.println ();
                else
                    break;
            }
            catch (IOException E)
            {}
        }

        try
        {
            Entrada.close ();
        }
        catch (IOException E)
        {
            System.err.println ();
            System.err.println ("O arquivo de entrada (" +
                                Args [0] +
                                ") nao pode ser fechado!");
            System.err.println ();
            return;
        }

        Saida.close ();
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_02> javac TudoMaiusculo.java
C:\ExplsJava\Expl_02> java TudoMaiusculo ORIGINAL.ARQ DESTINO.ARQ
```

Isto fará com que o arquivo ORIGINAL.ARQ seja copiado para o arquivo DESTINO.ARQ.

E/S Binária

A Classe java.io. RandomAccessFile

A classe java.io.RandomAccessFile implementa métodos que possibilitam a manutenção de dados em um arquivo de acesso aleatório. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

  - **RandomAccessFile (String N, String M):**
    Constroi uma instância da classe RandomAccessFile; N especifica o nome completo do arquivo (incluindo path) e M especifica o modo de abertura ("r" para leitura apenas, "w" para escrita apenas e "rw" para leitura e escrita);

  - **RandomAccessFile (File F, String M):**
    Funciona exatamente como o construtor acima, exceto pelo fato de que em vez de receber um String contendo o nome completo do arquivo, recebe um objeto da classe File que tem esta função;

- **Métodos:**

  - **void close ():**
    Fecha este RandomAccessFile;

  - **long getFilePointer ():**
    Retorna a posição corrente neste RandomAccessFile;

  - **long length ():**
    Retorna o tamanho em bytes do arquivo;

  - **int read ():**
    Lê um byte do RandomAccessFile e o retorna como um inteiro (retorna –1 se o fim do RandomAccessFile for alcançado);

  - **int read (byte V []):**
    Lê até V.length bytes e os posiciona no vetor V; bloqueia a continuidade do fluxo de execução até ter sido possível ler ao menos 1 byte; espera dados em formato binário, não em formato ASCII; retorna a quantidade de bytes lidos ou –1 em caso de fim do RandomAccessFile;

  - **int read (byte V [], int P, int Q):**
    Lê até Q bytes e os posiciona no vetor V, a partir da posição P; bloqueia a continuidade do fluxo de execução até ter sido possível ler ao menos 1 byte; espera

dados em formato binário, não em formato ASCII; retorna a quantidade de bytes lidos ou –1 em caso de fim do RandomAccessFile;

- **boolean readBoolean ():**
  Lê um valor lógico (true ou false); espera dados em formato binário, não em formato ASCII;

- **byte readByte ():**
  Lê um byte; espera dados em formato binário, não em formato ASCII;

- **char readChar ():**
  Lê um caractere (2 bytes, o mais significativo primeiro); espera dados em formato binário, não em formato ASCII;

- **double readDouble ():**
  Lê um real de dupla precisão; espera dados em formato binário, não em formato ASCII;

- **float readFloat ():**
  Lê um real; espera dados em formato binário, não em formato ASCII;

- **void readFully (byte V []):**
  Lê V.length bytes e os posiciona no vetor V; bloqueia a continuidade do fluxo de execução até ter sido possível ler V.length bytes; espera dados em formato binário, não em formato ASCII;

- **void readFully (byte V [], int P, int Q):**
  Lê até Q bytes e os posiciona no vetor V, a partir da posição P; bloqueia a continuidade do fluxo de execução até ter sido possível ler Q bytes; espera dados em formato binário, não em formato ASCII;

- **int readInt ():**
  Lê um inteiro; espera dados em formato binário, não em formato ASCII;

- **String readLine ():**
  Lê uma linha de texto; espera dados em formato ASCII;

- **long readLong ():**

  Lê um inteiro longo; espera dados em formato binário, não em formato ASCII;

- **short readShort ():**

  Lê um inteiro curto; espera dados em formato binário, não em formato ASCII;

- **int readUnsignedByte ():**

  Lê um inteiro sem sinal; espera dados em formato binário, não em formato ASCII;

- **int readUnsignedShort ():**

  Lê um inteiro curto sem sinal; espera dados em formato binário, não em formato ASCII;

- **void seek (long P):**

  Posiciona este RandomAccessFile no byte de número de ordem igual a P;

- **void setLength (long T):**

  Ajusta o tamanho deste RandomAccessFile para T bytes;

- **int skipBytes (int Q):**

  Este método despreza Q bytes do RandomAccessFile; bloqueia a continuidade do fluxo de execução até ter sido possível desprezar Q bytes pretendidos.

- **void write (byte V []):**

  Escreve múltiplos bytes; espera dados em formato binário, não em formato ASCII;

- **void write (byte V [], int P, int Q):**

  Escreve no máximo Q bytes do vetor V a partir da posição P; escreve dados em formato binário, não em formato ASCII;

- **void write (int B):**

  Escreve o byte menos significativos de B; escreve dados em formato binário, não em formato ASCII;

- **void writeBoolean (boolean B):**

  Escreve um valor lógico (true ou false); escreve dados em formato binário, não em formato ASCII;

- **void writeByte (byte B):**

  Escreve um byte; escreve dados em formato binário, não em formato ASCII;

- **void writeBytes (String S):**

  Escreve cada um dos caracteres de S, despresando seus 8 bits mais significativos; escreve dados em formato binário, não em formato ASCII;

- **void writeChar (char C):**

  Escreve um caractere (2 bytes, o byte mais significativo primeiro); escreve dados em formato binário, não em formato ASCII;

- **void writeChars (String S):**

  Escreve cada um dos caracteres de S (2 bytes, o byte mais significativo primeiro); escreve dados em formato binário, não em formato ASCII;

- **void writeDouble (double D):**

  Escreve um real de dupla precisão; escreve dados em formato binário, não em formato ASCII;

- **void writeFloat (float F):**

  Escreve um real; escreve dados em formato binário, não em formato ASCII;

- **void writeInt (int I):**

  Escreve um inteiro; escreve dados em formato binário, não em formato ASCII;

- **void writeLong (long L):**

  Escreve um inteiro longo; escreve dados em formato binário, não em formato ASCII;

- **void writeShort (short S):**

  Escreve um inteiro curto; escreve dados em formato binário, não em formato ASCII.

[C:\ExplsJava\Expl_03\BibAgenda\ExcecaoDeAgenda.java]

```
package BibAgenda;

public class ExcecaoDeAgenda extends Exception
{
    public static final int CAPACIDADE_INVALIDA   = 0,
                            AGENDA_CHEIA          = 1,
                            AGENDA_VAZIA          = 2,
```

```
                              NOME_INEXISTENTE      = 3,
                              NOME_JA_REGISTRADO    = 4,
                              FONE_FORA_DOS_LIMITES = 5,
                              NOME_FORA_DOS_LIMITES = 6,
                              EXCECAO_INVALIDA      = 7;

    private static final int MENOR_ID = CAPACIDADE_INVALIDA,
                              MAIOR_ID = NOME_FORA_DOS_LIMITES;

    public static final String Mensagem [] =
    {
         "Capacidade deve ser um numero natural positivo",
         "Esgotou-se a capacidade da agenda",
         "Nao ha contatos registrados na agenda",
         "O nome deste contato nao consta na agenda",
         "O nome deste contato ja consta na agenda",
         "So solicite telefones entre zero e a quantidade existente",
         "So solicite nomes entre zero e a quantidade existente",
         "Indicacao de excecao invalida"
    };


    private int Tipo;

    private ExcecaoDeAgenda (int T)
    {
        super (ExcecaoDeAgenda.Mensagem [T]);
        this.Tipo = T;
    }

    public static void indique (int T) throws ExcecaoDeAgenda
    {
        if (T<ExcecaoDeAgenda.MENOR_ID || T>ExcecaoDeAgenda.MAIOR_ID)
            throw new ExcecaoDeAgenda (ExcecaoDeAgenda.EXCECAO_INVALIDA);

        throw new ExcecaoDeAgenda (T);

    }

    public int seuTipo ()
    {
        return this.Tipo;
    }
}
```

[C:\ExplsJava\Expl_03\BibAgenda\Agenda.java]

```
package BibAgenda;

import java.io.*;

class Registro
{
    private static final int TAMANHO_NOME = 30;
    private static final int TAMANHO_TELEFONE = 13;

    public static final int TAMANHO = TAMANHO_NOME + TAMANHO_TELEFONE;

    private String Nome, Telefone;
```

```
    private static String replicacao (int Q, char C)
    {
        String R = "";

        for (int i=0; i<Q; i++)
            R = R + C;

        return R;
    }

    public Registro (String N, String T)
    {
        int TN        = Math.min(Registro.TAMANHO_NOME,N.length());
        this.Nome     = N.trim().substring(0,TN);
        int TF        = Math.min(Registro.TAMANHO_TELEFONE,T.length());
        this.Telefone = T.trim().substring(0,TF);
    }

    public Registro (byte B [])
    {
        this (new String (B, 0,                 Registro.TAMANHO_NOME),
              new String (B, Registro.TAMANHO_NOME,
Registro.TAMANHO_TELEFONE));
    }

    public String campoNome ()
    {
        return this.Nome;
    }

    public String campoTelefone ()
    {
       return this.Telefone;
    }

    public byte[] seusBytes ()
    {
        String R = this.Nome +
                   Registro.replicacao (
                   Registro.TAMANHO_NOME-this.Nome.length(), ' ') +
                   this.Telefone +
                   Registro.replicacao (
                   Registro.TAMANHO_TELEFONE-this.Telefone.length(), ' ');

        return R.getBytes();
    }
}

public class Agenda
{

    private final int TamQtsCts =  4;

    private RandomAccessFile Arq;

    private int ondeEsta (String N)
    {
        int Inicio = 0, Final = this.quantosContatos() - 1, Onde;

        while (Inicio <= Final)
        {
            Onde = (Inicio + Final) / 2;
```

```
            try
            {
                int Comparacao = N.compareTo (this.digaMeNome(Onde));

                if (Comparacao == 0)
                    return Onde;
                else
                    if (Comparacao < 0)
                        Final = Onde - 1;
                    else
                        Inicio = Onde + 1;
            }
            catch (ExcecaoDeAgenda E)
            {}
        }

        return -1;
    }

    private int ondeGuardar (String N)
    {
        int Inicio     = 0,
            Final      = this.quantosContatos() - 1,
            Onde       = 0,
            Comparacao = -1;

        while (Inicio <= Final)
        {
            Onde = (Inicio + Final) / 2;

            try
            {
                Comparacao = N.compareTo (this.digaMeNome(Onde));

                if (Comparacao == 0)
                    return -1;
                else
                    if (Comparacao < 0)
                        Final = Onde - 1;
                    else
                        Inicio = Onde + 1;
            }
            catch (ExcecaoDeAgenda E)
            {}
        }

        return Comparacao < 0? Onde: Onde + 1;
    }

    public Agenda (String NA)
    {

        try
        {
            File DescritorArq = new File (NA);

            if (!DescritorArq.exists ())
            {
                this.Arq = new RandomAccessFile (NA, "rw");
                this.Arq.writeInt (0);
                this.Arq.seek (0L);
            }
            else
```

```
                this.Arq = new RandomAccessFile (NA, "rw");
        }
        catch (IOException E)
        {}
    }

    public int quantosContatos ()
    {
        int R = 0;

        try
        {
            long FP = this.Arq.getFilePointer ();

            this.Arq.seek (0L);
            R = this.Arq.readInt ();

            this.Arq.seek (FP);
        }
        catch (IOException E)
        {}

        return R;
    }

    public boolean haRegistroDoContato (String N)
    {
        return this.ondeEsta (N) != -1;
    }

    public void registreOContato (String N, String T) throws ExcecaoDeAgenda
    {
        try
        {
            long FP = this.Arq.getFilePointer ();

            int Posicao = this.ondeGuardar (N);

            if (Posicao == -1)
                ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_JA_REGISTRADO);

            byte B [] = new byte [Registro.TAMANHO];

            for (int I = this.quantosContatos() - 1; I >= Posicao; I--)
            {
                this.Arq.seek  (I*Registro.TAMANHO + TamQtsCts);
                this.Arq.read  (B);
                this.Arq.write (B);
            }

            Registro R = new Registro (N,T);

            this.Arq.seek  (Posicao*Registro.TAMANHO + TamQtsCts);
            this.Arq.write (R.seusBytes());

            this.Arq.seek (0L);
            this.Arq.writeInt (this.quantosContatos()+1);

            this.Arq.seek (FP);
        }
        catch (IOException E)
        {}
    }
```

```
    public String digaMeNome (int P) throws ExcecaoDeAgenda
    {
        Registro R = null;

        try
        {
            long FP = this.Arq.getFilePointer ();

            int QtosContatos = this.quantosContatos ();

            if (QtosContatos == 0)
                ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_VAZIA);

            if (P < 0 || P >= QtosContatos)
                ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.NOME_FORA_DOS_LIMITES);

            byte B [] = new byte [Registro.TAMANHO];

            this.Arq.seek (P*Registro.TAMANHO + TamQtsCts);
            this.Arq.read (B);

            R = new Registro (B);

            this.Arq.seek (FP);
        }
        catch (IOException E)
        {}

        return R.campoNome();
    }

    public String digaMeTelefone (int P) throws ExcecaoDeAgenda
    {
        Registro R = null;

        try
        {
            long FP = this.Arq.getFilePointer ();


            int QtosContatos = this.quantosContatos ();

            if (QtosContatos == 0)
                ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_VAZIA);

            if (P < 0 || P >= QtosContatos)
                ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.FONE_FORA_DOS_LIMITES);

            byte B [] = new byte [Registro.TAMANHO];

            this.Arq.seek (P*Registro.TAMANHO + TamQtsCts);
            this.Arq.read (B);

            R = new Registro (B);

            this.Arq.seek (FP);
        }
        catch (IOException E)
        {}
```

```
        return R.campoTelefone();
    }

    public String digaMeTelefoneDe (String N) throws ExcecaoDeAgenda
    {
        String R = null;

        if (this.quantosContatos() == 0)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_VAZIA);

        int Posicao = this.ondeEsta (N);

        if (Posicao == -1)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_INEXISTENTE);

        try
        {
            R = this.digaMeTelefone (Posicao);;
        }
        catch (ExcecaoDeAgenda E)
        {}

        return R;
    }

    public void descarteContato (String N) throws ExcecaoDeAgenda
    {
        try
        {
            long FP = this.Arq.getFilePointer ();

            if (this.quantosContatos() == 0)
                ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_VAZIA);

            int Posicao = this.ondeEsta (N);

            if (Posicao == -1)
                ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_INEXISTENTE);

            int I;

            byte B [] = new byte [Registro.TAMANHO];

            for (I = Posicao; I < this.quantosContatos() - 1; I++)
            {
                this.Arq.seek ((I+1)*Registro.TAMANHO + TamQtsCts);
                this.Arq.read (B);

                this.Arq.seek  (I*Registro.TAMANHO + TamQtsCts);
                this.Arq.write (B);
            }

            this.Arq.setLength (this.Arq.length() - Registro.TAMANHO);

            this.Arq.seek (0L);
            this.Arq.writeInt (this.quantosContatos()-1);

            this.Arq.seek (FP);
        }
        catch (IOException E)
        {}
    }
}
```

[C:\ExplsJava\Expl_03\TesteDeAgenda.java]

```
import java.io.*;
import BibAgenda.*;

class TesteDeAgenda
{
    public static void main (String Args [])
    {
        BufferedReader Entrada = new BufferedReader
                                 (new InputStreamReader
                                 (System.in));

        String NomeArq = null;

        System.out.println ();

        System.out.print ("Nome da Agenda: ");
        try
        {
            NomeArq = Entrada.readLine ();
        }
        catch (IOException E)
        {}

        Agenda agenda = new Agenda (NomeArq);

        System.out.println ();

        String Nome = null, Telefone = null;

        char Opcao = ' ';

        do
        {
            System.out.println ();

            System.out.print ("Digite sua Opcao (" +
                              "I=Incluir/" +
                              "C=Consultar/" +
                              "L=Listar/" +
                              "E=Excluir/" +
                              "S=Sair)" +
                              ": ");

            try
            {
                Opcao = (Entrada.readLine ()).charAt (0);
            }
            catch (IOException E)
            {}

            switch (Opcao)
            {
                case 'i':
                case 'I':
                    try
                    {
                        System.out.print ("Nome....: ");
```

```
                    Nome = Entrada.readLine ();

                    System.out.print ("Telefone: ");
                    Telefone = Entrada.readLine ();
                }
                catch (IOException E)
                {}

                try
                {
                    agenda.registreOContato (Nome, Telefone);
                }
                catch (ExcecaoDeAgenda E)
                {
                    // certamente sera
                    // ExcecaoDeAgenda.NOME_JA_REGISTRADO

                    System.err.println (E.getMessage ());
                    System.err.println ();
                    break;
                }

                System.out.println ();
                break;


            case 'c':
            case 'C':
                try
                {
                    System.out.print ("Nome....: ");
                    Nome = Entrada.readLine ();
                }
                catch (IOException E)
                {}

                try
                {
                    Telefone = agenda.digaMeTelefoneDe (Nome);
                }
                catch (ExcecaoDeAgenda E)
                {
                    System.err.println (E.getMessage ());
                    System.err.println ();
                    break;
                }

                System.out.println ("Telefone: " + Telefone);
                System.out.println ();
                break;


            case 'l':
            case 'L':
                String Tecla;

                for (int I = 0;; I++)
                {
                    try
                    {
                        System.out.println ("Nome: " +
                                            agenda.digaMeNome (I) +
                                            " - Telefone: " +
```

```
                                                agenda.digaMeTelefone (I));
                        }
                        catch (ExcecaoDeAgenda E)
                        {
                            if (E.seuTipo () == ExcecaoDeAgenda.AGENDA_VAZIA)
                                System.err.println (E.getMessage ());

                            break;
                        }

                        if (I%23 == 22 && I < agenda.quantosContatos()-1)
                        {
                            System.out.println ();
                            System.out.print ("Tecle ENTER... ");

                            try
                            {
                                Tecla = Entrada.readLine ();
                            }
                            catch (IOException E)
                            {}

                            System.out.println ();
                        }
                    }

                    System.out.println ();
                    break;


                case 'e':
                case 'E':
                    System.out.print ("Nome....: ");
                    try
                    {
                        Nome = Entrada.readLine ();
                    }
                    catch (IOException E)
                    {}

                    try
                    {
                        agenda.descarteContato (Nome);
                    }
                    catch (ExcecaoDeAgenda E)
                    {
                        System.err.println (E.getMessage ());
                    }

                    System.out.println ();
                    break;


                case 's':
                case 'S':
                    break;


                default :
                    System.err.println ("Opcao invalida!");
                    System.err.println ();
            }
        }
```

```
        while ((Opcao != 's') && (Opcao != 'S'));
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_03> javac -classpath . TesteDeAgenda.java
C:\ExplsJava\Expl_03> java -classpath . TesteDeAgenda
```

Isto poderá produzir no console a seguinte saída:

```
Nome da Agenda: Agenda.dat

Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Jose
Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Jose
Esse nome ja consta na agenda!


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): e
Nome....: Joao
Nao consta na agenda!


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): c
Nome....: Jose
Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Joao
Telefone: 202.9955


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): l
Nome: Joao - Telefone: 202.9955
Nome: Jose - Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): s
```

Objetos Persistentes

Para poder ser lido e escrito, as classes dos objetos persistentes devem implementar a interface Serializable.

A Classe java.io.FileInputStream

Derivada de **InputStream**, a classe FileInputStream é útil por permitir operações simples de entrada de dados em arquivos. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

– **FileInputStream (String N):**
  Constroi uma instância da classe FileInputStream para promover entrada de dados no arquivo de nome N;

– **FileInputStream (File F):**
  Constroi uma instância da classe FileInputStream para promover entrada de dados no arquivo representado pelo objeto F da classe File.

A Classe java.io. ObjectInputStream

A classe java.io.ObjectInputStream implementa métodos que possibilitam a manutenção de objetos em um arquivo de acesso seqüencial. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

  – **ObjectInputStream (InputStream IS):**
    Constroi uma instância da classe ObjectInputStream; IS representa uma instância da classe InputStream;

- **Métodos:**

  – **int available ():**
    Retorna a quantidade de byte que podem ser lidos deste ObjectInputStream;

  – **void close ():**
    Fecha este ObjectInputStream;

- **int read ():**

  Lê um byte do ObjectInputStream e o retorna como um inteiro (retorna –1 se o fim do InputStream for alcançado);

- **int read (byte V [], int P, int Q):**

  Lê até Q bytes e os posiciona no vetor V, a partir da posição P; bloqueia a continuidade do fluxo de execução até ter sido possível ler ao menos 1 byte; espera dados em formato binário, não em formato ASCII; retorna a quantidade de bytes lidos ou –1 em caso de fim do InputStream;

- **boolean readBoolean ():**

  Lê um valor lógico (true ou false); espera dados em formato binário, não em formato ASCII;

- **byte readByte ():**

  Lê um byte; espera dados em formato binário, não em formato ASCII;

- **char readChar ():**

  Lê um caractere (2 bytes, o mais significativo primeiro); espera dados em formato binário, não em formato ASCII;

- **double readDouble ():**

  Lê um real de dupla precisão; espera dados em formato binário, não em formato ASCII;

- **float readFloat ():**

  Lê um real; espera dados em formato binário, não em formato ASCII;

- **void readFully (byte V []):**

  Lê V.length bytes e os posiciona no vetor V; bloqueia a continuidade do fluxo de execução até ter sido possível ler V.length bytes; espera dados em formato binário, não em formato ASCII;

- **void readFully (byte V [], int P, int Q):**

  Lê até Q bytes e os posiciona no vetor V, a partir da posição P; bloqueia a

continuidade do fluxo de execução até ter sido possível ler Q bytes; espera dados em formato binário, não em formato ASCII;

- **int readInt ():**
  Lê um inteiro; espera dados em formato binário, não em formato ASCII;
- **long readLong ():**
  Lê um inteiro longo; espera dados em formato binário, não em formato ASCII;
- **Object readObject ():**
  Lê uma instancia; espera dados em formato binário, não em formato ASCII;
- **short readShort ():**
  Lê um inteiro curto; espera dados em formato binário, não em formato ASCII;
- **int readUnsignedByte ():**
  Lê um inteiro sem sinal; espera dados em formato binário, não em formato ASCII;
- **int readUnsignedShort ():**
  Lê um inteiro curto sem sinal; espera dados em formato binário, não em formato ASCII;
- **int readUTF ():**
  Lê um string de formatação UTF;
- **int skipBytes (int Q):**
  Despreza, no máximo, Q bytes deste ObjectInputStream; retorna a quantidade de caracteres efetivamente desprezada.

A Classe java.io.FileOutputStream

Derivada de **OutputStream**, a classe FileOutputStream é útil por permitir operações simples de saida de dados em arquivos. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

- **FileInputStream (String N):**
  Constroi uma instância da classe FileInputStream para promover entrada de dados no arquivo de nome N;

- **FileInputStream (File F):**
  Constroi uma instância da classe FileInputStream para promover entrada de dados no arquivo representado pelo objeto F da classe File.

A Classe java.io. ObjectOutputStream

A classe java.io.ObjectOutputStream implementa métodos que possibilitam a manutenção de objetos em um arquivo de acesso seqüencial. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **ObjectOutputStream (OutputStream IS):**
    Constroi uma instância da classe ObjectInputStream; IS representa uma instância da classe InputStream;

- **Métodos:**
  - **void close ():**
    Fecha este ObjectOutputStream;
  - **void flush ():**
    Força o esvaziamento do buffer de saída deste ObjectOutpuStream, fazendo-o efetivar toda e qualquer saída pendente;
  - **void write (byte V []):**
    Escreve múltiplos bytes; espera dados em formato binário, não em formato ASCII;
  - **void write (byte V [], int P, int Q):**
    Escreve no máximo Q bytes do vetor V a partir da posição P; escreve dados em formato binário, não em formato ASCII;

- **void write (int B):**

  Escreve o byte menos significativos de B; escreve dados em formato binário, não em formato ASCII.

- **void writeBoolean (boolean B):**

  Escreve um valor lógico (true ou false); escreve dados em formato binário, não em formato ASCII;

- **void writeByte (byte B):**

  Escreve um byte; escreve dados em formato binário, não em formato ASCII;

- **void writeBytes (String S):**

  Escreve cada um dos caracteres de S, despresando seus 8 bits mais significativos; escreve dados em formato binário, não em formato ASCII;

- **void writeChar (char C):**

  Escreve um caractere (2 bytes, o byte mais significativo primeiro); escreve dados em formato binário, não em formato ASCII;

- **void writeChars (String S):**

  Escreve cada um dos caracteres de S (2 bytes, o byte mais significativo primeiro); escreve dados em formato binário, não em formato ASCII;

- **void writeDouble (double D):**

  Escreve um real de dupla precisão; escreve dados em formato binário, não em formato ASCII;

- **void writeFloat (float F):**

  Escreve um real; escreve dados em formato binário, não em formato ASCII;

- **void writeInt (int I):**

  Escreve um inteiro; escreve dados em formato binário, não em formato ASCII;

- **void writeLong (long L):**

  Escreve um inteiro longo; escreve dados em formato binário, não em formato ASCII;

- **void writeObject (Object O):**

  Escreve uma instancia; espera dados em formato binário, não em formato ASCII;

- **void writeShort (short S):**

  Escreve um inteiro curto; escreve dados em formato binário, não em formato ASCII.

- **void writeUTF ():**

  Escreve um string de formatação UTF.

[C:\ExplsJava\Expl_04\BibAgenda\ExcecaoDeAgenda.java]

```
package BibAgenda;

public class ExcecaoDeAgenda extends Exception
{
    public static final int CAPACIDADE_INVALIDA   = 0,
                            AGENDA_CHEIA          = 1,
                            AGENDA_VAZIA          = 2,
                            NOME_INEXISTENTE      = 3,
                            NOME_JA_REGISTRADO    = 4,
                            FONE_FORA_DOS_LIMITES = 5,
                            NOME_FORA_DOS_LIMITES = 6,
                            EXCECAO_INVALIDA      = 7;

    private static final int MENOR_ID = CAPACIDADE_INVALIDA,
                             MAIOR_ID = NOME_FORA_DOS_LIMITES;

    public static final String Mensagem [] =
    {
         "Capacidade deve ser um numero natural positivo",
         "Esgotou-se a capacidade da agenda",
         "Nao ha contatos registrados na agenda",
         "O nome deste contato nao consta na agenda",
         "O nome deste contato ja consta na agenda",
         "So solicite telefones entre zero e a quantidade existente",
         "So solicite nomes entre zero e a quantidade existente",
         "Indicacao de excecao invalida"
    };


    private int Tipo;

    private ExcecaoDeAgenda (int T)
    {
        super (ExcecaoDeAgenda.Mensagem [T]);
        this.Tipo = T;
    }

    public static void indique (int T) throws ExcecaoDeAgenda
    {
        if (T<ExcecaoDeAgenda.MENOR_ID || T>ExcecaoDeAgenda.MAIOR_ID)
            throw new ExcecaoDeAgenda (ExcecaoDeAgenda.EXCECAO_INVALIDA);

        throw new ExcecaoDeAgenda (T);
```

```
    }

    public int seuTipo ()
    {
        return this.Tipo;
    }
}
```

[C:\ExplsJava\Expl_04\BibAgenda\Agenda.java]

```
package BibAgenda;

import java.io.Serializable;

public class Agenda implements Serializable
{
    private int QtosContatos = 0;

    private String  Nome     [],
                    Telefone [];

    private int ondeEsta (String N)
    {
        int Inicio = 0, Final = this.QtosContatos - 1, Onde;

        while (Inicio <= Final)
        {
            Onde = (Inicio + Final) / 2;

            int Comparacao = N.compareTo (this.Nome [Onde]);

            if (Comparacao == 0)
                return Onde;
            else
                if (Comparacao < 0)
                    Final = Onde - 1;
                else
                    Inicio = Onde + 1;
        }

        return -1;
    }

    private int ondeGuardar (String N)
    {
        int Inicio     = 0,
            Final      = this.QtosContatos - 1,
            Onde       = 0,
            Comparacao = -1;

        while (Inicio <= Final)
        {
            Onde       = (Inicio + Final) / 2;
            Comparacao = N.compareTo (this.Nome [Onde]);

            if (Comparacao == 0)
                return -1;
            else
                if (Comparacao < 0)
                    Final = Onde - 1;
                else
```

```
                    Inicio = Onde + 1;
        }

        return Comparacao < 0? Onde: Onde + 1;
    }

    public Agenda (int C) throws ExcecaoDeAgenda
    {
        if (C <= 0)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.CAPACIDADE_INVALIDA);

        this.Nome     = new String [C];
        this.Telefone = new String [C];
    }

    public int capacidade ()
    {
        return this.Nome.length;
    }

    public int quantosContatos ()
    {
        return this.QtosContatos;
    }

    public boolean haRegistroDoContato (String N)
    {
        return this.ondeEsta (N) != -1;
    }

    public void registreOContato (String N, String T) throws ExcecaoDeAgenda
    {
        if (this.QtosContatos == this.Nome.length)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_CHEIA);

        int Posicao = this.ondeGuardar (N);

        if (Posicao == -1)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_JA_REGISTRADO);

        for (int I = this.QtosContatos - 1; I >= Posicao; I--)
        {
            this.Nome     [I+1] = this.Nome     [I];
            this.Telefone [I+1] = this.Telefone [I];
        }

        this.Nome     [Posicao] = N;
        this.Telefone [Posicao] = T;

        this.QtosContatos++;
    }

    public String digaMeONome (int P) throws ExcecaoDeAgenda
    {
        if (this.QtosContatos == 0)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_VAZIA);

        if (P < 0 || P >= this.QtosContatos)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_FORA_DOS_LIMITES);

        return this.Nome [P];
    }
```

```
    public String digaMeOTelefone (int P) throws ExcecaoDeAgenda
    {
        if (this.QtosContatos == 0)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_VAZIA);

        if (P < 0 || P >= this.QtosContatos)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.FONE_FORA_DOS_LIMITES);

        return this.Telefone [P];
    }

    public String digaMeOTelefoneDe (String N) throws ExcecaoDeAgenda
    {
        if (this.QtosContatos == 0)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_VAZIA);

        int Posicao = this.ondeEsta (N);

        if (Posicao == -1)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_INEXISTENTE);

        return this.Telefone [Posicao];
    }

    public void descarteContato (String N) throws ExcecaoDeAgenda
    {
        if (this.QtosContatos == 0)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.AGENDA_VAZIA);

        int Posicao = this.ondeEsta (N);

        if (Posicao == -1)
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_INEXISTENTE);

        int I;

        for (I = Posicao; I < this.QtosContatos - 1; I++)
        {
            this.Nome     [I] = this.Nome     [I+1];
            this.Telefone [I] = this.Telefone [I+1];
        }

        this.Nome     [I] = null;
        this.Telefone [I] = null;

        this.QtosContatos--;
    }
}
```

[C:\ExplsJava\Expl_04\TesteDeAgenda.java]

```
import java.io.*;
import BibAgenda.*;

class TesteDeAgenda
{
    public static void main (String Args [])
    {
        BufferedReader Entrada = new BufferedReader
                                 (new InputStreamReader
```

```
                                (System.in));

        String NomeArq = null;

        System.out.println ();

        System.out.print ("Nome da Agenda: ");
        try
        {
            NomeArq = Entrada.readLine ();
        }
        catch (IOException E)
        {}

        Agenda agenda = null;

        try
        {
            ObjectInputStream Ent = new ObjectInputStream
                                    (new FileInputStream
                                    (NomeArq));

            agenda = (Agenda) Ent.readObject ();

            Ent.close ();
        }
        catch (Throwable E)
        {
            for (;;)
            {
                System.out.println ();

                System.out.print ("Capacidade desejada para a Agenda: ");
               try
                {
                    int C  = Integer.parseInt (Entrada.readLine ());
                    agenda = new Agenda (C);
                }
                catch (IOException e)
                {}
                catch (NumberFormatException e)
                {
                    System.err.println ("Nao foi digitado um numero inteiro");
                    System.err.println ();
                    continue;
                }
                catch (ExcecaoDeAgenda e)
                {
                    System.err.println (E.getMessage());
                    System.err.println ();
                    continue;
                }

                break;
            }
        }

        System.out.println ();

        String Nome = null, Telefone = null;

        char Opcao = ' ';
```

```
        do
        {
            System.out.println ();

            System.out.print ("Digite sua Opcao (" +
                              "I=Incluir/" +
                              "C=Consultar/" +
                              "L=Listar/" +
                              "E=Excluir/" +
                              "S=Sair)" +
                              ": ");

            try
            {
                Opcao = (Entrada.readLine ()).charAt (0);
            }
            catch (IOException E)
            {}

    Opcoes: switch (Opcao)
            {
                case 'i':
                case 'I':
                    if (agenda.quantosContatos () == agenda.capacidade ())
                    {
                        System.err.println
(ExcecaoDeAgenda.Mensagem[ExcecaoDeAgenda.AGENDA_CHEIA]);
                        System.err.println ();
                        break;
                    }

                    try
                    {
                        System.out.print ("Nome....: ");
                        Nome = Entrada.readLine ();
                    }
                    catch (IOException E)

                    {}

                    if (agenda.haRegistroDoContato (Nome))
                    {
                        System.err.println
(ExcecaoDeAgenda.Mensagem[ExcecaoDeAgenda.NOME_JA_REGISTRADO]);
                        System.err.println ();
                        break;
                    }

                    try
                    {
                        System.out.print ("Telefone: ");
                        Telefone = Entrada.readLine ();
                    }
                    catch (IOException E)
                    {}

                    try
                    {
                        agenda.registreOContato (Nome, Telefone);
                    }
                    catch (ExcecaoDeAgenda E)
```

```
                    {}

                    System.out.println ();
                    break;


                case 'c':
                case 'C':
                    if (agenda.quantosContatos () == 0)
                    {
                        System.err.println

(ExcecaoDeAgenda.Mensagem[ExcecaoDeAgenda.AGENDA_VAZIA]);
                        System.err.println ();
                        break;
                    }

                    try

                    {
                        System.out.print ("Nome....: ");
                        Nome = Entrada.readLine ();
                    }
                    catch (IOException E)
                    {}

                    try
                    {
                        Telefone = agenda.digaMeOTelefoneDe (Nome);
                    }
                    catch (ExcecaoDeAgenda E)
                    {
                        System.err.println

(ExcecaoDeAgenda.Mensagem[ExcecaoDeAgenda.NOME_INEXISTENTE]);
                        System.err.println ();
                        break;
                    }

                    System.out.println ("Telefone: " + Telefone);
                    System.out.println ();
                    break;


                case 'l':
                case 'L':
                    String Tecla;

        Exibicao: for (int I = 0;; I++)
                    {
                        try
                        {
                            System.out.println ("Nome: " +
                                                agenda.digaMeONome (I) +
                                                " - Telefone: " +
                                                agenda.digaMeOTelefone (I));
                        }
                        catch (ExcecaoDeAgenda E)
                        {
                            switch (E.seuTipo ())
                            {
                                case ExcecaoDeAgenda.AGENDA_VAZIA:
                                    System.err.println (E.getMessage ());
```

```
                                    System.err.println ();
                                    break Opcoes;

                                case ExcecaoDeAgenda.NOME_FORA_DOS_LIMITES:
                                    break Exibicao;
                            }
                        }

                        if (I%23 == 22 && I < agenda.quantosContatos()-1)
                        {
                            System.out.println ();
                            System.out.print ("Tecle ENTER... ");

                            try
                            {
                                Tecla = Entrada.readLine ();
                            }
                            catch (IOException E)
                            {}

                            System.out.println ();
                        }
                    }

                    System.out.println ();
                    break;


                case 'e':
                case 'E':
                    if (agenda.quantosContatos () == 0)
                        System.err.println

(ExcecaoDeAgenda.Mensagem[ExcecaoDeAgenda.AGENDA_VAZIA]);

                    else
                    {
                          System.out.print ("Nome....: ");
                          try
                          {
                              Nome = Entrada.readLine ();
                          }
                          catch (IOException E)
                          {}

                          try
                          {
                              agenda.descarteContato (Nome);
                          }
                          catch (ExcecaoDeAgenda E)
                          {
                              // certamente sera
                              // ExcecaoDeAgenda.NOME_INEXISTENTE

                              System.err.println (E.getMessage ());

                          }
                    }

                    System.out.println ();
                    break;
```

```
                case 's':
                case 'S':
                    break;


                default :
                    System.err.println ("Opcao invalida");
                    System.err.println ();
            }
        }
        while ((Opcao != 's') && (Opcao != 'S'));

        try
        {
            ObjectOutputStream Sai = new ObjectOutputStream
                                     (new FileOutputStream
                                     (NomeArq));

            Sai.writeObject (agenda);
            Sai.close ();
        }
        catch (IOException E)
        {
            System.err.println ("Nao foi possivel gravar a agenda!");
            System.err.println ();
        }
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_04> javac -classpath . TesteDeAgenda.java
C:\ExplsJava\Expl_04> java -classpath . TesteDeAgenda
```

Isto poderá produzir no console a seguinte saída:

```
Nome da Agenda: Agenda.dat

Capacidade desejada para a Agenda: 100

Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Jose
Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Jose
Esse nome ja consta na agenda!


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): e
Nome....: Joao
Nao consta na agenda!
```

```
Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): c
Nome....: Jose
Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Joao
Telefone: 202.9955


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): l
Nome: Joao - Telefone: 202.9955
Nome: Jose - Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): s
```

Acesso a Bancos de Dados (O Pacote java.sql)

Entendemos, por Banco de Dados, um depósito de dados armazenados, em geral, de forma integrada e compartilhada.

Podemos imaginar um banco de dados como sendo a unificação de diversos arquivos que de outra forma seriam distinguíveis, eliminando total ou parcialmente (mas de qualquer forma mantendo sob controle) qualquer redundância entre aqueles arquivos.

Os dados em um banco de dados podem em geral ser compartilhados entre diversas aplicações e usuários diferentes.

A Classe java.sql.DriverManager

A classe java.sql.DriverManager implementa métodos que possibilitam a manutenção de drivers de acesso a bancos de dados. Veja abaixo a interface que a classe especifica para comunicação com ela própria:

- **Métodos:**

- **static void deregisterDriver (Driver D):**
  Remove um driver da lista de drivers do DriverManager;
- **static Connection getConnection (String URL):**
  Tenta estabelecer uma conexão com um banco de dados;
- **static Connection getConnection (String URL, Properties Info):**
  Tenta estabelecer uma conexão com um banco de dados;
- **static Connection getConnection (String URL, String Usr, String Pwd):**
  Tenta estabelecer uma conexão com um banco de dados;
- **static Driver getDriver (String URL):**
  Tenta localizar um driver que compreenda a URL dada;
- **static Enumeration getDrivers ():**
  Resulta uma instância da interface Enumeration com todos os drivers JDBC carregados aos quais se tenha acesso;
- **static int getLoginTimeout ():**
  Resulta o tempo máximo em segundos que um driver pode esperar quando tentando conexão com um banco de dados;
- **static PrintWriter getLogWriter ():**
  Resulta uma instância da classe PrintWriter que representa o registrador de Log do DriverManager;
- **static void println (String Msg):**
  Escreve a mensagem Msg no restriador de Log do DriverManager;
- **static void registerDriver (Driver D):**
  Registra o driver dado no DriverManager;
- **static void setLoginTimeout (int T):**
  Ajusta o tempo máximo em segundos que um driver pode esperar quando tentando conexão com um banco de dados;

- **static void setLogWriter (PrintWriter P):**
  Torna P o registrador de Log do DriverManager.

A Interface java.sql.Connection

Esta interface define o comportamento de um gerenciador de conexões genérico. Veja abaixo a como se comunicar com instâncias das classes que a implementam:

Campos:

Dizemos que ocorreu uma leitura "suja" quando uma transação altera uma linha e outra transação lê aquela linha, antes mesmo da primeira transação ter sido confirmada. Caso a primeira transação seja desfeita, a segunda transação teria recuperado uma linha inválida.

Dizemos que ocorreu uma leitura não repetitível quando uma transação lê uma linha, uma segunda transação altera aquela linha e a primeira transação lê novamente aquela linha, obtendo valores diferentes na segunda leitura.

Dizemos que ocorreu uma leitura fantasma quando uma transação recupera um conjunto de linhas, uma segunda transação insere uma nova linha e a primeira transação recupera novamente o conjunto de dados que já havia recuperado uma vez, encontrando nele a nova linha inserida pela segunda transação.

- **static int TRANSACTION_NONE:**
  Indica que transações não são suportadas;
- **static int TRANSACTION_READ_COMMITTED:**
  Previne leituras "sujas"; leituras não repetitíveis e leituras fantasma podem acontecer;
- **static int TRANSACTION_READ_UNCOMMITTED:**
  Leituras "sujas", leituras não repetitíveis e leituras fantasma podem acontecer;
- **static int TRANSACTION_REPEATABLE_READ:**
  Previne leituras "sujas" e leituras não repetitíveis; leituras fantasma podem acontecer;
- **static int TRANSACTION_SERIALIZABLE :**
  Previne leituras "sujas", leituras não repetitíveis e leituras fantasma;

- **Métodos:**
  - **void clearWarning ():**
    Limpa todas as advertências reportadas por esta conexão;
  - **void close ():**
    Desfaz uma conexão com um banco de dados, liberando os recursos da JDBC imediatamente, em vez de esperar que eles sejam automaticamente liberados;
  - **void commit ():**
    Efetiva todas as mudanças feitas desde o último commit/rollback e libera quaisquer locks do banco de dados que estivessem sendo mantidos por esta conexão;
  - **Statement createStatement ():**
    Cria uma instância da classe Statement, através da qual, comandos SQL podem ser enviados ao banco de dados com o qual esta conexão conecta;
  - **boolean getAutoCommit ():**
    Resulta true, caso Auto Commit esteja ativado, e false, caso contrário;
  - **String getCatalogName ():**
    Resulta o nome do catálogo associado a esta conexão
  - **DataBaseMetaData getMetaData ():**
    Resulta os metadados associados ao banco de dados relativo a esta conexão;
  - **Int getTransactionIsolation ():**
    Retorna o nível atual de isolamento de transações (um dos campos acima mencionados);
  - **Map getTypeMap ():**
    JDBC recupera o mapa de tipos associado a esta conexão;
  - **SQLWarning getWarning ():**
    Retorna a primeira advertência reportada por esta conexão;
  - **boolean isClosed ():**
    Retorna true, se esta conexão estiver fechada, e false, caso contrário;

- **boolean isReadOnly ():**
  Retorna true, se esta conexão for de leitura apenas, e false, caso contrário;

- **String nativeSQL (String SQL):**
  Converte o comando SQL fornecido para a sintaxe do SGBD ao qual se conecta esta conexão;

- **CallableStatement prepareCall (String SQL):**
  Cria uma instância da classe CallableStatement que pode ser usado para invocar procedimentos armazenados no banco de dados ao qual se conecta esta conexão;

- **CallableStatement prepareCall (String SQL, int RST, int RSC):**
  Cria uma instância da classe CallableStatement que pode ser usado para invocar procedimentos armazenados no banco de dados ao qual se conecta esta conexão com o tipo de conjunto resultado dado por RST e com o tipo de concorrência do conjunto resultado dado por RSC;

- **PreparedStatement prepareStatement (String SQL):**
  Cria uma instância da classe PreparedStatement que pode ser empregada para enviar comandos SQL parametrizados para o banco de dados ao qual se conecta esta conexão;

- **PreparedStatement prepareStatement (String SQL, int RST, int RSC):**
  Cria uma instância da classe PreparedStatement que pode ser empregada para enviar comandos SQL parametrizados para o banco de dados ao qual se conecta esta conexão com o tipo de conjunto resultado dado por RST e com o tipo de concorrência do conjunto resultado dado por RSC;

- **void rollback ():**
  Desfaz todas as alterações feitas no banco de dados desde o último commit/rollback e libera quaisquer locks do banco de dados que estivessem sendo mantidos por esta conexão;

- **void setAutoCommit (boolean E):**
  Habilita ou desabilita o Auto Commit;

- **void setCatalog (String C):**
  Seleciona um catálogo, i.e., um sub espaço do banco de dados ao qual se conecta esta conexão, no qual trabalhar;

- **void setReadOnly (boolean RO):**
  Habilita ou desabilita o uso exclusivo de operações de leitura no banco de dados ao qual se conecta esta conexão;

- **void setTransactionIsolation (int TI):**
  Ajusta o nível atual de isolamento de transações (TI deve ser um dos campos acima mencionados);

- **void setTypeMap (Map M):**
  JDBC instala o mapa de tipos dado, tornando-o o mapa de tipos desta conexão.

A Interface java.sql.Statement

Esta interface define o comportamento de instâncias que permitem dar comandos SQL, através de uma conexão, a um banco de dados e obter o resultado desses comandos. Veja abaixo a como se comunicar com instâncias das classes que a implementam:

- **Métodos:**

  - **`void addBatch (String SQL):`**
    Adiciona o comando SQL ao lote de comandos a ser executado por este Statement;

  - **`void cancel ():`**
    Cancela este Statement, se ambos, o SGBD e o driver suportarem cancelamentos de comandos SQL;

  - **`void clearBatch ():`**
    Torna vazio o lote de comandos a ser executado por este Statement;

  - **void clearWarning ():**
    Limpa todas as advertências reportadas sobre este Statement;

- **void close ():**

  Libera imediatamente os recursos alocados a este Statement em vez de esperar que eles sejam liberados automaticamente;

- **boolean execute (String SQL):**

  Executa um comando SQL que pode retornar muitos resultados (normalmente, um procedimento armazenado);

- **int[] executeBatch ():**

  Submete um lote de comandos à execução;

- **ResultSet executeQuery (String SQL):**

  Executa um comando SQL que retorna um único resultado;

- **int ExecuteUpdate (String SQL):**

  Executa um comando INSERT, UPDATE ou DELETE;

- **Connection getConnection ():**

  Retorna a instância da classe Connection que produziu este Statement;

- **int getFetchDirection ():**

  Retorna a direção padrão usada na a busca de linhas ResultSets produzidos por este Statement;

- **int getFetchSize ():**

  Retorna a quantidade de linhas que devem constituir uma busca qualquer de linhas em ResultSets produzidos por este Statement;

- **int getMaxFieldSize ():**

  Retorna o tamanho máximo, em bytes, do valor de uma coluna qualquer;

- **int getMaxRows ():**

  Retorna a quantidade máxima que uma instância da classe ResultSet pode conter;

- **boolean getMoreResults ():**

  Posiciona no próximo resultado deste Statement;

- **int getQueryTimeout ():**

  Retorna o tempo máximo, em segundo, que o driver aguarda pela execução de um Statement;

- **ResultSet getResultSet ():**

  Retorna o resultado corrente deste Statement na forma de uma instância da classe ResultSet;

- **int getResultSetConcurrency ():**

  Retorna a concorrência do ResultSet que é o resultado corrente deste Statement;

- **int getResultSetType ():**

  Retorna o tipo do ResultSet que é o resultado corrente deste Statement;

- **int getUpdateCount ():**

  Retorna o resultado corrente deste Statement na forma de uma contagem das tuplas afetadadas por este Statement;

- **SQLWarning getWarning ():**

  Retorna a primeira advertência reportada pela execução deste Statement;

- **void setCursorName (String N):**

  Define o nome do cursor que será usado por subseqüentes métodos de execução deste Statement;

- **void setEscapeProcessing (boolean EP):**

  Habilita ou desabilita o processamento de ESCs antes de solicitar a execução de comandos SQL;

- **int setFetchDirection ():**

  Ajusta a direção padrão a ser usada na a busca de linhas ResultSets produzidos por este Statement;

- **int setFetchSize ():**

  Ajusta a quantidade de linhas que devem constituir uma busca qualquer de linhas em ResultSets produzidos por este Statement;

- **int setMaxFieldSize ():**

  Ajusta o tamanho máximo, em bytes, do valor de uma coluna qualquer;

- **int setMaxRows ():**

  Ajusta a quantidade máxima que uma instância da classe ResultSet pode conter;

- **int setQueryTimeout ():**

  Ajusta o tempo máximo, em segundo, que o driver aguarda pela execução de um Statement;

A Interface java.sql.ResultSet

Esta interface define o comportamento de instâncias que permitem dar comandos SQL, através de uma conexão, a um banco de dados e obter o resultado desses comandos. Veja abaixo a como se comunicar com instâncias das classes que a implementam:

- **Campos:**

  - **`static int CONCUR_READ_ONLY:`**

    Indica impossibilidade de atualização;

  - **`static int CONCUR_UPDATABLE:`**

    Indica possibilidade de atualização;

  - **`static int FETCH_FORWARD:`**

    Indica que as linhas serão processadas da primeira para a última;

  - **`static int FETCH_REVERSE:`**

    Indica que as linhas serão processadas da última para a primeira;

  - **`static int FETCH_UNKNOWN:`**

    Indica que as linhas serão processadas em uma ordem desconhecida;

  - **`static int TYPE_FORWARD_ONLY:`**

    Indica que o cursor pode se mover apenas para frente;

- **`static int TYPE_SCROLL_INSENSITIVE:`**

  Indica que o cursor pode se mover para frente e para trás, mas sem sofrer a influência da movimentação de outros cursores;

- **`static int TYPE_SCROLL_SENSITIVE:`**

  Indica que o cursor pode se mover para frente e para trás, possivelmente sofrendo a influência da movimentação de outros cursores;

- **Métodos:**

  - **`boolean absolute (int L):`**

    Posiciona o cursor na linha L, se L for positivo, e L linhas antes do final deste ResultSet, se L for negativo;

  - **`void afterLast ():`**

    Move o cursor para o final deste ResultSet, logo após sua última linha;

  - **`void beforeFirst ():`**

    Move o cursor para o inicio deste ResultSet, logo antes de sua primeira linha;

  - **`void cancelRowUpdates ():`**

    Cancela as atualizações feitas na linha corrente;

  - **`void clearWarnings ():`**

    Após o uso deste método, o método getWarning retornará null até que uma nova advertência seja reportada para este ResultSet;

  - **`void close ():`**

    Libera imediatamente os recursos alocados a este ResultSet em vez de esperar que eles sejam liberados automaticamente;

  - **`void deleteRow ():`**

    Delete a linha corrente deste ResultSet de do banco de dados associado a ele;

  - **`int findColumn (String N):`**

    Torna corrente a coluna com o nome N;

- **`boolean first ():`**

  Move o cursor para a primeira linha deste ResultSet; retorna true,se a movimentação pode ser feita com sucesso, e false, caso contrário;

- **`Array getArray (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe Array (para SQL ARRAY);

- **`Array getArray (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe Array (para SQL ARRAY);

- **`InputStream getAsciiStream (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe InputStream;

- **`InputStream getAsciiStream (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe InputStream;

- **`BigDecimal getBigDecimal (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe BigDecimal;

- **`BigDecimal getBigDecimal (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe BigDecimal;

- **`Blob getBlob (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe Blob;

- **`Blob getBlob (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe Blob;

- **`boolean getBoolean (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de um boolean;

- **`boolean getBoolean (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de um boolean;

- **`byte getByte (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de um byte;

- **`byte getByte (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de um byte;

- **`byte[] getBytes (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de um vetor de bytes;

- **`byte[] getBytes (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de um vetor de bytes;

- **`Reader getCharacterStream (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe Reader;

- **`Reader getCharacterStream (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe Reader;

- **`Clob getClob (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe Clob;

- **`Clob getClob (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe Clob;

- **`int getConcurrency ():`**

  Retorna o modo de tratar concorrência deste ResultSet;

- **`String getCursorName ():`**

  Retorna o nome do cursor usado por este ResultSet;

- **`Date getDate (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe Date;

- **`Date getDate (int C, Calendar Cal):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe Date; este método emprega o Calendar Cal para construir apropriadamente a instância retornada;

- **`Date getDate (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe Date;

- **`Date getDate (String N, Calendar Cal):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe Date; este método emprega o Calendar Cal para construir apropriadamente a instância retornada;

- **`double getDouble (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de um double;

- **`double getDouble (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de um double;

- **`int getFetchDirection ():`**

  Retorna a direção padrão que este ResultSet emprega na busca por linhas;

- **`int getFetchSize ():`**

  Retorna a quantidade padrão de linhas que este ResultSet recupera por busca;

- **`float getFloat (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de um float;

- **`float getFloat (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de um float;

- **`int getInt (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de um int;

- **`int getInt (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de um int;

- **`long getLong (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de um long;

- **`long getLong (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de um long;

- **`ResultSetMetaData getMetaData ():`**

  Resulta os metadados associados a este ResultSet;

- **`Object getObject (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe Object;

- **`Object getObject (int C, Map M):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma

instância da classe Object; este método emprega o objeto M da classe Map para mapear adequadamente o tipo da coluna que está sendo recuperada;

- **`Object getObject (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe Object;

- **`Object getObject (String N, Map M):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe Object; este método emprega o objeto M da classe Map para mapear adequadamente o tipo da coluna que está sendo recuperada;

- **`Ref getRef (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de um SQL REF;

- **`Ref getRef (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de um SQL REF;

- **`int getRow ():`**

  Retorna o número da linha corrente;

- **`short getShort (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de um short;

- **`short getShort (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de um short;

- **`Statement getStatement ():`**

  Retorna a instância da classe Statement que foi usada para criar este ResultSet;

- **`String getString (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de um String;

- **`String getString (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de um String;

- **`Time getTime (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe Time;

- **`Time getTime (int C, Calendar Cal):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe Time; este método emprega o Calendar Cal para construir apropriadamente a instância retornada;

- **`Time getTime (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe Time;

- **`Time getTime (String N, Calendar Cal):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe Time; este método emprega o Calendar Cal para construir apropriadamente a instância retornada;

- **`Timestamp getTimestamp (int C):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe Timestamp;

- **`Timestamp getTimestamp (int C, Calendar Cal):`**

  Recupera o valor da coluna C da linha corrente deste ResultSet na forma de uma instância da classe Timestamp; este método emprega o Calendar Cal para construir apropriadamente a instância retornada;

- **`Timestamp getTimestamp (String N):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe Timestamp;

- **`Timestamp getTimestamp (String N, Calendar Cal):`**

  Recupera o valor da coluna de nome N da linha corrente deste ResultSet na forma de uma instância da classe Timestamp; este método emprega o Calendar Cal para construir apropriadamente a instância retornada;

- **`int getType ():`**

  Retorna o tipo deste ResultSet;

- **`SQLWarning getWarnings ():`**

  Retorna a primeira advertência reportada a este ResultSet;

- **`void insertRow ():`**

  Insere uma linha no ResultSet e no banco de dados associado a ele; deve-se estar na posição de inserção ao se empregar este método;

- **`boolean isAfterLast ():`**

  Verifica se a posição corrente deste ResultSet é a posição de seu final (após sua última linha);

- **`boolean isBeforeFirst ():`**

  Verifica se a posição corrente deste ResultSet é a posição de seu início (antes de sua primeira linha);

- **`boolean isFirst ():`**

  Verifica se a posição corrente deste ResultSet é a posição de sua primeira linha;

- **`boolean isLast ():`**

  Verifica se a posição corrente deste ResultSet é a posição de sua última linha;

- **`boolean Last ():`**

  Move o cursor para a última linha deste ResultSet; retorna true,se a movimentação pode ser feita com sucesso, e false, caso contrário;

- **`void moveToCurrentRow ():`**

  Move o cursor para a posição marcada para ser lembrada;

- **`void moveToInsertRow ():`**

  Move o cursor para a posição da linha inserida;

- **`boolean Next ():`**

  Move o cursor para a próxima linha deste ResultSet; retorna true,se a movimentação pode ser feita com sucesso, e false, caso contrário;

- **`boolean Previous ():`**

  Move o cursor para a linha anterior deste ResultSet; retorna true,se a movimentação pode ser feita com sucesso, e false, caso contrário;

- **`void refreshRow ():`**

  Atualiza a linha corrente com os valores que constam no banco de dados associado a este ResultSet;

- **`boolean relative (int Q):`**

  Adiciona Move o cursor Q linhas para frente, se Q for positivo, ou Q linhas para trás, se Q for negativo, da posição corrennte deste ResultSet; retorna true,se a movimentação pode ser feita com sucesso, e false, caso contrário;

- **`boolean rowDeleted ():`**

  Verifica se a linha corrente foi deletada;

- **`boolean rowInserted ():`**

  Verifica se a linha corrente foi inserida;

- **`boolean rowUpdated ():`**

  Verifica se a linha corrente foi atualizada;

- **`int setFetchDirection ():`**

  Ajusta a direção padrão que este ResultSet emprega na busca por linhas;

- **`int setFetchSize ():`**

  Ajusta a quantidade padrão de linhas que este ResultSet recupera por busca;

- **`void updateAsciiStream (int C, InputStream IS, int T):`**
  Atualiza o valor da coluna C da linha corrente deste ResultSet com T bytes a serem lidos do stream ASCII IS;

- **`void updateAsciiStream (String N, InputStream IS, int T):`**
  Atualiza o valor da coluna de nome N da linha corrente deste ResultSet com T bytes a serem lidos do stream ASCII IS;

- **`void updateBigDecimal (int C, BigDecimal BD):`**
  Atualiza o valor da coluna C da linha corrente com a instância da classe BigDecimal que se encontra no objeto BD;

- **`void updateBigDecimal (String N, BigDecimal BD):`**
  Atualiza o valor da coluna de nome N da linha corrente com a instância da classe BigDecimal que se encontra no objeto BD;

- **`void updateBinaryStream (int C, InputStream IS, int T):`**
  Atualiza o valor da coluna C da linha corrente deste ResultSet com T bytes a serem lidos do stream binário IS;

- **`void updateBinaryStream (String N, InputStream IS, int T):`**
  Atualiza o valor da coluna de nome N da linha corrente deste ResultSet com T bytes a serem lidos do stream binário IS;

- **`void updateBoolean (int C, boolean B):`**
  Atualiza o valor da coluna C da linha corrente com o boolean B;

- **`void updateBoolean (String N, boolean B):`**
  Atualiza o valor da coluna de nome N da linha corrente com o boolean B;

- **`void updateByte (int C, byte B):`**
  Atualiza o valor da coluna C da linha corrente com o byte B;

- **`void updateByte (String N, byte B):`**
  Atualiza o valor da coluna de nome N da linha corrente com o byte B;

- **`void updateBytes (int C, bytes B[]):`**
  Atualiza o valor da coluna C da linha corrente com o vetor de bytes B;

- **`void updateBytes (String N, byte B[]):`**
  Atualiza o valor da coluna de nome N da linha corrente com o vetor de bytes B;

- **`void updateCharacterStream (int C, Reader R, int T):`**
  Atualiza o valor da coluna C da linha corrente deste ResultSet com T bytes a serem lidos do stream de caracteres IS;

- **`void updateCharacterStream (String N, Reader R, int T):`**
  Atualiza o valor da coluna de nome N da linha corrente deste ResultSet com T bytes a serem lidos do stream de caracteres IS;

- **`void updateDate (int C, Date D):`**
  Atualiza o valor da coluna C da linha corrente com a instância da classe Date que se encontra no objeto D;

- **`void updateDate (String N, Date D):`**
  Atualiza o valor da coluna de nome N da linha corrente com a instância da classe Date que se encontra no objeto D;

- **`void updateDouble (int C, double D):`**
  Atualiza o valor da coluna C da linha corrente com o double D;

- **`void updateDouble (String N, double D):`**
  Atualiza o valor da coluna de nome N da linha corrente com o double D;

- **`void updateFloat (int C, float F):`**
  Atualiza o valor da coluna C da linha corrente com o float F;

- **`void updateFloat (String N, float F):`**
  Atualiza o valor da coluna de nome N da linha corrente com o float F;

- **`void updateInt (int C, int I):`**
  Atualiza o valor da coluna C da linha corrente com o int I;

- **`void updateInt (String N, int I):`**
  Atualiza o valor da coluna de nome N da linha corrente com o int I;

- **`void updateLong (int C, long L):`**
  Atualiza o valor da coluna C da linha corrente com o long L;

- **`void updateLong (String N, long L):`**
  Atualiza o valor da coluna de nome N da linha corrente com o long L;

- **`void updateNull (int C):`**
  Atualiza o valor da coluna C da linha corrente com NULL;

- **`void updateNull (String N):`**
  Atualiza o valor da coluna de nome N da linha corrente com NULL;

- **`void updateObject (int C, Object O):`**
  Atualiza o valor da coluna C da linha corrente com a instância da classe Object que se encontra no objeto O;

- **`void updateObject (int C, Object O, int P):`**
  Atualiza o valor da coluna C da linha corrente com a instância da classe Object que se encontra no objeto O; para os tipos java.sql.Types.DECIMAL e java.sql.Types.NUMERIC, P é a precisão, i.e., a quantidade de dígitos após o ponto decimal; para todos os demais tipos, este parâmetro será ignorado;

- **`void updateObject (String N, Object O):`**
  Atualiza o valor da coluna de nome N da linha corrente com a instância da classe Object que se encontra no objeto O;

- **`void updateObject (String N, Object O, int E):`**
  Atualiza o valor da coluna de nome N da linha corrente com a instância da classe Object que se encontra no objeto O; para os tipos java.sql.Types.DECIMAL e java.sql.Types.NUMERIC, P é a precisão, i.e., a quantidade de dígitos após o ponto decimal; para todos os demais tipos, este parâmetro será ignorado;

- **`void updateRow ():`**
  Atualiza o banco de dados com os novos valores da linha corrente deste ResultSet;

- **void updateShort (int C, short S):**
  Atualiza o valor da coluna C da linha corrente com o short S;
- **void updateShort (String N, short S):**
  Atualiza o valor da coluna de nome N da linha corrente com o short S;
- **void updateString (int C, String S):**
  Atualiza o valor da coluna C da linha corrente com o String S;
- **void updateString (String N, String S):**
  Atualiza o valor da coluna de nome N da linha corrente com o String S;
- **void updateTime (int C, Time T):**
  Atualiza o valor da coluna C da linha corrente com a instância da classe Time que se encontra no objeto T;
- **void updateTime (String N, Time T):**
  Atualiza o valor da coluna de nome N da linha corrente com a instância da classe Time que se encontra no objeto T;
- **void updateTimestamp (int C, TimeStamp TS):**
  Atualiza o valor da coluna C da linha corrente com a instância da classe TimeStamp que se encontra no objeto TS;
- **void updateTimestamp (String N, TimeStamp TS):**
  Atualiza o valor da coluna de nome N da linha corrente com a instância da classe TimeStamp que se encontra no objeto TS;
- **boolean wasNull ():**
  Verifica se a última coluna que teve seu valor recuperado tinha valor SQL NULL.

Veja abaixo um exemplo no qual se faz o acesso a um banco de dados via ODBC:

[C:\ExplsJava\Expl_05\BibAgenda\ExcecaoDeAgenda.java]

```
package BibAgenda;
```

```
public class ExcecaoDeAgenda extends RuntimeException
{
    public static final int IMPOSSIVEL_CARREGAR_DRIVER     =  0,
                            IMPOSSIVEL_ESTABELECER_CONEXAO =  1,
                            IMPOSSIVEL_CRIAR_COMANDO       =  2,
                            IMPOSSIVEL_EXECUTAR_COMANDO    =  3,
                            IMPOSSIVEL_POSICIONAR_EM_LINHA =  4,
                            IMPOSSIVEL_RECUPERAR_COLUNA    =  5,
                            PROBLEMAS_DE_ACESSO_AO_BD      =  6,
                            NOME_INEXISTENTE               =  7,
                            NOME_JA_REGISTRADO             =  8,
                            FONE_FORA_DOS_LIMITES          =  9,
                            NOME_FORA_DOS_LIMITES          = 10,
                            EXCECAO_INVALIDA               = 11;

    private static final int MENOR_ID = IMPOSSIVEL_CARREGAR_DRIVER,
                             MAIOR_ID = NOME_FORA_DOS_LIMITES;

    public static final String Mensagem [] =
    {
         "Nao foi possivel carregar o driver de acesso ao BD",
         "Nao foi possivel estabelecer conexao com o SGBD",
         "Nao foi possivel preparar comando para o SGBD",
         "Nao foi possivel para o SGBD executar comando",
         "Nao foi possivel posicionar em uma linha",
         "Nao foi possivel recuperar uma coluna",
         "Ocorreram problemas de acesso ao BD",
         "O nome deste contato nao consta na agenda",
         "O nome deste contato ja consta na agenda",
         "So solicite telefones entre zero e a quantidade existente",
         "So solicite nomes entre zero e a quantidade existente",
         "Indicacao de excecao invalida"
    };


    private int Tipo;

    private ExcecaoDeAgenda (int T)
    {
        super (ExcecaoDeAgenda.Mensagem [T]);
        this.Tipo = T;
    }

    public static void indique (int T) throws ExcecaoDeAgenda
    {
        if (T<ExcecaoDeAgenda.MENOR_ID || T>ExcecaoDeAgenda.MAIOR_ID)
            throw new ExcecaoDeAgenda (ExcecaoDeAgenda.EXCECAO_INVALIDA);

        throw new ExcecaoDeAgenda (T);

    }

    public int seuTipo ()
    {
        return this.Tipo;
    }
}
```

[C:\ExplsJava\Expl_05\BibAgenda\Agenda.java]

```
package BibAgenda;

import java.sql.*;

public class Agenda
{
    private Connection Conexao;
    private Statement  Comando;
    private ResultSet  Resultado;

    public Agenda()  throws ExcecaoDeAgenda
    {
        try
        {
            Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
        }
        catch(java.lang.ClassNotFoundException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_CARREGAR_DRIVER);
        }

        try
        {
            Conexao = DriverManager.getConnection
                      ("jdbc:odbc:ALIAS","USUARIO","SENHA");
        }
        catch(SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_ESTABELECER_CONEXAO);
        }

        try
        {
            Comando =
Conexao.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                                ResultSet.CONCUR_READ_ONLY);
        }
        catch(SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_CRIAR_COMANDO);
        }
    }
    public int quantosContatos () throws ExcecaoDeAgenda
    {
        try
        {
            Resultado = Comando.executeQuery
                      ("select count(*) as QtosContatos from Agenda");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }

        try
        {
            Resultado.first ();
        }
```

```
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_POSICIONAR_EM_LINHA);
        }

        int R = 0;

        try
        {
            R = Resultado.getInt ("QtosContatos");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_RECUPERAR_COLUNA);
        }

        return R;
    }

    public boolean haRegistroDoContato (String N) throws ExcecaoDeAgenda
    {
        try
        {
            Resultado = Comando.executeQuery
                        ("Select count(*) as Qtd from Agenda where Nome = \'"
+
                         N +
                        "\'");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }

        try
        {
            Resultado.first ();
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_POSICIONAR_EM_LINHA);
        }

        boolean R = false;

        try
        {
            R = Resultado.getInt ("Qtd") != 0;
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_RECUPERAR_COLUNA);
        }

        return R;
    }

    public void registreOContato (String N, String T) throws ExcecaoDeAgenda
```

```
    {
        if (this.haRegistroDoContato (N))
        {
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_JA_REGISTRADO);
        }

        try
        {
            Comando.executeUpdate
            ("Insert into Agenda (Nome, Telefone) values
(\'"+N+"\',\'"+T+"\')");
        }
        catch(SQLException e)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }
    }

    public String digaMeNome (int P) throws ExcecaoDeAgenda
    {
        int QC = this.quantosContatos ();

        if (P < 0 || P >= QC)
        {
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_FORA_DOS_LIMITES);
        }

        try
        {
            Resultado = Comando.executeQuery
                        ("select Nome from Agenda");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }

        try
        {
            Resultado.absolute (P+1);
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_POSICIONAR_EM_LINHA);
        }

        String R = null;

        try
        {
            R = Resultado.getString ("Nome");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_RECUPERAR_COLUNA);
        }

        return R;
    }
```

```
    public String digaMeTelefone (int P) throws ExcecaoDeAgenda
    {
        int QC = this.quantosContatos ();

        if (P < 0 || P >= this.quantosContatos ())
        {
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.FONE_FORA_DOS_LIMITES);
        }

        try
        {
            Resultado = Comando.executeQuery
                        ("select Telefone from Agenda");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }

        try
        {
            Resultado.absolute (P+1);
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_POSICIONAR_EM_LINHA);
        }

        String R = null;

        try
        {
            R = Resultado.getString ("Telefone");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_RECUPERAR_COLUNA);
        }

        return R;
    }
    public String digaMeTelefoneDe (String N) throws ExcecaoDeAgenda
    {
        if (!this.haRegistroDoContato (N))
        {

            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_INEXISTENTE);
        }

        try
        {
            Resultado = Comando.executeQuery
                        ("select Telefone from Agenda where Nome = \'" +
                         N +
                         "\'");
        }
        catch (SQLException E)
        {
```

```
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }

        try
        {
            Resultado.first ();
        }

        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_POSICIONAR_EM_LINHA);
        }

        String R = null;

        try
        {
            R = Resultado.getString ("Telefone");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_RECUPERAR_COLUNA);
        }

        return R;
    }

    public void descarteContato (String N) throws ExcecaoDeAgenda
    {
        if (!this.haRegistroDoContato (N))
        {
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_INEXISTENTE);
        }

        try
        {
            Comando.executeUpdate
            ("Delete from Agenda where Nome=\'" + N + "\'");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }
    }
}
```

[C:\ExplsJava\Expl_05\TesteDeAgenda.java]

```
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import BibAgenda.*;

class TesteDeAgenda
{
    public static void main (String Args [])
```

```
    {
        BufferedReader Entrada = new BufferedReader
                                 (new InputStreamReader
                                 (System.in));

        Agenda agenda;

        try
        {
            agenda = new Agenda ();
        }
        catch (ExcecaoDeAgenda E)
        {
            System.err.print (E.getMessage ()+"\n\n");
            return;
        }

        char   Opcao = ' ';

        do
        {
            System.out.println ();

            System.out.print ("Digite sua Opcao (" +
                              "I=Incluir/" +
                              "C=Consultar/" +
                              "L=Listar/" +
                              "E=Excluir/" +
                              "S=Sair)" +
                              ": ");

            try
            {
                Opcao = (Entrada.readLine ()).charAt (0);
            }
            catch (IOException E)
            {}

            String Nome = null, Telefone = null;

            switch (Opcao)
            {
                case 'i':
                case 'I':
                    System.out.print ("Nome....: ");
                    try
                    {
                        Nome = Entrada.readLine ();
                    }
                    catch (IOException E)
                    {}

                    try
                    {
                        System.out.print ("Telefone: ");
                        Telefone = Entrada.readLine ();

                        agenda.registreOContato (Nome, Telefone);
                    }
                    catch (IOException E)
                    {}
                    catch (ExcecaoDeAgenda E)
                    {
```

```
                        System.err.print (E.getMessage ()+"\n\n");
                    }

                    System.out.println ();
                    break;


                case 'c':
                case 'C':
                    System.out.print ("Nome....: ");
                    try
                    {
                        Nome = Entrada.readLine ();
                    }
                    catch (IOException E)
                    {}

                    try
                    {
                        Telefone = agenda.digaMeTelefoneDe (Nome);
                    }
                    catch (ExcecaoDeAgenda E)
                    {
                        System.err.print (E.getMessage ()+"\n\n");
                    }

                    System.out.println ("Telefone: " +
                                        Telefone);

                    System.out.println ();
                    break;


                case 'l':
                case 'L':
                    try
                    {
                        String Tecla;

                        for (int I = 0;
                             I < agenda.quantosContatos ();
                             I++)
                        {
                            System.out.println ("Nome: " +
                                                agenda.digaMeNome (I) +
                                                " - Telefone: " +
                                                agenda.digaMeTelefone (I));

                            if (I%23 == 22 && I < agenda.quantosContatos()-1)
                            {
                                System.out.println ();
                                System.out.print ("Tecle ENTER... ");
                                Tecla = Entrada.readLine ();
                                System.out.println ();
                            }
                        }
                    }
                    catch (IOException E)
                    {}
                    catch (ExcecaoDeAgenda E)
                    {
                        if (E.seuTipo () ==
ExcecaoDeAgenda.PROBLEMAS_DE_ACESSO_AO_BD)
```

```
                            System.err.print (E.getMessage ()+"\n\n");

                        // Nada e feito se
                        // E.seuTipo () ==
ExcecaoDeAgenda.NOME_FORA_DOS_LIMITES
                    }

                    System.out.println ();
                    break;


                case 'e':
                case 'E':
                    System.out.print ("Nome....: ");
                    try
                    {
                        Nome = Entrada.readLine ();
                    }
                    catch (IOException E)
                    {}

                    try
                    {
                        agenda.descarteContato (Nome);
                    }
                    catch (ExcecaoDeAgenda E)
                    {
                        System.err.print (E.getMessage ()+"\n\n");
                    }

                    System.out.println ();
                    break;



                case 's':
                case 'S':
                    break;


                default :
                    System.err.println ("Opcao invalida");
                    System.err.println ();
            }
        }
        while ((Opcao != 's') && (Opcao != 'S'));
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_05> javac -classpath . TesteDeAgenda.java
C:\ExplsJava\Expl_05> java -classpath . TesteDeAgenda
```

Isto proderá produzir no console a seguinte saída:

```
Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Jose
```

```
Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Jose
Esse nome ja consta na agenda!


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): e
Nome....: Joao
Nao consta na agenda!


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): c
Nome....: Jose
Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Joao
Telefone: 202.9955


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): l
Nome: Joao - Telefone: 202.9955
Nome: Jose - Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): s
```

Veja abaixo um exemplo no qual se faz o acesso direto, sem o uso de ODBC, a um banco de dados Oracle:

C:\ExplsJava\Expl_06\BibAgenda\ExcecaoDeAgenda.java]

```
package BibAgenda;

public class ExcecaoDeAgenda extends RuntimeException
{
    public static final int IMPOSSIVEL_CARREGAR_DRIVER     =  0,
                            IMPOSSIVEL_ESTABELECER_CONEXAO =  1,
                            IMPOSSIVEL_CRIAR_COMANDO       =  2,
                            IMPOSSIVEL_EXECUTAR_COMANDO    =  3,
                            IMPOSSIVEL_POSICIONAR_EM_LINHA =  4,
                            IMPOSSIVEL_RECUPERAR_COLUNA    =  5,
                            PROBLEMAS_DE_ACESSO_AO_BD      =  6,
                            NOME_INEXISTENTE               =  7,
                            NOME_JA_REGISTRADO             =  8,
```

```
                            FONE_FORA_DOS_LIMITES        =  9,
                            NOME_FORA_DOS_LIMITES        = 10,
                            EXCECAO_INVALIDA             = 11;

    private static final int MENOR_ID = IMPOSSIVEL_CARREGAR_DRIVER,
                             MAIOR_ID = NOME_FORA_DOS_LIMITES;

    public static final String Mensagem [] =
    {
         "Nao foi possivel carregar o driver de acesso ao BD",
         "Nao foi possivel estabelecer conexao com o SGBD",
         "Nao foi possivel preparar comando para o SGBD",
         "Nao foi possivel para o SGBD executar comando",
         "Nao foi possivel posicionar em uma linha",
         "Nao foi possivel recuperar uma coluna",
         "Ocorreram problemas de acesso ao BD",
         "O nome deste contato nao consta na agenda",
         "O nome deste contato ja consta na agenda",
         "So solicite telefones entre zero e a quantidade existente",
         "So solicite nomes entre zero e a quantidade existente",
         "Indicacao de excecao invalida"
    };


    private int Tipo;

    private ExcecaoDeAgenda (int T)
    {
        super (ExcecaoDeAgenda.Mensagem [T]);
        this.Tipo = T;
    }

    public static void indique (int T) throws ExcecaoDeAgenda
    {
        if (T<ExcecaoDeAgenda.MENOR_ID || T>ExcecaoDeAgenda.MAIOR_ID)
            throw new ExcecaoDeAgenda (ExcecaoDeAgenda.EXCECAO_INVALIDA);

        throw new ExcecaoDeAgenda (T);

    }

    public int seuTipo ()
    {
        return this.Tipo;
    }
}
```

[C:\ExplsJava\Expl_06\BibAgenda\Agenda.java]

```
package BibAgenda;

import java.sql.*;

public class Agenda
{
    private Connection Conexao;
    private Statement  Comando;
    private ResultSet  Resultado;

    public Agenda()  throws ExcecaoDeAgenda
```

```
    {
        try
        {
            Class.forName ("oracle.jdbc.driver.OracleDriver");
        }
        catch(java.lang.ClassNotFoundException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_CARREGAR_DRIVER);
        }

        try
        {
            // Se o serviror for a propria maquina, IPSERVIDOR=127.0.0.1
            // Normalmente, PORTA=1521
            // A menos que tenha sido criada outra, INSTANCIA=orcl

            Conexao = DriverManager.getConnection
                      ("jdbc:oracle:thin:@IPSERVIDOR:PORTA:INSTANCIA",
"USUARIO", "SENHA");
        }
        catch(SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_ESTABELECER_CONEXAO);
        }

        try
        {
            Comando =
Conexao.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                                ResultSet.CONCUR_READ_ONLY);
        }
        catch(SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_CRIAR_COMANDO);
        }
    }

    public int quantosContatos () throws ExcecaoDeAgenda
    {
        try
        {
            Resultado = Comando.executeQuery
                        ("select count(*) as QtosContatos from Agenda");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }

        try
        {
            Resultado.first ();
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_POSICIONAR_EM_LINHA);
        }
```

```
        int R = 0;

        try
        {
            R = Resultado.getInt ("QtosContatos");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_RECUPERAR_COLUNA);
        }

        return R;
    }

    public boolean haRegistroDoContato (String N) throws ExcecaoDeAgenda
    {
        try
        {
            Resultado = Comando.executeQuery
                        ("Select count(*) as Qtd from Agenda where Nome = \'"
+
                          N +
                         "\'");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }

        try
        {
            Resultado.first ();
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_POSICIONAR_EM_LINHA);
        }

        boolean R = false;

        try
        {
            R = Resultado.getInt ("Qtd") != 0;
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_RECUPERAR_COLUNA);
        }

        return R;
    }

    public void registreOContato (String N, String T) throws ExcecaoDeAgenda
    {
        if (this.haRegistroDoContato (N))
        {
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_JA_REGISTRADO);
        }
```

```
        try
        {
            Comando.executeUpdate
            ("Insert into Agenda (Nome, Telefone) values
(\'"+N+"\',\'"+T+"\')");
        }
        catch(SQLException e)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }
    }

    public String digaMeNome (int P) throws ExcecaoDeAgenda
    {
        int QC = this.quantosContatos ();

        if (P < 0 || P >= QC)
        {
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_FORA_DOS_LIMITES);
        }

        try
        {
            Resultado = Comando.executeQuery
                        ("select Nome from Agenda");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }

        try
        {
            Resultado.absolute (P+1);
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_POSICIONAR_EM_LINHA);
        }

        String R = null;

        try
        {
            R = Resultado.getString ("Nome");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_RECUPERAR_COLUNA);
        }

        return R;
    }

    public String digaMeTelefone (int P) throws ExcecaoDeAgenda
    {
        int QC = this.quantosContatos ();

        if (P < 0 || P >= this.quantosContatos ())
```

```
        {
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.FONE_FORA_DOS_LIMITES);
        }

        try
        {
            Resultado = Comando.executeQuery
                        ("select Telefone from Agenda");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }

        try
        {
            Resultado.absolute (P+1);
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_POSICIONAR_EM_LINHA);
        }

        String R = null;

        try
        {
            R = Resultado.getString ("Telefone");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_RECUPERAR_COLUNA);
        }

        return R;
    }

    public String digaMeTelefoneDe (String N) throws ExcecaoDeAgenda
    {
        if (!this.haRegistroDoContato (N))
        {
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_INEXISTENTE);
        }

        try
        {
            Resultado = Comando.executeQuery
                        ("select Telefone from Agenda where Nome = \'" +
                         N +
                         "\'");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }

        try
        {
            Resultado.first ();
```

```
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_POSICIONAR_EM_LINHA);
        }

        String R = null;

        try
        {
            R = Resultado.getString ("Telefone");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_RECUPERAR_COLUNA);
        }

        return R;
    }

    public void descarteContato (String N) throws ExcecaoDeAgenda
    {
        if (!this.haRegistroDoContato (N))
        {
            ExcecaoDeAgenda.indique (ExcecaoDeAgenda.NOME_INEXISTENTE);
        }

        try
        {
            Comando.executeUpdate
            ("Delete from Agenda where Nome=\'" + N + "\'");
        }
        catch (SQLException E)
        {
            ExcecaoDeAgenda.indique
(ExcecaoDeAgenda.IMPOSSIVEL_EXECUTAR_COMANDO);
        }
    }
}
```

[C:\ExplsJava\Expl_06\TesteDeAgenda.java]

```
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import BibAgenda.*;

class TesteDeAgenda
{
    public static void main (String Args [])
    {
        BufferedReader Entrada = new BufferedReader
                                 (new InputStreamReader
                                 (System.in));

        Agenda agenda;

        try
```

```
        {
            agenda = new Agenda ();
        }
        catch (ExcecaoDeAgenda E)
        {
            System.err.print (E.getMessage ()+"\n\n");
            return;
        }

        char   Opcao = ' ';

        do
        {
            System.out.println ();

            System.out.print ("Digite sua Opcao (" +
                              "I=Incluir/" +
                              "C=Consultar/" +
                              "L=Listar/" +
                              "E=Excluir/" +
                              "S=Sair)" +
                              ": ");

            try
            {
                Opcao = (Entrada.readLine ()).charAt (0);
            }
            catch (IOException E)
            {}

            String Nome = null, Telefone = null;

            switch (Opcao)
            {
                case 'i':
                case 'I':
                    System.out.print ("Nome....: ");
                    try
                    {
                        Nome = Entrada.readLine ();
                    }
                    catch (IOException E)
                    {}

                    try
                    {
                        System.out.print ("Telefone: ");
                        Telefone = Entrada.readLine ();

                        agenda.registreOContato (Nome, Telefone);
                    }
                    catch (IOException E)
                    {}
                    catch (ExcecaoDeAgenda E)
                    {
                        System.err.print (E.getMessage ()+"\n\n");
                    }

                    System.out.println ();
                    break;


                case 'c':
```

```
                case 'C':
                    System.out.print ("Nome....: ");
                    try
                    {
                        Nome = Entrada.readLine ();
                    }
                    catch (IOException E)
                    {}

                    try
                    {
                        Telefone = agenda.digaMeTelefoneDe (Nome);
                    }
                    catch (ExcecaoDeAgenda E)
                    {
                        System.err.print (E.getMessage ()+"\n\n");
                    }

                    System.out.println ("Telefone: " +
                                        Telefone);

                    System.out.println ();
                    break;


                case 'l':
                case 'L':
                    try
                    {
                        String Tecla;

                        for (int I = 0;
                             I < agenda.quantosContatos ();
                             I++)
                        {
                            System.out.println ("Nome: " +
                                                agenda.digaMeNome (I) +
                                                " - Telefone: " +
                                                agenda.digaMeTelefone (I));

                            if (I%23 == 22 && I < agenda.quantosContatos()-1)
                            {
                                System.out.println ();
                                System.out.print ("Tecle ENTER... ");
                                Tecla = Entrada.readLine ();
                                System.out.println ();
                            }
                        }
                    }
                    catch (IOException E)
                    {}
                    catch (ExcecaoDeAgenda E)
                    {
                        if (E.seuTipo () ==
ExcecaoDeAgenda.PROBLEMAS_DE_ACESSO_AO_BD)
                            System.err.print (E.getMessage ()+"\n\n");

                        // Nada e feito se
                        // E.seuTipo () ==
ExcecaoDeAgenda.NOME_FORA_DOS_LIMITES
                    }

                    System.out.println ();
```

```
                    break;


                case 'e':
                case 'E':
                    System.out.print ("Nome....: ");
                    try
                    {
                        Nome = Entrada.readLine ();
                    }
                    catch (IOException E)
                    {}

                    try
                    {
                        agenda.descarteContato (Nome);
                    }
                    catch (ExcecaoDeAgenda E)
                    {
                        System.err.print (E.getMessage ()+"\n\n");
                    }

                    System.out.println ();
                    break;



                case 's':
                case 'S':
                    break;


                default :
                    System.err.println ("Opcao invalida");
                    System.err.println ();
            }
        }
        while ((Opcao != 's') && (Opcao != 'S'));
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_06> javac -classpath . TesteDeAgenda.java
C:\ExplsJava\Expl_06> java -classpath .;classes12.zip TesteDeAgenda
```

Deve-se ressaltar que classes12.zip é um arquivo que acompanha o Oracle e que contem drivers que permitem o acesso a um banco de dados Oracle.

A execução do programa poderia produzir no console a seguinte saída:

```
Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Jose
Telefone: 211.4466
```

```
Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Jose
Esse nome ja consta na agenda!


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): e
Nome....: Joao
Nao consta na agenda!


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): c
Nome....: Jose
Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): i
Nome....: Joao
Telefone: 202.9955


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): l
Nome: Joao - Telefone: 202.9955
Nome: Jose - Telefone: 211.4466


Digite sua Opcao (I=Incluir/C=Consultar/L=Listar/E=Excluir/S=Sair): s
```

Aplicações com Interface Gráfica (O Pacote java.awt)

O pacote java.awt (abstract window toolkit) é uma biblioteca de componentes gráficos bem projetada e muito portável. Ele é parte integrante do ambiente Java e provê todas funcionalidades básicas que seriam esperadas de um sistema moderno de janelas.

Este pacote consiste dos componentes gráficos que são padrão nas GUI (graphical user interface) e é muito fácil de ser entendido e usado.

Quando um componente é criado, ele é normalmente adicionado a um *container*. Um *container* pode ser entendido simplesmente como uma área da tela na qual componentes (e mesmo outros *containers*) podem ser colocados.

Todo componente é uma instância de uma classe derivada da classe Component. Isto possibilita a existência de um conjunto central de métodos que agem sobre todos tipos de componentes, tais como métodos que ajustam a cor, por exemplo.

Componentes nunca são dispostos em posições absolutas dentro de um *container* e sim em posições relativas a outros objetos. Isto porque Java foi projetada para operar em diferentes plataformas e, por isso, precisa cuidar de manter a aparência da GUI coerente e consistente dentro do ambiente do sistema operacional nativo.

Desta forma, o tamanho e a forma exata de um componente não são conhecidas pelo projetista da interface. Embora isto pareça estranho a princípio, trata-se de uma poderosa técnica de programação que visa tornar suas aplicações mais robustas.

Componentes

Componentes são como blocos que nos permitem construir uma GUI. Os componentes do pacote java.awt podem, conceitualmente, serem agrupados em quatro categorias principais, a saber:

- **Containers:**

  Esta categoria de componentes engloba componentes que representam áreas nas quais componentes de interface podem ser posicionados. Como exemplos de componentes desta categoria, podemos citar paineis, janelas, applets, etc;

- **Gerenciadores de Layout:**

  Esta categoria de componentes engloba componentes que têm como missão controlar a disposição dos objetos contidos por um container. Como exemplos de componentes desta categoria, podemos citar FlowLayout, BorderLayout, GridLayout, etc;

- **Controles:**

  Esta categoria de componentes engloba componentes normalmente associados com sistemas de janelas. Como exemplos de componentes desta categoria, podemos citar botões, rótulos, caixas de texto, etc;

- **Detectores de Evento:**

Esta categoria de componentes engloba componentes capazes de detectar os diversos tipos de eventos que podem ser encaminhados a um controle ou container. Como exemplos de componentes desta categoria, podemos citar MouseListener, FocusListener, KeyListener, etc.

Controles do Pacote java.awt

Controles são componentes de interface especificamente projetados para obter e fornecer informações ao usuário.

java.awt.Component

A classe abstrata java.awt.Component é base para diversas classe de objetos que têm uma representação gráfica que pode ser mostrada na tela e que pode interagir com o usuário.

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com instâncias de classes dela derivadas:

- **Métodos:**
  - **void add (PopupMenu M):**
    Adiciona o PopupMenu M ao Component;
  - **void addComponentListener (ComponentListener L):**
    Registra L como receptor de ComponentEvent's do Component;
  - **void addFocusListener (FocusListener L):**
    Registra L como receptor de FocusEvent's do Component;
  - **void addKeyListener (KeyListener L):**
    Registra L como receptor de KeyEvent's do Component;
  - **void addMouseListener (MouseListener L):**
    Registra L como receptor de MouseEvent's relacionados a ações do mouse sobre o Component;

- **void addMouseMotionListener (MouseMotionListener L):**
  Registra L como receptor de MouseEvent's relacionados a movimentos dou mouse sobre o Component;
- **void dispatchEvent (AWTEvent e):**
  Despacha o evento e;
- **float getAlignmentX ():**
  Retorna o alinhamento ao longo do eixo X deste Component;
- **float getAlignmentY ():**
  Retorna o alinhamento ao longo do eixo Y deste Component;
- **Color getBackground ():**
  Retorna a cor de fundo deste Component;
- **Rectangle getBounds ():**
  Retorna os limites deste Component;
- **Cursor getCursor ():**
  Retorna o cursor usado pelo mouse sobre este Component;
- **Font getFont ():**
  Retorna o fonte utilizado para escrever neste Component;
- **FontMetrics getFontMetrics (Font F):**
  Retorna as métricas empregadas no fonte especificado;
- **Color getForeground ():**
  Retorna a cor usada para escrever neste Component;
- **Graphics getGraphics ():**
  Retorna o contexto gráfico deste Component;
- **int getHeight ():**
  Retorna a altura deste Compoent;

- **Dimension getMaximumSize ():**

  Retorna o tamanho máximo deste Component;

- **Dimension getMinimumSize ():**

  Retorna o tamanho mínimo deste Component;

- **String getName ():**

  Retorna o nome deste Component;

- **Container getParent ():**

  Retorna o Container que comtém este Component;

- **Dimension getPreferredSize ():**

  Retorna o melhor tamanho deste Component;

- **Dimension getSize ():**

  Retorna o tamanho deste Component;

- **int getWidth ():**

  Retorna a largura deste Component;

- **int getX ():**

  Retorna a coordenada X da origem deste Component;

- **int getY ():**

  Retorna a coordenada Y da origem deste Component;

- **boolean hasFocus ():**

  Verifica se este Component tem o foco;

- **void invalidate ():**

  Invalida este Component;

- **boolean isDisplayable ():**

  Verifica se este Component é passível de ser mostrado na tela;

- **boolean isEnabled ():**

  Verifica se este Component está habilitado;

- **boolean isFocusTraversable ():**

  Verifica se este Component pode ser atingido e receber o foco através de sucessivos pressionamentos da tecla TAB ou SHIFT-TAB;

- **boolean isOpaque ():**

  Verifica se este Component é completamente opaco (o padrão é que retorne false);

- **boolean isShowing ():**

  Verifica se este Component está aparente na tela;

- **boolean isValid ():**

  Verifica se este Component é válido;

- **boolean isVisible ():**

  Verifica se este Component está visível;

- **void paint (Graphics G):**

  Pinta este Component;

- **void paintAll (Graphics G):**

  Pinta este Component e todos os seus subcomponentes;

- **void remove (PopupMenu M):**

  Remove o PopupMenu M do Component;

- **void removeComponentListener (ComponentListener L):**

  Cancela o registro de L como receptor de ComponentEvent's do Component;

- **void removeFocusListener (FocusListener L):**

  Cancela o registro de L como receptor de FocusEvent's do Component;

- **void removeKeyListener (KeyListener L):**

  Cancela o registro de L como receptor de KeyEvent's do Component;

- **void removeMouseListener (MouseListener L):**

  Cancela o registro de L como receptor de MouseEvent's relacionados a ações do mouse sobre o Component;

- **void removeMouseMotionListener (MouseMotionListener L):**
  Cancela o registro de L como receptor de MouseEvent's relacionados a movimentos dou mouse sobre o Component;

- **void repaint ():**
  Repinta este Component;

- **void requestFocus ():**
  Requisita o foco para este Component;

- **void setBackground (Color C):**
  Ajusta a cor de fundo deste Component;

- **void setBounds (int x, int y, int l, int a):**
  Move este componente para o ponto (x,y) e ajusta o tamanho seu tamanho para l pontos de largura e a pontos de altura;

- **void setBounds (Rectangle R):**
  Move e ajusta os limites deste Component para estar de conformidade com o retângulo R que o limita;

- **void setCursor (Cursor C):**
  Ajusta o cursor usado pelo mouse sobre este Component;

- **void setEnabled (boolean B):**
  Habilita ou desabilita este Component;

- **void setFont (Font F):**
  Ajusta o fonte utilizado para escrever neste Component;

- **void setForeground (Color C):**
  Ajusta a cor usada para escrever neste Component;

- **void setName (String N):**
  Ajusta o nome deste Component;

- **void setSize (Dimension D):**
  Ajusta o tamanho deste Component;

- **void setSize (int l, int a):**

  Ajusta o tamanho deste Component l pontos de largura e a pontos de altura;

- **void setVisible (boolean B):**

  Ajusta a visibilidade deste Component;

- **void transferFocus ():**

  Transfere o foco deste Component para o próximo;

- **void update (Graphics G):**

  Atualiza este Component;

- **void validate ():**

  Assegura que este Component tenha um layout válido.

java.awt.Button

A classe java.awt.Button implementa um botão. Além das estruturas e do comportamento herdado da classe java.awt.Component, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

  - **Button ():**

    Constroi uma instância da classe Button sem nenhum rótulo;

  - **Button (String R):**

    Constroi uma instância da classe Button com o rótulo R;

- **Métodos:**

  - **String getLabel ():**

    Retorna o String que lhe serve de rótulo;

  - **void setLabel (String R):**

    Faz do String R o rótulo do botão.

java.awt.Canvas

Um canvas é um componente completamente genérico. Serve de base para diversos componentes gráficos interessantes. Canvases não são muito úteis para programas de complexidade média ou baixa, mas são extremamente úteis em programas nos quais é preciso criar do zero componentes gráficos novos.

Além das estruturas e do comportamento herdado da classe java.awt.Component, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **Canvas ():**
    Constroi uma instância da classe Canvas;
- **Métodos:**
  - **void paint (Graphics G):**
    Pinta o canvas com a cor padrão de fundo.

java.awt.Checkbox

Uma Checkbox é uma pequena caixa (com um rótulo opcional) que pode ser marcada ou não. Checkboxes podem ser agrupadas em um CheckboxGroup de forma a somente permitir que uma Checkbox de um grupo esteja marcada em um certo instante.

Além das estruturas e do comportamento herdado da classe java.awt.Component, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **Checkbox ():**
    Constroi uma instância da classe Checkbox sem nenhum rótulo;
  - **Checkbox (String R):**
    Constroi uma instância da classe Checkbox com o rótulo R;

- **Checkbox (String R, boolean E, CheckboxGroup G):**
  Constroi uma instância da classe Checkbox com o rótulo R, contido no grupo G; caso E seja true, a Checkbox será criada marcada, caso contrário será criada desmarcada;
- **Checkbox (String R, CheckboxGroup G, boolean E):**
  Constroi uma instância da classe Checkbox com o rótulo R, contido no grupo G; caso E seja true, a Checkbox será criada marcada, caso contrário será criada desmarcada;

- **Métodos:**
  - **String getLabel ():**
    Retorna o String que lhe serve de rótulo;
  - **void setLabel (String R):**
    Faz do String R o rótulo da Checkbox;
  - **boolean getState ():**
    Retorna true, se a Checkbox estiver marcada, e false, caso contrário;
  - **void setState (boolean E):**
    Caso E seja true, a Checkbox será marcada, caso contrário será desmarcada;
  - **CheckboxGroup getCheckboxGroup ():**
    Retorna o grupo ao qual pertence a Checkbox;
  - **void setCheckboxGroup (CheckboxGroup G):**
    Faz com que a Checkbox passe a integrar o grupo G.

java.awt.Choice

Uma Choice apresenta um menu pop-up de opções. A opção corrente é mostrada como título da Choice.

Além das estruturas e do comportamento herdado da classe java.awt.Component, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

- **Choice ():**

  Constroi uma instância da classe Choice;

- **Métodos:**

  - **void add (String I):**

    Adiciona o ítem I à Choice;

  - **void addItem (String I):**

    O mesmo que add, ou seja, adiciona o ítem I à Choice;

  - **String getItem (int i):**

    Retorna o ítem de número i da Choice;

  - **int getItemCount ():**

    Retorna a quantidade de ítens em uma Choice;

  - **int getSelectedIndex ():**

    Retorna o número do ítem selecionado;

  - **String getSelectedItem ():**

    Retorna uma representação do ítem selecionado;

  - **void insert (String I, int i):**

    Insere o ítem I como i$^{\underline{\text{ézimo}}}$ ítem da Choice;

  - **void remove (int i):**

    Remove o ítem de número i da Choice;

  - **void remove (String I):**

    Remove a primeira ocorrência do ítem I da Choice;

  - **void removeAll ():**

    Remove todos os ítens da Choice;

  - **void select (int i):**

    Seleciona o ítem de número i da Choice;

- **void select (String I):**
  Seleciona a primeira ocorrência do ítem I da Choice.

java.awt.Label

Um Label é somente um texto que pode ser disposto em um container. Apesar de não fazerem muito, são extremamente úteis para deixar claro a funcionalidade de uma aplicação.

Além das estruturas e do comportamento herdado da classe java.awt.Component, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **Label ():**
    Constroi uma instância da classe Label que não contém nenhum texto;
  - **Label (String T):**
    Constroi uma instância da classe Label contendo o texto T;
  - **Label (String T, int A):**
    Constroi uma instância da classe Label contendo o texto T alinhado conforme especificado por seu parâmetro A, que pode valer: (1) Label.LEFT para alinhamento a esquerda; (2) Label.CENTER para centralizar; ou (3) Label.Right para alinhamento a direita.
- **Métodos:**
  - **int getAlignment ():**
    Retorna o alinhamento de um Label; os valores válidos para retorno deste método são os valores mencionados como válidos para o 2º parâmetro do 3º construtor desta classe;
  - **int setAlignment (int A):**
    Faz com que o Label tenha o alinhamento A; os valores válidos para parâmetro deste método são os valores mencionados como válidos para o 2º parâmetro do 3º construtor desta classe;

- **String getText ():**
  Retorna o String que lhe serve de texto;
- **void setText (String T):**
  Faz com que o String T se torne o texto do Label.

java.awt.List

Uma lista nada mais é que uma relação de ítens dentre os quais o usuário pode escolher um ou mais. Além das estruturas e do comportamento herdado da classe java.awt.Component, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **List ():**
    Constroi uma instância da classe List sem nenhuma linha visível e que não possíbilita múltiplas seleções;
  - **List (int Q, boolean SM):**
    Constroi uma instância da classe List com Q linhas visíveis; se SM valer true, será possível seleções múltiplas, caso contrário, não serão permitidas seleções múltiplas;
- **Métodos:**
  - **void addItem (String I):**
    Adiciona o ítem I no final da lista
  - **void addItem (String I, int P):**
    Adiciona o ítem I na posição P da lista (todos os ítens subseqüentes são deslocados para baixo);
  - **void clear ():**
    Limpa a lista removendo todos os seus ítens;
  - **int countItems:**
    Retorna a quantidade de ítens presentes na lista;

- **void delItem (int P):**
  Remove o ítem que se encontra na posição P da lista;
- **String getItem (int P):**
  Retorna o ítem que se encontra na posição P da lista;
- **void replaceItem (String I, int P):**
  Troca o ítem que se encontra na posição P por I.

java.awt.Scrollbar

Uma Scrollbar nada mais é que uma barra deslizante que pode ser empregada em diversas circunstâncias. Podem ser horizontais ou verticais.

Além das estruturas e do comportamento herdado da classe java.awt.Component, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **Scrollbar (int O):**
    Constroi uma instância da classe Scrollbar orientada horizontal ou verticalmente conforme especificado por seu parâmetro O, que pode valer:
    (1) Scrollbar.HORIZONTAL; ou (2) Scrollbar.VERTICAL.
  - **Scrollbar (int O, int DV, int PV, int vm, int VM):**
    Constroi uma instância da classe Scrollbar orientada horizontal ou verticalmente conforme especificado por seu parâmetro O (os valores válidos para este parâmetro são os mesmos especificados como valores válidos para o parâmetro do primeiro construtor da classe Scrollbar), tendo DV como seu valor *default*, PV como sua porção visível, vm como seu valor mínimo e VM como seu valor máximo.
- **Métodos:**
  - **int getOrientation ():**
    Retorna a orientação da Scrollbar (os valores válidos para retorno deste método são os mesmos especificados como valores válidos para o parâmetro do primeiro construtor da classe Scrollbar);

- **void setValue (int V):**
  Torna V o valor da Scrollbar;
- **int getMinimum ():**
  Retorna o valor mínimo da Scrollbar;
- **int getMaximum ():**
  Retorna o valor máximo da Scrollbar;
- **int getVisible ():**
  Retorna a porção visível da Scrollbar.

java.awt.TextField

Um TextField é um componente que permite que o usuário digite uma única linha de texto. Embora seu nome sugira que serve somente para a entrada de texto, lembre-se de que números também são texto.

Além das estruturas e do comportamento herdado da classe java.awt.Component, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **TextField ():**
    Constroi uma instância da classe TextField;
  - **TextField (int C):**
    Constroi uma instância da classe Textfield com C colunas;
  - **TextField (String T):**
    Constroi uma instância da classe TextField contendo o texto T;
  - **TextField (String T, int C):**
    Constroi uma instância da classe TextField com C colunas e contendo o texto T;
- **Métodos:**
  - **int getColumns ():**
    Retorna a quantidade de colunas do TextField;

- **String getText ():**

  Retorna o texto contido no TextField;

- **void setText (String T):**

  Faz com que T se torne o texto contido no TextField.

java.awt.TextArea

Uma TextArea é um componente editor de texto que se parece muito com um TextField, exceto pelo fato de que permite a entrada de múltiplas linhas.

Além das estruturas e do comportamento herdado da classe java.awt.Component, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

  - **TextArea (int L, int C):**

    Constroi uma instância da classe TextArea com L linhas e C colunas;

  - **TextArea (String T):**

    Constroi uma instância da classe TextArea contendo o texto T;

  - **TextArea (String T, int L, int C):**

    Constroi uma instância da classe TextArea com L linhas e C colunas contendo o texto T;

- **Métodos:**

  - **int getColumns ():**

    Retorna a quantidade de colunas do TextArea;

  - **int getRows ():**

    Retorna a quantidade de linhas do TextArea;

  - **String getText ():**

    Retorna o texto contido no TextArea;

  - **void setText (String T):**

    Faz com que T se torne o texto contido no TextArea.

java.awt.MenuComponent

A classe abstrata java.awt.MenuComponent é base para todas as classes de componentes relacionadas a menus. Neste sentido, a classe java.awt.MenuComponent é análoga à classe java.awt.Component para os componentes AWT.

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com instâncias de classes dela derivadas:

- **Métodos:**
  - **void dispatchEvent (AWTEvent e):**
    Despacha o evento e;
  - **Font getFont ():**
    Retorna o fonte utilizado para escrever neste MenuComponent;
  - **String getName ():**
    Retorna o nome deste MenuComponent;
  - **Container getParent ():**
    Retorna o Container que comtém este MenuComponent;
  - **void setFont (Font F):**
    Ajusta o fonte utilizado para escrever neste MenuComponent;
  - **void setName (String N):**
    Ajusta o nome deste MenuComponent.

java.awt.MenuItem

Todos os ítens em um menu devem ser instâncias da classe MenuItem ou de uma de suas subclasses. O padrão é que um MenuItem seja constituido tão somente por um ítem rotulado em um menu.

Além das estruturas e do comportamento herdado da classe java.awt.MenuComponent, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **MenuItem ():**
    Constroi uma instância da classe MenuItem sem rótulo e sem tecla de atalho;
  - **MenuItem (String R):**
    Constroi uma instância da classe MenuItem com rótulo R e sem tecla de atalho;
  - **MenuItem (String R, MenuShortcut S):**
    Constroi uma instância da classe MenuItem com rótulo R e com tecla de atalho representada pelo MenuShortcut S;
- **Métodos:**
  - **void addActionListener (ActionListener L):**
    Registra L como receptor de ActionEvent's do MenuItem;
  - **void deleteMenuShortcut (MenuShortcut S):**
    Faz S deixar de ser o MenuShortcut desta MenuBar;
  - **String getActionCommand ():**
    Retorna o nome do comando do ActionEvent disparado por este MenuItem;
  - **String getLabel ():**
    Retorna o rótulo deste MenuItem;
  - **boolean isEnabled ():**
    Verifica se este MenuItem está habilitado;
  - **void removeActionListener (ActionListener L):**
    Cancela o registro de L como receptor de ActionEvent's direcionados ao MenuItem;
  - **void setActionCommand (String C):**
    Faz de C o comando associado a este MenuItem;
  - **void setEnabled (boolean B):**
    Habilita ou desabilita este MenuItem;

- **void setLabel (String R):**
  Rotula este MenuItem com o rótulo R;
- **void setShortcut (MenuShortcut S):**
  Torna a tecla de atalho representada por S a tecla de atalho deste MenuItem.

java.awt.MenuShortcut

Classe quer representa uma tecla de atalho para um MenuItem.

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **MenuShortcut (int T):**
    Constroi uma instância da classe MenuShortcut que representa a tecla de atalho com código T;
  - **MenuShortcut (int T, boolean S):**
    Constroi uma instância da classe MenuShortcut que representa a tecla de atalho com código T (S indica o uso do SHIFT como modificador);
- **Métodos:**
  - **int getKey ():**
    Retorna o código da tecla de atalho representada por este MenuShortcut;
  - **boolean usesShiftModifier ():**
    Verifica se este MenuShortcut usa o modificador SHIFT).

java.awt.CheckboxMenuItem

Esta classe representa Checkbox'es que podem ser incluidas como MenuItem's em um Menu. Além das estruturas e do comportamento herdado da classe java.awt.MenuItem, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

- **CheckboxMenuItem ():**
  Constroi uma instância da classe CheckboxMenuItem sem rótulo;
- **CheckboxMenuItem (String R):**
  Constroi uma instância da classe CheckboxMenuItem com rótulo R;
- **CheckboxMenuItem (String R, boolean S):**
  Constroi uma instância da classe CheckboxMenuItem com rótulo R; caso S seja true, o CheckboxMenuItem será criado marcado, e caso seja false, será criado desmarcado;

- **Métodos:**
  - **void addItemListener (ItemListener L):**
    Registra L como receptor de ItemEvent's do CheckboxMenuItem;
  - **boolean getState ():**
    Retorna o estado do CheckboxMenuItem (true, se marcado, false, caso contrário);
  - **void removeItemListener (ItemListener L):**
    Cancela o registro de L como receptor de ItemEvent's direcionados ao CheckboxMenuItem;
  - **void setState (boolean S):**
    Marca ou desmarca este CheckboxMenuItem;

java.awt.Menu

Esta classe representa pull-down menus e são ligados uma MenuBar ou a um outro Menu (de quem seria submenu). Pode, opcionalmente, ser destacável, i.e., arrastado para longe de sua MenuBar ou do Menu de quem é submenu.

Além das estruturas e do comportamento herdado da classe java.awt.MenuItem, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **Menu ():**
    Constroi uma instância da classe Menu sem rótulo;

- **Menu (String R):**

  Constroi uma instância da classe Menu com rótulo R;

- **Menu (String R, boolean D):**

  Constroi uma instância da classe MenuItem com rótulo R; este Menu será destacável, caso D valha true, e será fixo, caso contrário;

- **Métodos:**

  - **MenuItem add (MenuItem I):**

    Adiciona o MenuItem I a este Menu;

  - **void add (String R):**

    Adiciona um ítem rotulado com o rótulo R a este Menu;

  - **void addSeparator ():**

    Adiciona uma linha de separação neste Menu;

  - **MenuItem getItem (int i):**

    Retorna o i$^{\underline{\text{ézimo}}}$ ítem de menu deste Menu;

  - **int getItemCount ():**

    Retorna a quantidade de ítens neste Menu;

  - **void insert (MenuItem I, int i):**

    Insere o MenuItem I na posição i deste Menu;

  - **void insert (String R, int i):**

    Insere um novo MenuItem rotulado com rótulo R na posição i deste Menu;

  - **void insertSeparator (int i):**

    Insere uma linha de separação na posição i deste Menu;

  - **boolean isTearOff ():**

    Verifica se este menu é destacável;

  - **void remove (int i):**

    Remove i$^{\underline{\text{ézimo}}}$ ítem deste Menu;

- **void remove (MenuComponent C):**
  Remove o MenuComponent C deste Menu;
- **void removeAll ():**
  Remove todos os ítens deste Menu.

java.awt.MenuBar

Uma MenuBar é um componente que representa um menu de opções associado a um Frame. Para associar uma MenuBar a um Frame, basta empregar o método setMenuBar do Frame.

Além das estruturas e do comportamento herdado da classe java.awt.MenuComponent, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **MenuBar ():**
    Constroi uma instância da classe MenuBar;
- **Métodos:**
  - **Menu add (Menu M):**
    Adiciona M a esta MenuBar;
  - **void deleteMenuShortcut (MenuShortcut S):**
    Faz S deixar de ser o MenuShortcut desta MenuBar;
  - **Menu getHelpMenu ():**
    Retorna o Menu de ajuda desta MenuBar;
  - **Menu getMenu (int i):**
    Retorna o Menu cujo número de ordem foi especificado;
  - **int getMenuCount ():**
    Retorna a quantidade de Menu's nesta MenuBar;

- **MenuItem getShortcutMenuItem (MenuShortcut S):**
  Retorna o MenuItem associado ao MenuShortcut S; retorna null caso não haja nenhum;

- **void remove (int i):**
  Remove desta MenuBar o Menu cujo número de ordem foi especificado;

- **void remove (MenuComponent C):**
  Remove desta MenuBar o MenuComponent C;

- **void setHelpMenu (Menu M):**
  Torna M o Menu de ajuda desta MenuBar;

- **Enumeration Shortcuts ():**
  Retorna uma enumeração de todos os MenuShortcut's que esta MenuBar está gerenciando.

java.awt.PopupMenu

PopupMenu's são menus que podem aparecer dinamicamente em uma posição específica de um componente. Além das estruturas e do comportamento herdado da classe java.awt.Menu, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

  - **PopupMenu ():**
    Constroi uma instância da classe PopupMenu;

  - **PopupMenu (String N):**
    Constroi uma instância da classe PopupMenu e dá a ela o nome de N;

- **Métodos:**

  - **void show (Component C, int x, int y):**
    Exibe este PopupMenu na posição (x,y) do Component C.

Containers do Pacote java.awt

Container's são componentes que podem conter outros componentes. Janelas são Container's típicos, embora existam também outros Container's.

java.awt.Container

A classe java.awt.Container é base para todas as classes de componentes capazes de conter outros componentes. Além das estruturas e do comportamento herdado da classe java.awt.Component, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **Container ():**
    Constroi uma instância da classe Container;
- **Métodos:**
  - **Component add (Component C):**
    Adiciona o componente C a este Container;
  - **Component add (Component C, int i):**
    Adiciona o componente C como o i$^{\text{ézimo}}$ Component deste Container;
  - **void addContainerListener (ContainerListener L):**
    Registra L como receptor de ContainerEvent's deste Container;
  - **void doLayout ():**
    Faz com que o Container disponha seus Component's;
  - **Component findComponentAt (int x, int y):**
    Retorna o Component visível que se encontra na posição (x,y) deste Container;
  - **Component findComponentAt (Point P):**
    Retorna o Component visível que se encontra no ponto P deste Container;
  - **float getAlignmentX ():**
    Retorna o alinhamento deste Container ao longo do eixo X;

- **float getAlignmentY ():**

  Retorna o alinhamento deste Container ao longo do eixo Y;

- **Component getComponent (int i):**

  Retorna o i$^{\text{ézimo}}$ Component deste Container;

- **Component getComponentAt (int x, int y):**

  O mesmo que findComponent (int x, int y);

- **Component getComponentAt (Point P):**

  O mesmo que findComponent (Point P);

- **int getComponentCount ():**

  Retorna a quantidade de Component's deste Container;

- **Component[] getComponents ():**

  Retorna um vetor com todos os Component's deste Container;

- **Insets getInsets ():**

  Retorna os Insets deste Container (que determinam a o tamanho da borda do Container);

- **LayoutManager getLayout ():**

  Retorna o LayoutManager deste Container;

- **Dimension getMaximumSize ():**

  Retorna o tamanho máximo deste Container;

- **Dimension getMinimumSize ():**

  Retorna o tamanho mínimo deste Container;

- **Dimension getPreferedSize ():**

  Retorna o tamanho padrão deste Container;

- **void invalidate ():**

  Invalida este Container;

- **boolean isAncestorOf (Component C):**
  Verifica se este Container é ancestral do Component C;
- **void paint (Graphics G):**
  Pinta este Container;
- **void paintComponents (Graphics G):**
  Pinta os Component's deste Container;
- **void remove (Component C):**
  Remove o Component C deste Container;
- **void remove (int i):**
  Remove o i$^{\underline{\text{ézimo}}}$ Component deste Container;
- **void removeAll ():**
  Remove todos os Component's deste Container;
- **void removeContainerListener (ContainerListener L):**
  Cancela o registro de L como receptor de ContainerEvent's direcionados a este Container;
- **void setCursor (Cursor C):**
  Ajusta o cursor usado pelo mouse sobre este Container;
- **void setFont (Font F):**
  Ajusta o fonte utilizado para escrever neste Container;
- **void setLayout (LayoutManager M):**
  Ajusta o LayoutManager deste Container;
- **void update (Graphics G):**
  Atualiza este Container;
- **void validate ():**
  Valida este Container e todos os seus subcomponentes.

java.awt.Window

A classe java.awt.Window implementa uma janela sem bordas e sem MenuBar. O gerenciador de layout padrão para uma Window é o BorderLayout. Toda janela precisa ser de um Frame, de um Dialog ou então de outra janela. São capazes de gerar os eventos WindowOpened e WindowClosed.

Além das estruturas e do comportamento herdado da classe java.awt.Container, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **Window (Frame F):**
    Constroi uma instância da classe Window de propriedade do Frame F;
  - **Window (Window W):**
    Constroi uma instância da classe Window de propriedade da Window W;
- **Métodos:**
  - **void addWindowListener (WindowListener L):**
    Registra L como receptor de WindowEvent's desta Window;
  - **Component getFocusOwner ():**
    Retorna o componente desta Window que detem o foco se e somente se ela estiver ativa;
  - **Window[] getOwnedWindows ():**
    Retorna um vetor com as Windows de propripriedade desta Window;
  - **Window getOwner ():**
    Retorna a Window proprietária desta Window;
  - **String getWarningString ():**
    Retorna o String de advertência que é exibido com esta Window;
  - **void hide ():**
    Esconde esta janela e todos os seus subcomponentes;

- **boolean isShowing ():**
  Verifica se esta janela está exibida na tela;
- **void pack ():**
  Faz com que esta janela seja dimensionada com sua dimensão padrão (levando em conta as dimensões padrão e a disposição de todos os seus subcomponentes);
- **void removeWindowListener (WindowListener L):**
  Cancela o registro de L como receptor de WindowEvent's direcionados a esta Window;
- **void setCursor (Cursor C):**
  Ajusta o cursor usado pelo mouse sobre esta Window;
- **void show ():**
  Torna esta Window visível;
- **void toBack ():**
  Envia esta janela para trás das outras também exibidas;
- **void toFront ():**
  Traz esta janela para frente das outras também exibidas.

java.awt.Frame

A classe java.awt.Frame implementa uma janela com título, borda e, eventualmente, barra de menu. O gerenciador de layout padrão para um Frame é o BorderLayout. São capazes de gerar os eventos WindowOpened, WindowClosing, WindowClosed, WindowActivated, WindowDeactivated, WindowIconified e WindowDeiconified.

Além das estruturas e do comportamento herdado da classe java.awt.Window, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**
  - **static int ICONIFIED:**
    Representa o estado iconizado de um Frame;

- **static int NORMAL:**

  Representa o estado normal de um Frame;

- **Construtores:**

  - **Frame ():**

    Constroi uma instância da classe Frame;

  - **Frame (String T):**

    Constroi uma instância da classe Frame com título T;

- Métodos:

  - **Frame[] getFrames ():**

    Retorna um vetor com todos os Frame's criados pela aplicação;

  - **Image getIconImage ():**

    Retorna a imagem do ícone deste Frame;

  - **MenuBar getMenuBar ():**

    Retorna a MenuBar deste Frame;

  - **int getState ():**

    Retorna o estado deste Frame (Frame.ICONIFIED ou Frame.NORMAL);

  - **String getTitle ():**

    Retorna o título deste Frame;

  - **boolean isResizable ():**

    Verifica se este Frame pode ser alterado em tamanho;

  - **void remove (MenuComponent M):**

    Remove o MenuBar M deste Frame;

  - **void setIconImage (Image I):**

    Faz da Image I o ícone deste Frame;

  - **void setMenuBar (MenuBar M):**

    Torna M o MenuBar deste Frame;

- **void setResizable (boolean B):**
  Torna este Frame redimensionável ou não;
- **void setState (int S):**
  Coloca este Frame no estado S (Frame.ICONIFIED ou Frame.NORMAL);
- **void setTitle (String T):**
  Dá o título T a este Frame.

java.awt.Panel

A classe java.awt.Panel implementa um container muito útil para dividir um Container em áreas menores. Além das estruturas e do comportamento herdado da classe java.awt.Container, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **Panel ():**
    Constroi uma instância da classe Panel;
  - **Panel (LayoutManager M):**
    Constroi uma instância da classe Panel com layout gerenciado pelo LayoutManager M.

Gerenciadores de Layout do Pacote java.awt

Gerenciadores de Layout são componentes que servem ao propósito de controlar o posicionamento de objetos dentro de um Container. Layouts são nomeados para um container através da chamada do método setLayout.

java.awt.LayoutManager

Esta interface define o comportamento de um gerenciador de layout genérico. Veja abaixo a como se comunicar com instâncias das classes que a implementam:

- **Métodos:**

- **void addLayoutComponent (String N, Component C):**
  Adiciona o Component C com nome N ao layout;
- **void layoutContainer (Container C):**
  Dispõe os componentes do Container C;
- **Dimension minimumLayoutSize (Container C):**
  Calcula a dimensão mínima para o Container C, levando em conta seus componentes;
- **Dimension preferredLayoutSize (Container C):**
  Calcula a dimensão padrão do Container C, levando em conta seus componentes;
- **void removeLayoutComponent (Component C):**
  Remove o Component C do layout.

java.awt.GridLayout

Divide o container em uma matriz de áreas retangulares menores. Chamadas do método add para o container com layout controlado por um GridLayout adicionam objetos de modo a preencher a primeira linha do grid a partir da primeira coluna, e assim sucessivamente para as linhas que se sucedem.

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **GridLayout ():**
    Constroi uma instância da classe GridLayout;
  - **GridLayout (int L, int C):**
    Constroi uma instânca da classe GridLayout com L linhas e C colunas;
  - **GridLayout (int L, int C, int EH, int EV):**
    Constroi uma instância da classe GridLayout com L linhas, C colunas, EH pontos de espaçamento horizontal e EV pontos de espaçamento vertical;
- **Métodos:**

- **void addLayoutComponent (String N, Component C):**
  Adiciona o Component C com nome N ao layout;
- **int getColumns ():**
  Retorna a quantidade de colunas do GridLayout;
- **int getHgap ():**
  Retorna o espaçamento horizontal do GridLayout;
- **int getRows ():**
  Retorna a quantidade de linhas do GridLayout;
- **int getVgap ():**
  Retorna o espaçamento vertical do GridLayout;
- **void layoutContainer (Container C):**
  Dispõe os componentes do Container C;
- **Dimension minimumLayoutSize (Container C):**
  Calcula a dimensão mínima para o Container C, levando em conta seus componentes;
- **Dimension preferredLayoutSize (Container C):**
  Calcula a dimensão padrão do Container C, levando em conta seus componentes;
- **void removeLayoutComponent (Component C):**
  Remove o Component C do layout;
- **void setColumns (int C):**
  Ajusta a quantidade de colunas do GridLayout;
- **void setHgap (int EH):**
  Ajusta o espaçamento horizontal do GridLayout;
- **void setRows (int L):**
  Ajusta a quantidade de linhas do GridLayout;
- **void setVgap (int EV):**
  Ajusta o espaçamento vertical do GridLayout.

java.awt.FlowLayout

Divide o container em áreas que se sucedem, lado a lado, uma após a outra. Não cabendo mais nenhum objeto numa linha, passa-se para a próxima, e assim por diante.

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**
  - **static int CENTER:**
    Indica que cada linha de componentes deve ser centralizada;
  - **static int LEADING:**
    Indica que cada linha de componentes deve ser alinhada com o primeiro lado de seu Container (levando-se em conta a orientação do Container);
  - **static int LEFT:**
    Indica que cada linha de componentes deve ser alinhada a esquerda;
  - **static int RIGHT:**
    Indica que cada linha de componentes deve ser alinhada a direita;
  - **static int TRAILING:**
    Indica que cada linha de componentes deve ser alinhada com o último lado de seu Container (levando-se em conta a orientação do Container);
- **Construtores:**
  - **FlowLayout ():**
    Constroi uma instância da classe FlowLayout com alinhamento central e espaçamento vertical e horizontal de 5 pontos;
  - **FlowLayout (int A):**
    Constroi uma instância da classe FlowLayout com o alinhamento A dado (um dos campos acima) e espaçamento vertical e horizontal de 5 pontos;
  - **FlowLayout (int A, int EH, int EV):**
    Constroi uma instância da classe FlowLayout com o alinhamento A dado (um dos

campos acima), EH pontos de espaçamento horizontal e EV pontos de espaçamento vertical;

- **Métodos:**
  - **void addLayoutComponent (String N, Component C):**
    Adiciona o Component C com nome N ao layout;
  - **int getAlignment ():**
    Retorna o alinhamento do FlowLayout (um dos campos acima);
  - **int getHgap ():**
    Retorna o espaçamento horizontal do GridLayout;
  - **int getVgap ():**
    Retorna o espaçamento vertical do GridLayout;
  - **void layoutContainer (Container C):**
    Dispõe os componentes do Container C;
  - **Dimension minimumLayoutSize (Container C):**
    Calcula a dimensão mínima para o Container C, levando em conta seus componentes;
  - **Dimension preferredLayoutSize (Container C):**
    Calcula a dimensão padrão do Container C, levando em conta seus componentes;
  - **void removeLayoutComponent (Component C):**
    Remove o Component C do layout;
  - **void setAlignment (int A):**
    Ajusta o alinhamento do FlowLayout (um dos campos acima);
  - **void setHgap (int EH):**
    Ajusta o espaçamento horizontal do GridLayout;
  - **void setVgap (int EV):**
    Ajusta o espaçamento vertical do GridLayout.

java.awt.LayoutManager2

Esta interface extende minimamente o comportamento de um gerenciador de layout genérico previsto pela interface java.awt.LayoutManager. Além das estruturas e do comportamento herdado da interface java.awt.LayoutManager, veja abaixo como se comunicar com instâncias das classes que a implementam:

- **Métodos:**
  - **float getAlignmentX (Container C):**
    Retorna o alinhamento ao longo do eixo X;
  - **float getAlignmentY (Container C):**
    Retorna o alinhamento ao longo do eixo Y;
  - **void invalidateLayout (Container C):**
    Invalida o layout indicando que, se o gerenciador de layout armazenou informações, estas devem ser descartadas;
  - **Dimension maximumLayoutSize (Container C):**
    Calcula a dimensão máxima para o Container C, levando em conta seus componentes.

java.awt.BorderLayout

Este gerenciador de layout divide o container em 5 regiões, a de cima, a de baixo, a da esquerda, a da direita e a do centro. Dispõe 1 objeto em cada uma dessas regiões.

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**
  - **static String NORTH:**
    O Component vai no topo do Container;
  - **static String SOUTH:**
    O Component vai na base do Container;

- **static String EAST:**
  O Component vai no lado direito do Container;
- **static String WEST:**
  O Component vai no lado esquerdo do Container;
- **static String CENTER:**
  O Component vai no centro do Container;
- **static String AFTER_LAST_LINE:**
  O Component vai na direção que sucede a última linha do layout (o mesmo que SOUTH na disposição ocidental: da esquerda para a direita, de cima para baixo);
- **static String AFTER_LINE_ENDS:**
  O Component vai na direção que sucede o fim das linhas do layout (o mesmo que EAST na disposição ocidental: da esquerda para a direita, de cima para baixo);
- **static String BEFORE_FIRST_LINE:**
  O Component vai na direção que antecede a primeira linha do layout (o mesmo que NORTH na disposição ocidental: da esquerda para a direita, de cima para baixo);
- **static String BEFORE_LINE_BEGINS:**
  O Component vai na direção que atecede o início das linhas do layout (o mesmo que WEST na disposição ocidental: da esquerda para a direita, de cima para baixo);

- **Construtores:**
  - **BorderLayout ():**
    Constroi uma instância da classe BorderLayout sem espaçamento horizontal e sem espaçamento vertical;
  - **BorderLayout (int EH, int EV):**
    Constroi uma instância da classe BorderLayout com EH pontos de espaçamento horizontal e EV pontos de espaçamento vertical;

- **Métodos:**

- **void addLayoutComponent (String N, Component C):**
  Adiciona o Component C com nome N ao layout;
- **int getHgap ():**
  Retorna o espaçamento horizontal do GridLayout;
- **float getAlignmentX (Container C):**
  Retorna o alinhamento ao longo do eixo X;
- **float getAlignmentY (Container C):**
  Retorna o alinhamento ao longo do eixo Y;
- **int getVgap ():**
  Retorna o espaçamento vertical do GridLayout;
- **void invalidateLayout (Container C):**
  Invalida o layout indicando que, se o gerenciador de layout armazenou informações, estas devem ser descartadas;
- **void layoutContainer (Container C):**
  Dispõe os componentes do Container C;
- **Dimension maximumLayoutSize (Container C):**
  Calcula a dimensão máxima para o Container C, levando em conta seus componentes.
- **Dimension minimumLayoutSize (Container C):**
  Calcula a dimensão mínima para o Container C, levando em conta seus componentes;
- **Dimension preferredLayoutSize (Container C):**
  Calcula a dimensão padrão do Container C, levando em conta seus componentes;
- **void removeLayoutComponent (Component C):**
  Remove o Component C do layout;
- **void setHgap (int EH):**
  Ajusta o espaçamento horizontal do GridLayout;

- **void setVgap (int EV):**
  Ajusta o espaçamento vertical do GridLayout.

java.awt.CardLayout

Este gerenciador de layout encara cada Component no Container como um cartão. Somente um cartão está visível em um dado instante e o Container age como uma pilha de cartões. O primeiro Component adicionado ao Container será o cartão visível quando o Container for exibido. A ordem dos cartões é a ordem de adição dos Components no Container.

Este gerenciador de layout define métodos que permitem a aplicação navegar através destes cartões seqüencialmente ou então mostrar um cartão específico. Pode-se associar um String como identificador de um cartão para fins de acesso aleatório rápido.

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **CardLayout ():**
    Constroi uma instância da classe CardLayout sem espaçamento horizontal e sem espaçamento vertical;
  - **CardLayout (int EH, int EV):**
    Constroi uma instância da classe CardLayout com EH pontos de espaçamento horizontal e EV pontos de espaçamento vertical;
- **Métodos:**
  - **void addLayoutComponent (String N, Component C):**
    Adiciona o Component C com nome N ao layout;
  - **void first (Container C):**
    Vai para o primeiro cartão do Container C;
  - **int getHgap ():**
    Retorna o espaçamento horizontal do GridLayout;

- **float getAlignmentX (Container C):**
  Retorna o alinhamento ao longo do eixo X;
- **float getAlignmentY (Container C):**
  Retorna o alinhamento ao longo do eixo Y;
- **int getVgap ():**
  Retorna o espaçamento vertical do GridLayout;
- **void invalidateLayout (Container C):**
  Invalida o layout indicando que, se o gerenciador de layout armazenou informações, estas devem ser descartadas;
- **void last (Container C):**
  Vai para o último cartão do Container C;
- **void layoutContainer (Container C):**
  Dispõe os componentes do Container C;
- **Dimension maximumLayoutSize (Container C):**
  Calcula a dimensão máxima para o Container C, levando em conta seus componentes.
- **Dimension minimumLayoutSize (Container C):**
  Calcula a dimensão mínima para o Container C, levando em conta seus componentes;
- **void next (Container C):**
  Vai para o próximo cartão do Container C;
- **Dimension preferredLayoutSize (Container C):**
  Calcula a dimensão padrão do Container C, levando em conta seus componentes;
- **void previous (Container C):**
  Vai para o cartão anterior do Container C;
- **void removeLayoutComponent (Component C):**
  Remove o Component C do layout;

- **void setHgap (int EH):**
  Ajusta o espaçamento horizontal do GridLayout;
- **void setVgap (int EV):**
  Ajusta o espaçamento vertical do GridLayout;
- **void show (Container C, String N):**
  Vai para o cartão quer representa o Component adicionado com o nome N pelo método addLayoutComponent.

java.awt.GridBagLayout

A classe GridBagLayout é um gerenciador de layout flexível que alinha Component's vertical e horizontalmente, sem requisitar que os Component's tenham o mesmo tamanhoa. Cada GridBagLayout mantem uma grade retangular dinâmica de células, com cada Component ocupando uma area de exibição formada por uma ou mais células.

Cada Component gerenciado por um GridBagLayout é associado com uma instância da classe GridBagConstraints que especifica como o Component é disposto em sua área de exibição.

A forma pela qual um GridBagLayout dispõe um conjunto de Component's depende das GridBagConstraint's associadas com cada Component, e também dos tamanhos mínimo e padrão do Container dos Component's.

Para se fazer bom uso de um GridBagLayout, deve-se you must customize one or more of the GridBagConstraints associados com seus Compoente's. Isto é feito através do ajuste de uma ou mais de suas propriedades.

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **GridBagLayout ():**
    Constroi uma instância da classe GridBagLayout;
- **Métodos:**

- **void addLayoutComponent (String N, Component C):**
  Adiciona o Component C com nome N ao layout;
- **GridBagConstraints getConstraints (Component C):**
  Retorna as restrições para o Component especificado;
- **float getAlignmentX (Container C):**
  Retorna o alinhamento ao longo do eixo X;
- **float getAlignmentY (Container C):**
  Retorna o alinhamento ao longo do eixo Y;
- **int[][] getDimensions ():**
  Retorna a largura das colunas e a altura das linhas para este GridBagLayout;
- **Point getLayoutOrigin ():**
  Retorna a origem do GridBagLayout;
- **double[][] getLayoutWeights ():**
  Retorna os pesos das linhas e colunas do GridBagLayout;
- **void invalidateLayout (Container C):**
  Invalida o layout indicando que, se o gerenciador de layout armazenou informações, estas devem ser descartadas;
- **Point location (int x, int y):**
  Retorna um par ordenado (linha,coluna) que identifica a célula do GridBagLayout que contem o ponto (x,y);
- **Dimension maximumLayoutSize (Container C):**
  Calcula a dimensão máxima para o Container C, levando em conta seus componentes.
- **Dimension minimumLayoutSize (Container C):**
  Calcula a dimensão mínima para o Container C, levando em conta seus componentes;
- **Dimension preferredLayoutSize (Container C):**
  Calcula a dimensão padrão do Container C, levando em conta seus componentes;

- **void removeLayoutComponent (Component C):**
  Remove o Component C do layout;
- **void setConstraints (Component C, GridBagConstraints R):**
  Ajusta as restrições para o componente especificado do GridBagLayout.

Classes de Apoio do Pacote java.awt

java.awt.Color

Uma cor é uma classe que permite especificar e criar cores.

Cores são normalmente expressas em RGB (red-green-blue) ou em HSB (hue-saturation-brightness).

O primeiro é um padrão que especifica uma cor especificando quantidades de vermelho, verde e azul que devem ser misturadas para produzí-la. Cores podem ser expressas em RGB (1) através de 3 inteiros, respectivamente R, G e B, que podem assumir valores que vão desde 0 até 255; ou (2) através de um inteiro empacotado (com o seguinte formato: 0xffRRGGBB, onde, em hexadecimal, RR representa o nível de vermelho, GG representa o nível de verde e BB representa o nível de azul).

O segundo é um padrão que especifica uma cor especificando quantidades de hue (ou matiz), saturation (ou saturação) e bright (ou brilho). Hue é um número entre 0.0 e 1.0 (vermelho-laranja-amarelo-verde-azul-índigo-violeta), saturation é um número entre 0.0 e 1.0 (de tons pasteis a tons intensos) e bright é um número entre 0.0 e 1.0 (de branco brinlhante a preto).

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**
  - **static Color black:**
    Contém a especificação da cor preta;
  - **static Color blue:**
    Contém a especificação da cor azul;

- **static Color cyan:**

  Contém a especificação da cor azul piscina;

- **static Color darkGray:**

  Contém a especificação da cor cinza escuro;

- **static Color gray:**

  Contém a especificação da cor cinza;

- **static Color green:**

  Contém a especificação da cor verde;

- **static Color lightGray:**

  Contém a especificação da cor verde claro;

- **static Color magenta:**

  Contém a especificação da cor rosa-choque;

- **static Color orange:**

  Contém a especificação da cor laranja;

- **static Color pink:**

  Contém a especificação da cor-de-rosa;

- **static Color red:**

  Contém a especificação da cor vermelha;

- **static Color white:**

  Contém a especificação da cor branca;

- **static Color yellow:**

  Contém a especificação da cor amarela;

- **Construtores:**

  - **Color (int R, int G, int B):**

    Constroi uma instância da classe Color para representar a cor especificada em RGB por seus parâmetros;

- **Color (int C):**

  Constroi uma instância da classe Color a partir de um inteiro C que especifica uma cor em RGB (C deve ter o seguinte formato: 0xffRRGGBB, onde, em hexadecimal, RR representa o nível de vermelho, GG representa o nível de verde e BB representa o nível de azul);

- **Color (float R, float G, float B):**

  Constroi uma instância da classe Color para representar a cor especificada em RGB por seus parâmetros (números reais entre 0.0 e 1.0 que especificam percentualmente as quantidades de vermelho, verde e azul que constituem a cor);

- **Métodos:**

  - **static int HSBtoRGB (float H, float S, float B):**

    Este método retorna um inteiro RGB empacotado;

  - **static void RGBtoHSB (int R, int G, int B, float HSB []):**

    Coloca em HSB valores que correspondem, em HSB, à cor RGB expressa pelos parâmetros R, G e B;

  - **int getRed ():**

    Retorna a quantidade de vermelho (de 0 a 255) que especifica a cor;

  - **int getGreen ():**

    Retorna a quantidade de verde (de 0 a 255) que especifica a cor;

  - **int getBlue ():**

    Retorna a quantidade de azul (de 0 a 255) que especifica a cor;

  - **int getRGB ():**

    Retorna um inteiro RGB empacotado.

java.awt.Font

Uma cor é uma classe que permite especificar fontes para a escrita de caracteres. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**

- **static int BOLD:**

  Especifica fonte com o efeito negrito;

- **static int CENTER_BASELINE:**

  Especifica a linha usada como base para a escrita ideográfica (como chinês, japonês, etc);

- **static int HANGING_BASELINE:**

  Especifica a linha usada como base para a escrita em devanigiri e similares;

- **static int ITALIC:**

  Especifica fonte com o efeito itálico;

- **static int PLAIN:**

  Especifica fonte sem os efeitos negrito e itálico;

- **static int ROMAN_BASELINE:**

  Especifica a linha usada como base para a escrita romana;

- **Construtores:**

  - **Font (String N, int E, int T):**

    Constroi uma instância da classe Font para representar a fonte de letra de nome N, efeito E (Font.PLAIN, Font.BOLD, Font.ITALIC ou Font.BOLD+Font.ITALIC) e tamanho T.

- **Métodos:**

  - **boolean canDisplay (char C):**

    Verifica se o caractere C pode ser exibido neste Font;

  - **int canDisplayUpTo (char V [], int I, int Q):**

    Verifica se todos os Q caracteres de V a partir da posição I podem ser exibidos neste Font; retorna o índice do primeiro caractere que não pode ser exibido neste fonte, se houver algum, ou -1, caso todos esses caracteres possam ser exibidos neste Font;

  - **int canDisplayUpTo (String S):**

    Verifica se todos os caracteres de S podem ser exibidos neste Font; retorna o número

de ordem do primeiro caractere que não pode ser exibido neste fonte, se houver algum, ou -1, caso todos esses caracteres possam ser exibidos neste Font;

- **static Font decode (String N):**
  Retorna o Font descrito pelo nome N;

- **Font deriveFont (float T):**
  Cria um novo Font, réplica deste Font, exceto pelo tamanho T;

- **Font deriveFont (int E):**
  Cria um novo Font, réplica deste Font, exceto pelo efeito E (Font.PLAIN, Font.BOLD, Font.ITALIC ou Font.BOLD+Font.ITALIC);

- **Font deriveFont (int E, float T):**
  Cria um novo Font, réplica deste Font, exceto pelo exceto pelo efeito E (Font.PLAIN, Font.BOLD, Font.ITALIC ou Font.BOLD+Font.ITALIC) e pelo tamanho T;

- **byte getBaseLineFor (char C):**
  Retorna a linha base adequada para a exibição do caractere C com este Font;

- **String getFamily ():**
  Retorna a família deste Font;

- **static Font getFont (String P):**
  Verifica se exite, dentre as propriedades do sistema, uma de nome P que se refira a um Font, retornando este Font;

- **static Font getFont (String P, Font F):**
  Verifica se exite, dentre as propriedades do sistema, uma de nome P que se refira a um Font; em caso afirmativo, retorna este Font, e, em caso negativo, retorna o Font F;

- **String getFontName ():**
  Retorna o nome deste Font;

- **float getItalicAngle ():**
  Retorna o ângulo empregado na produção do itálico deste Font;

- **String getName ():**

  Retorna o nome lógico deste Font;

- **String getPSName ():**

  Retorna o nome PostScript deste Font;

- **int getSize ():**

  Retorna o tamanho deste Font arredondado para um inteiro;

- **float getSize2D ():**

  Retorna um número real que expressa o tamanho deste Font;

- **int getStyle ():**

  Retorna os efeitos empregados neste Font;

- **boolean hasUniformLineMetrics ():**

  Verifica se este fonte tem métrica uniforme;

- **boolean isBold ():**

  Verifica se este fonte emprega o efeito negrito;

- **boolean isItalic ():**

  Verifica se este fonte emprega o efeito itálico;

- **boolean isPlain ():**

  Verifica se este fonte não emprega os efeitos negrito nem itálico;

java.awt.Cursor

Esta classe encapsula a representação bitmap do cursor do mouse. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**

  - **static int CROSSHAIR_CURSOR:**

    Representa o Cursor do tipo crosshair;

  - **static int CUSTOM_CURSOR:**

    Representa Cursor's definidos pelo usuário;

- **static int DEFAULT_CURSOR:**
  Representa o Cursor padrão;
- **static int E_RESIZE_CURSOR:**
  Representa o Cursor de redimensionamento com ponto de pega na borda direita;
- **static int HAND_CURSOR:**
  Representa o Cursor do tipo mão;
- **static int MOVE_CURSOR:**
  Representa o Cursor de movimentação;
- **static int N_RESIZE_CURSOR:**
  Representa o Cursor de redimensionamento com ponto de pega na borda superior;
- **static int N_RESIZE_CURSOR:**
  Representa o Cursor de redimensionamento com ponto de pega na borda superior;
- **static int NE_RESIZE_CURSOR:**
  Representa o Cursor de redimensionamento com ponto de pega no canto superior esquerdo;
- **static int NW_RESIZE_CURSOR:**
  Representa o Cursor de redimensionamento com ponto de pega no canto superior direito;
- **static int S_RESIZE_CURSOR:**
  Representa o Cursor de redimensionamento com ponto de pega na borda inferior;
- **static int SE_RESIZE_CURSOR:**
  Representa o Cursor de redimensionamento com ponto de pega no canto inferior esquerdo;
- **static int SW_RESIZE_CURSOR:**
  Representa o Cursor de redimensionamento com ponto de pega no canto inferior direito;

- **static int NE_RESIZE_CURSOR:**
  Representa o Cursor de redimensionamento com ponto de pega no canto superior esquerdo;
- **static int TEXT_CURSOR:**
  Representa o Cursor de texto;
- **static int NE_RESIZE_CURSOR:**
  Representa o Cursor de redimensionamento com ponto de pega no canto superior esquerdo;
- **static int W_RESIZE_CURSOR:**
  Representa o Cursor de redimensionamento com ponto de pega na borda esquerda;
- **static int WAIT_CURSOR:**
  Representa o Cursor de espera;

- **Construtores:**
  - **Cursor (int T):**
    Constroi uma instância da classe Cursor do tipo T especificado (um dos campos acima);
- **Métodos:**
  - **static Cursor getDefaultCursor ():**
    Retorna o Cursor padrão;
  - **String getName ():**
    Retorna o nome deste Cursor;
  - **static Cursor getPredefinedCursor (int T):**
    Retorna o Cursor predefinido de tipo T (um do campos acima);
  - **static Cursor getSystemCustomCursor (String N):**
    Retorna o Cursor de nome N (específico do sistema do usuário);

– **int getType ():**

Retorna o tipo deste Cursor (um dos campos acima).

java.awt.Insets

Esta classe especifica o espaço que um Container deve reservar para cada um de seus lados. Tal espaço pode se destinar a uma borda, pode ficar em branco, ou pode conter um título. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**

  - **int bottom:**

    Representa o espaço ocupado pelo dado inferior de um Container;

  - **int left:**

    Representa o espaço ocupado pelo dado esquerdo de um Container;

  - **int right:**

    Representa o espaço ocupado pelo dado direito de um Container;

  - **int top:**

    Representa o espaço ocupado pelo dado superior de um Container;

- **Construtores:**

  – **Insets (int S, int E, int I, int D):**

  Constroi uma instância da classe Insets para representar as laterais de um Container com um espaço S ocupado pelo lado superior, com um espaço E ocupado pelo lado esquerdo, com um espaço I ocupado pelo lado inferior e com um espaço D ocupado pelo lado direito.

java.awt.GridBagConstraints

Esta classe especifica restrições para Components que são dispostos em um Container por um GridBagLayout. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**
  - **int anchor:**
    Este campo é usado quando o Compoent é menor que sua área de exibição;
  - **static int BOTH:**
    Redimensiona o Component tanto horizontal como verticalmente;
  - **static int CENTER:**
    Posiciona o Component no meio da área de exibição;
  - **static int EAST:**
    Posiciona o Component à direita da área de exibição, centralizado verticalmente;
  - **int fill:**
    Este campo é usado quando a área de exibição do Component é maior do que o tamanho do Component;
  - **int gridheight:**
    Especifica a quantidade de células em uma coluna para a área de exibição do Component;
  - **int gridwidth:**
    Especifica a quantidade de células em uma linha para a área de exibição do Component;
  - **int gridx:**
    Especifica a célula da esquerda da área de exibição do Component (a célula mais à esquerda tem número zero);
  - **int gridy:**
    Especifica a célula do topo da área de exibição do Component (a célula mais ao topo tem número zero);
  - **static int HORIZONTAL:**
    Redimensiona o Component horizontalmente mas não verticalmente;

- **Insets insets:**

  Este campo especifica a distância entre o Component e os lados de sua área de exibição;

- **int ipadx:**

  Este campo especifica quanto espaço adicionar à largura mínima do Component;

- **int ipady:**

  Este campo especifica quanto espaço adicionar à altura mínima do Component;

- **static int NONE:**

  Não redimensiona o Component;

- **static int NORTH:**

  Posiciona o Component no topo de sua área de exibição, centralizado horizontalmente;

- **static int NORTHEAST:**

  Posiciona o Component no canto superior direito de sua área de exibição;

- **static int NORTHWEST:**

  Posiciona o Component no canto superior esquerdo de sua área de exibição;

- **static int RELATIVE:**

  Posiciona o Component ao lado do último Component de sua linha ou de sua coluna (gridwidtdh/gridheight), ou ao lado do último Component adicionado (gridx/gridy);

- **static int REMAINDER:**

  Posiciona o Component como último de sua linha ou coluna;

- **static int SOUTH:**

  Posiciona o Component na base de sua área de exibição, centralizado horizontalmente;

- **static int SOUTHEAST:**

  Posiciona o Component no canto inferior direito de sua área de exibição;

- **static int SOUTHWEST:**

  Posiciona o Component no canto inferior esquerdo de sua área de exibição;

- **static int VERTICAL:**

  Redimensiona o Component verticalmente mas não horizontalmente;

- **double weightx:**

  Especifica como distribuir um eventual espaço horizontal extra;

- **double weighty:**

  Especifica como distribuir um eventual espaço vertical extra;

- **static int WEST:**

  Posiciona o Component à esquerda da área de exibição, centralizado verticalmente.

- **Construtores:**

  - **GridBagLayoutConstraints ():**

    Constroi uma instância da classe GridBagLayoutConstraints especificando o padrão para todas as restrições;

  - **GridBagLayoutConstraints (int gridx, int gridy,**
    int gridwidth, int gridheight,
    double weightx, double weighty,
    int anchor, int fill, Insets insets,
    int ipadx, int ipady):

    Constroi uma instância da classe GridBagLayoutConstraints com todos os campos ajustados aos valores fornecidos.

java.awt.Graphics

Esta classe abstrata representa um contexto gráfico que pode ser usado pelas aplicações para traçar texto, desenhos e imagens em um componente. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com instâncias de suas classes derivadas:

- **Métodos:**

  - **void clearRect (int x, int y, int L, int A):**

    Limpa o retângulo que se origina no ponto (x,y) e tem largura L e altura A com a cor de fundo corrente da superfície de desenho;

- **void copyArea (int x, int y, int L, int A, int dx, int dy):**
  Copia a área contida no retângulo que se origina no ponto (x,y) e tem largura L e altura A para o ponto (x+dx , y+dy);

- **Graphics create ():**
  Retorna um novo contexto gráfico que é cópia daquele representado por este Graphics;

- **Graphics create (int x, int y, int L, int A):**
  Retorna um novo contexto gráfico que é cópia daquele representado por este Graphics limitado ao retângulo que se origina no ponto (x,y) e tem largura L e altura A;

- **void dispose ():**
  Libera o contexto gráfico representado por este Graphics, liberando também os recursos que ele estava usando;

- **void draw3DRect (int x, int y, int L, int A, boolean E):**
  Traça o retângulo que se origina no ponto (x,y) e tem largura L e altura A; E especifica se o retângulo deve parecer elevar-se da superfície do desenho;

- **void drawArc (int x, int y, int L, int A, int AI, int AA):**
  Traça o arco circular ou elíptico limitado ao retângulo que se origina no ponto (x,y) e tem largura L e altura A, a partir do ângulo inicial AI e varrendo o ângulo AA;

- **void drawBytes (byte V [], int P, int Q, int x, int y):**
  Traça o texto especificado pelos Q bytes do vetor V a partir da posição P no ponto (x,y);

- **void drawChars (char V [], int P, int Q, int x, int y):**
  Traça o texto especificado pelos Q caracteres do vetor V a partir da posição P no ponto (x,y);

- **void drawImage (Image I, int x, int y, Color C, this):**
  Traça a imagem transparente representada pela Image I (com a cor de fundo C), a partir do ponto (x,y); o último parâmetro deve ser sempre this; a imagem é redimensionada para caber na área de desenho;

- **void drawImage (Image I, int x, int y, this):**
  Traça a imagem representada pela Image I, a partir do ponto (x,y); o último parâmetro deve ser sempre this; a imagem é redimensionada para caber na área de desenho;

- **void drawImage (Image I, int x, int y, int L, int A, Color C, this):**
  Traça a imagem transparente representada pela Image I (com a cor de fundo C), dentro da área retangular que se origina no ponto (x,y) e tem largura L e altura A; o último parâmetro deve ser sempre this; a imagem é redimensionada para caber na área retangular especificada;

- **void drawImage (Image I, int x, int y, int L, int A, this):**
  Traça a imagem representada pela Image I, dentro da área retangular que se origina no ponto (x,y) e tem largura L e altura A; o último parâmetro deve ser sempre this; a imagem é redimensionada para caber na área retangular especificada;

- **void drawImage (Image I,**
  int Dx1, int Dy1, int Dx2, int Dy2,
  int Ox1, int Oy1, int Ox2, int Oy2,
  Color C, this):
  Traça a região retangular com canto superior esquerdo (Ox1 , Oy1) e canto inferior direito em (Ox2 , Oy2) da imagem transparente representada pela Image I (com a cor de fundo C), dentro da área retangular com canto superior esquerdo em (Dx1 , Dy1) e canto inferior direito em (Dx2 , Dy2); o último parâmetro deve ser sempre this; a imagem é redimensionada para caber na área retangular especificada;

- **void drawImage (Image I,**
  int Dx1, int Dy1, int Dx2, int Dy2,
  int Ox1, int Oy1, int Ox2, int Oy2,
  this):
  Traça a região retangular com canto superior esquerdo (Ox1 , Oy1) e canto inferior direito em (Ox2 , Oy2) da imagem representada pela Image I, dentro da área retangular com canto superior esquerdo em (Dx1 , Dy1) e canto inferior direito em (Dx2 , Dy2); o último parâmetro deve ser sempre this; a imagem é redimensionada para caber na área retangular especificada;

- **void drawLine (int x1, int y1, int x2, int y2):**
  Traça uma linha do ponto (x1 , y1) ao ponto (x2 , y2);

- **void drawOval (int x, int y, int L, int A):**
  Traça uma elipse limitada dentro da região retangular que se origina no ponto (x,y) e tem largura L e altura A; linha do ponto (x1 , y1) ao ponto (x2 , y2);

- **void drawPolygon (int X [], int Y [], int Q):**
  Traça um polígono fechado que passa pelos Q vértices cujas coordenadas se encontram nos vetors X e Y;

- **void drawPolyline (int X [], int Y [], int Q):**
  Traça uma seqüência de linhas que passa pelos Q pontos cujas coordenadas se encontram nos vetors X e Y;

- **void drawRect (int x, int y, int L, int A):**
  Traça um retângulo que se origina no ponto (x,y) e tem largura L e altura A;

- **void drawRoundRect (int x, int y, int L, int A, int LA, int AA):**
  Traça um retângulo com bordas arredondadas que se origina no ponto (x,y) e tem largura L e altura A; os arcos dos cantos terão largura LA e altura AA;

- **void drawString (String S, int x, int y):**
  Traça o texto representado pelo String S a partir do ponto (x,y);

- **void fill3DRect (int x, int y, int L, int A, boolean E):**
  Pinta o retângulo que se origina no ponto (x,y) e tem largura L e altura A; E especifica se o retângulo deve parecer elevar-se da superfície do desenho;

- **void fillArc (int x, int y, int L, int A, int AI, int AA):**
  Pinta o arco circular ou elíptico limitado ao retângulo que se origina no ponto (x,y) e tem largura L e altura A, a partir do ângulo inicial AI e varrendo o ângulo AA;

- **void fillOval (int x, int y, int L, int A):**
  Pinta uma elipse limitada dentro da região retangular que se origina no ponto (x,y) e tem largura L e altura A; linha do ponto (x1 , y1) ao ponto (x2 , y2);

- **void fillPolygon (int X [], int Y [], int Q):**
  Pinta um polígono fechado que passa pelos Q vértices cujas coordenadas se encontram nos vetors X e Y;

- **void fillRect (int x, int y, int L, int A):**
  Pinta um retângulo que se origina no ponto (x,y) e tem largura L e altura A;

- **void fillRoundRect (int x, int y, int L, int A, int LA, int AA):**
  Pinta um retângulo com bordas arredondadas que se origina no ponto (x,y) e tem largura L e altura A; os arcos dos cantos terão largura LA e altura AA;

- **Color getColor ():**
  Retorna uma instância da classe Color que representa a cor usada para traçar e pintar;

- **Font getFont ():**
  Retorna uma instância da classe Font que representa a fonte utilizada para traçar texto;

- **FontMetrics getFontMetrics ():**
  Retorna as métricas empregadas no fonte corrente;

- **FontMetrics getFontMetrics (Font F):**
  Retorna as métricas empregadas no fonte especificado;

- **void setColor (Color C):**
  Ajusta a cor usada para traçar e pintar;

- **void setFont (Font F):**
  Ajusta a fonte utilizada para traçar texto;

- **void setPaintMode ():**
  Configura este contexto gráfico para empregar sua cor corrente nos traçados;

- **void setXORMode (Color C):**
  Configura este contexto gráfico para intercalar sua cor corrente com a cor especificada nos traçados;

- **void translate (int x, int y):**

  Translada a origem deste contexto gráfico para o ponto (x,y) do sistema de coordenadas corrente.

java.awt.Point

Esta classe representa um lugar no plano através de coordenadas (x,y) inteiras. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**
  - **int x:**

    Representa a coordenada x deste Point;
  - **int y:**

    Representa a coordenada y deste Point;
- **Construtores:**
  - **Point ():**

    Constroi uma instância da classe Point para representar a origem do plano;
  - **Point (int x, int y):**

    Constroi uma instância da classe Point para representar o Point de coordenadas (x,y);
  - **Point (Point P):**

    Constroi uma instância da classe Point para representar o mesmo Point que P;
- **Métodos:**
  - **double distance (double x, double y):**

    Retorna a distância deste Point ao Point de coordenadas (x,y);
  - **static double distance (double x1, double y1, double x2, double y2):**

    Retorna a distância entre os Point's de coordenadas (x_1,y_1) e (x_2,y_2);
  - **double distance (Point P):**

    Retorna a distância deste Point ao Point P;

- **double distanceSq (double x, double y):**
  Retorna o quadrado da distância deste Point ao Point de coordenadas (x,y);
- **static double distanceSq (double x1, double y1, double x2, double y2):**
  Retorna o quadrado da distância entre os Point's (x_1,y_1) e (x_2,y_2);
- **double distanceSq (Point P):**
  Retorna o quadrado da distância deste Point ao Point P;
- **Point getLocation ():**
  Retorna a localização deste Point;
- **double getX ():**
  Retorna a coordenada X deste Point em dupla precisão;
- **double getY ():**
  Retorna a coordenada Y deste Point em dupla precisão;
- **void move (int x, int y):**
  Move este Point para o Point (x,y);
- **void setLocation (double x, double y):**
  Ajusta as coordenadas x e y deste Point;
- **void setLocation (int x, int y):**
  Ajusta as coordenadas x e y deste Point;
- **void setLocation (Point P):**
  Ajusta as coordenadas x e y deste Point;
- **void translate (int dx, int dy):**
  Translada este Point.

java.awt.Dimension

Esta classe encapsula a largura e a altura de um Component. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**

- **int height:**

  Representa a altura representada por esta Dimension;

- **int width:**

  Representa a largura representada por esta Dimension;

- **Construtores:**

  - **Dimension ():**

    Constroi uma instância da classe Dimension para representar altura igual a zero e largura também igual a zero;

  - **Dimension (Dimension D):**

    Constroi uma instância da classe Dimension para representar uma dimensão identica àquela representada pela Dimension D dada;

  - **Dimension (int L, int A):**

    Constroi uma instância da classe Dimension para representar altura igual a A e largura igual a L;

- **Métodos:**

  - **double getHeight ():**

    Retorna a altura desta Dimension;

  - **Dimension getSize ():**

    Retorna a dimensão desta Dimension;

  - **double getWidth ():**

    Retorna a largura desta Dimension;

  - **void setSize (Dimension D):**

    Ajusta a dimensão desta Dimension para dimensões idênticas àquelas representadas pela Dimension D dada;

  - **void setSize (double A, double L):**

    Ajusta a largura desta Dimension para altura igual a A e largura igual a L;

- **void setSize (int A, int L):**

  Ajusta a largura desta Dimension para altura igual a A e largura igual a L.

java.awt.Rectangle

Esta classe representa uma área no plano através das coordenadas de seu cantu superior direito (x,y) e de sua largura e altura. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**

  - **int x:**

    Representa a coordenada x deste Rectangle;

  - **int y:**

    Representa a coordenada y deste Rectangle;

  - **int height:**

    Representa a altura deste Rectangle;

  - **int width:**

    Representa a largura deste Rectangle;

- **Construtores:**

  - **Rectangle ():**

    Constroi uma instância da classe Rectangle para representar um retângulo com canto superior esquerdo na origem do plano, algura igual a zero e largura também igual a zero;

  - **Rectangle (Dimension D):**

    Constroi uma instância da classe Rectangle para representar um retângulo com canto superior esquerdo na origem do plano e com dimensão D;

  - **Rectangle (int L, int A):**

    Constroi uma instância da classe Rectangle para representar um retângulo com canto superior esquerdo na origem do plano, altura igual a A e largura igual a L;

- **Rectangle (int x, int y, int L, int A):**
  Constroi uma instância da classe Rectangle para representar um retângulo com canto superior esquerdo no ponto (x,y), altura igual a A e largura igual a L;

- **Rectangle (Point P):**
  Constroi uma instância da classe Rectangle para representar um retângulo com canto superior esquerdo no ponto P, altura igual a zero e largura também igual a zero;

- **Rectangle (Point P, Dimension D):**
  Constroi uma instância da classe Rectangle para representar um retângulo com canto superior esquerdo no ponto P e dimensão D;

- **Rectangle (Rectangle R):**
  Constroi uma instância da classe Rectangle para representar um retângulo idêntico ao Rectangle R;

- **Métodos:**

  - **void add (int x, int y):**
    Adiciona o ponto (x,y) a este Rectangle (que passa a ser o menor retângulo que contenha o sua instância inicial e o ponto (x,y));

  - **void add (Point P):**
    Adiciona o ponto P a este Rectangle (que passa a ser o menor retângulo que contenha o sua instância inicial e o ponto P);

  - **void add (Rectangle R):**
    Adiciona o retângulo R a este Rectangle (que passa a ser o menor retângulo que contenha o sua instância inicial e o retângulo R);

  - **boolean contains (int x, int y):**
    Verifica se este Rectangle contem o ponto (x,y);

  - **boolean contains (int x, int y, int L, int A):**
    Verifica se este Rectangle contem o retângulo que tem seu canto superior esquerdo no ponto (x,y), largura L e altura A;

- **boolean contains (Point P):**

  Verifica se este Rectangle contem o ponto P;

- **boolean contains (Rectangle R):**

  Verifica se este Rectangle contem o Rectangle R;

- **Rectangle createIntersection (Rectangle R):**

  Retorna a interseção deste Rectangle com o Rectangle R;

- **Rectangle createUnion (Rectangle R):**

  Retorna a união deste Rectangle com o Rectangle R;

- **Rectangle getBounds ():**

  Retorna o Rectangle que limita este Rectangle;

- **double getHeight ():**

  Retorna a altura deste Rectangle em dupla precisão;

- **Point getLocation ():**

  Retorna a localização do canto superior esquerdo deste Rectangle;

- **Dimension getSize ():**

  Retorna a dimensão deste Rectangle;

- **double getWidth ():**

  Retorna a largura deste Rectangle em dupla precisão;

- **double getX ():**

  Retorna com dupla precisão a coordenada X do Rectangle que limita este Rectangle;

- **double getY ():**

  Retorna com dupla precisão a coordenada Y do Rectangle que limita este Rectangle;

- **void grow (int dL, int dA):**

  Redimensiona este Rectangle;

- **Rectangle intersection (Rectangle R):**

  Retorna a interseção deste Rectangle com o Rectangle R;

- **boolean intersects (Rectangle R):**

  Verifica se há interseção deste Rectangle com o Rectangle R dado;

- **boolean isEmpty ():**

  Verifica se este Rectangle é vazio;

- **void setBounds (int x, int y, int L, int A):**

  Ajusta os limites deste Rectangle (para a posição (x,y), largura L e altura A);

- **void setBounds (Rectangle R):**

  Ajusta os limites deste Rectangle para limites idênticos aos do Rectangle R;

- **void setLocation (int x, int y):**

  Ajusta a localização deste Rectangle (para a posição (x,y));

- **void setLocation (Point P):**

  Ajusta a localização deste Resctangle (para a posição dada pelo Point P);

- **void setRect (int x, int y, int L, int A):**

  O mesmo que setBounds (int x, int y, int L, int A);

- **void setSize (Dimension D):**

  Ajusta a dimensão deste Rectangle (para a dimensão dada pela Dimension D);

- **void setSize (int L, int A):**

  Ajusta a dimensão deste Rectangle (para largura L e altura A);

- **void translate (int dx, int dy):**

  Translada este Rectangle;

- **Rectangle union (Rectangle R):**

  Retorna a união deste Rectangle com o Rectangle R.

Dinâmica de uma Aplicação Gráfica

Dizemos que aplicações gráficas são orientadas a eventos, i.e., não são programas construídos para executar de modo linear, e sim para serem capazes de responder às ações do usuário.

Um evento é comunicação, uma indicação, de que algo ocorreu no mundo exterior. Eles tem origem nas ações que o usuário pode exercer nos componentes e podem ser detectados por detectores de eventos e tratados pela aplicação.

Detectores de Evento

Detectores de evento são classes que podem dar origem a instância capazes de detectar eventos e encaminhá-los a tratadores de evento.

Existem diversos tipos de detectores de eventos, um para cada tipo de evento (evento de mouse, evento de teclado, etc), assim como existem, para cada detector de evento, diversos tipos de tratadores, um para cada tipo de ação que pode ser realizada pelo usuário (no caso do evento de mouse, tratador de pressionamento de botão, tratador de liberação de botão, etc).

Para emprega-los no tratamento de eventos, deveremos ter uma classe derivada (anônima ou nomeada) de um deles na qual reimplementemos o método tratador correspondente ao evento que desejamos tratar; uma instância desta classe derivada deverá ser nomeada *listener* o objeto que desejamos que seja sensível a eventos.

Veja abaixo a lista dos detectores de evento, acompanhados da lista de seus respectivos tratadores de ação:

1. **ComponentAdapter:**

   - **void componentHidden (ComponentEvent e):**
     É invocado sempre que o componente torna-se invisível;
   - **void componentMoved (ComponentEvent e):**
     É invocado sempre que a posição do componente muda;
   - **void componentResized (ComponentEvent e):**
     É invocado sempre que o componente muda de tamanho;
   - **void componentShown (ComponentEvent e):**
     É invocado sempre que o componete torna-se visível;

2. **ContainerAdapter:**

- **void componentAdded (ContainerEvent e):**

  É invocado sempre que o componente é adicionado a um container;

- **void componentRemoved (ContainerEvent e):**

  É invocado sempre que o componente é removido de um container.

3. **FocusAdapter:**

- **void focusGained (FocusEvent e):**

  É invocado sempre que o componente ganha o foco;

- **void focusLost (FocusEvent e):**

  É invocado sempre que o componente perde o foco;

4. **KeyAdapter:**

- **void keyPressed (KeyEvent e):**

  É invocado sempre que uma tecla foi pressionada;

- **void keyReleased (KeyEvent e):**

  É invocado sempre que uma tecla foi liberada;

- **void keyTyped (KeyEvent e):**

  É invocada sempre que uma tecla foi digitada;

5. **MouseAdapter:**

- **void mouseClicked (MouseEvent e):**

  É invocado sempre que o usuário clica com o mouse sobre o componente;

- **void mouseEntered (MouseEvent e):**

  É invocado sempre que o usuário leva o cursor do mouse sobre o componente;

- **void mouseExited (MouseEvent e):**

  É invocado sempre que o usuário retira o cursor do mouse de sobre o componente;

- **void mousePressed (MouseEvent e):**

  É invocado sempre que o usuário pressiona o botão do mouse sobre o componente;

- **void mouseReleased (MouseEvent e):**
  É invocado sempre que o usuário libera o botão do mouse que mantinha pressionado sobre o componente;

6. **MouseInputAdapter:**

- **void mouseClicked (MouseEvent e):**
  É invocado sempre que o usuário clica com o mouse sobre o componente;
- **void mouseDragged (MouseEvent e):**
  É invocado sempre que um botão do mouse é pressionado sobre o componente e então o mouse é arrastado;
- **void mouseEntered (MouseEvent e):**
  É invocado sempre que o usuário leva o cursor do mouse sobre o componente;
- **void mouseExited (MouseEvent e):**
  É invocado sempre que o usuário retira o cursor do mouse de sobre o componente;
- **void mouseMoved (MouseEvent e):**
  É invocado sempre que o mouse se move por sobre o componente (sem nenhum botão pressionado);
- **void mousePressed (MouseEvent e):**
  É invocado sempre que o usuário pressiona o botão do mouse sobre o componente;
- **void mouseReleased (MouseEvent e):**
  É invocado sempre que o usuário libera o botão do mouse que mantinha pressionado sobre o componente;

7. **MouseMotionAdapter:**

- **void mouseMoved (MouseEvent e):**
  É invocado sempre que o mouse se move por sobre o componente (sem nenhum botão pressionado);

- **void mouseDragged (MouseEvent e):**

  É invocado sempre que um botão do mouse é pressionado sobre o componente e então o mouse é arrastado;

8. WindowAdapter:

- **void windowActivated (windowEvent e):**

  É invocado sempre que a janela é ativada;

- **void windowClosed (windowEvent e):**

  É invocado sempre que a janela é fechada;

- **void windowClosing (windowEvent e):**

  É invocado sempre que é solicitado o fechamento da janela;

- **void windowDeactivated (windowEvent e):**

  É invocado sempre que a janela é desativada;

- **void windowDeiconified (windowEvent e):**

  É invocado sempre que a janela é desiconificada;

- **void windowIconified (windowEvent e):**

  É invocado sempre que a janela é iconificada;

- **void windowOpened (windowEvent e):**

  É invocado sempre que a janela é aberta.

Para fazer uso desses detectores de evento, tudo o que temos que fazer é, para cada componente que deve ser sensível a eventos, registrar como detector de evento uma instância de uma classe anônima que deriva do detector desejado e que redefine o método que trata o evento desejado.

[C:\ExplsJava\Expl_07\Calculadora.java]

```
import java.awt.*;
import java.awt.event.*;

class Calculadora
{
    private Frame  Janela  = new Frame  ();
```

```
    private Label  Display = new Label  ("", Label.RIGHT);

    private Button B0      = new Button ("0"),
                   B1      = new Button ("1"),
                   B2      = new Button ("2"),
                   B3      = new Button ("3"),
                   B4      = new Button ("4"),
                   B5      = new Button ("5"),
                   B6      = new Button ("6"),
                   B7      = new Button ("7"),
                   B8      = new Button ("8"),
                   B9      = new Button ("9"),
                   BA      = new Button ("+"),
                   BS      = new Button ("-"),
                   BM      = new Button ("*"),
                   BD      = new Button ("/"),
                   BI      = new Button ("="),
                   BL      = new Button ("C");

    private boolean Primeiro = true;

    private double  Operando1;
    private char    Operador = ' ';

    private void TrateClickEmBotaoNumerico (char N)
    {
        if (this.Primeiro)
        {
            this.Display.setText ("" + N);
            this.Primeiro = false;
        }
        else
            this.Display.setText (this.Display.getText () + N);
    }

    private void TrateClickEmBotaoDeOperacao (char O)
    {
        if (!this.Display.getText ().equals (""))
        {
            this.TrateClickNoBotaoIgual ();

            this.Operando1 = new Double
                           (this.Display.getText ())
                           .doubleValue ();

            this.Operador = O;
        }
    }

    private void TrateClickNoBotaoIgual ()
    {
        if ((!(this.Display.getText ().equals (""))) && (this.Operador != ' '))
        {
            double Operando2 = new Double
                             (this.Display.getText ()).doubleValue (),

                   Resultado = 0;

            switch (this.Operador)
            {
                case '+':
                    Resultado = this.Operando1 + Operando2;
                    break;

                case '-':
                    Resultado = this.Operando1 - Operando2;
                    break;

                case '*':
```

```
                    Resultado = this.Operando1 * Operando2;
                    break;

                case '/':
                    Resultado = this.Operando1 / Operando2;
            }

            if (Math.round (Resultado) == Resultado)
                this.Display.setText ("" + Math.round (Resultado));
            else
                this.Display.setText ("" + Resultado);

            this.Operador = ' ';
        }

        this.Primeiro = true;
    }

    private void TrateClickNoBotaoDeLimpar ()
    {
        this.Display.setText ("");

        this.Operador = ' ';
        this.Primeiro = true;
    }

    private void TrateFechamentoDaJanela ()
    {
        System.exit (0);
    }

    public Calculadora ()
    {
        BorderLayout LayoutCalculadora = new BorderLayout ();

        this.Janela.setTitle  ("Calculadora");
        this.Janela.setSize   (200, 200);
        this.Janela.setLayout (LayoutCalculadora);

        Panel      Botoes       = new Panel ();
        GridLayout LayoutBotoes = new GridLayout (4, 4);

        Botoes.setLayout (LayoutBotoes);

        Botoes.add (this.B7);
        Botoes.add (this.B8);
        Botoes.add (this.B9);
        Botoes.add (this.BA);
        Botoes.add (this.B4);
        Botoes.add (this.B5);
        Botoes.add (this.B6);
        Botoes.add (this.BS);
        Botoes.add (this.B1);
        Botoes.add (this.B2);
        Botoes.add (this.B3);
        Botoes.add (this.BM);
        Botoes.add (this.B0);
        Botoes.add (this.BI);
        Botoes.add (this.BL);
        Botoes.add (this.BD);

        this.Janela.add ("North",  this.Display);
        this.Janela.add ("Center", Botoes);

        this.B0.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
```

```
                {
                    TrateClickEmBotaoNumerico ('0');
                }
            }
        );

        this.B1.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('1');
                }
            }
        );

        this.B2.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('2');
                }
            }
        );

        this.B3.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('3');
                }
            }
        );

        this.B4.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('4');
                }
            }
        );

        this.B5.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('5');
                }
            }
        );

        this.B6.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('6');
                }
```

```
        }
    );

    this.B7.addMouseListener
    (
        new MouseAdapter ()
        {
            public void mouseReleased (MouseEvent e)
            {
                TrateClickEmBotaoNumerico ('7');
            }
        }
    );

    this.B8.addMouseListener
    (
        new MouseAdapter ()
        {
            public void mouseReleased (MouseEvent e)
            {
                TrateClickEmBotaoNumerico ('8');
            }
        }
    );

    this.B9.addMouseListener
    (
        new MouseAdapter ()
        {
            public void mouseReleased (MouseEvent e)
            {
                TrateClickEmBotaoNumerico ('9');
            }
        }
    );

    this.BA.addMouseListener
    (
        new MouseAdapter ()
        {
            public void mouseReleased (MouseEvent e)
            {
                TrateClickEmBotaoDeOperacao ('+');
            }
        }
    );

    this.BS.addMouseListener
    (
        new MouseAdapter ()
        {
            public void mouseReleased (MouseEvent e)
            {
                TrateClickEmBotaoDeOperacao ('-');
            }
        }
    );

    this.BM.addMouseListener
    (
        new MouseAdapter ()
        {
            public void mouseReleased (MouseEvent e)
            {
                TrateClickEmBotaoDeOperacao ('*');
            }
        }
    );
```

```
        this.BD.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoDeOperacao ('/');
                }
            }
        );

        this.BI.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickNoBotaoIgual ();
                }
            }
        );

        this.BL.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickNoBotaoDeLimpar ();
                }
            }
        );

        this.Janela.addWindowListener
        (
            new WindowAdapter ()
            {
                public void windowClosing (WindowEvent e)
                {
                    TrateFechamentoDaJanela ();
                }
            }
         );

        this.Janela.show ();
    }
}
```

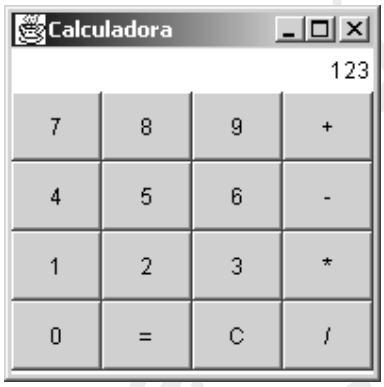
[C:\ExplsJava\Expl_07\TesteDeCalculadora.java]

```
class TesteDeCalculadora
{
    public static void main (String Args [])
    {
        Calculadora calculadora = new Calculadora();
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_07> javac TesteDeCalculadora.java
C:\ExplsJava\Expl_07> javaw TesteDeCalculadora
```

Isto fará com que se abra na tela a seguinte janela:



O usuário poderá interagir com ela de forma semelhante àquela que emprega quando usa uma calculadora simples.

Em vez de declarar e instanciar um objeto da classe Frame interno à classe Calculadora, poder-se-ia fazer com que esta última derivasse da classe Frame, passando ela própria a exibir características de janela.

[C:\ExplsJava\Expl_08\Calculadora.java]

```
import java.awt.*;
import java.awt.event.*;

class Calculadora extends Frame
{
    private Label  Display = new Label  ("", Label.RIGHT);

    private Button B0     = new Button ("0"),
                   B1     = new Button ("1"),
                   B2     = new Button ("2"),
                   B3     = new Button ("3"),
                   B4     = new Button ("4"),
                   B5     = new Button ("5"),
                   B6     = new Button ("6"),
                   B7     = new Button ("7"),
                   B8     = new Button ("8"),
                   B9     = new Button ("9"),
                   BA     = new Button ("+"),
                   BS     = new Button ("-"),
                   BM     = new Button ("*"),
                   BD     = new Button ("/"),
                   BI     = new Button ("="),
                   BL     = new Button ("C");

    private boolean Primeiro = true;

    private double  Operando1;
    private char    Operador = ' ';

    private void TrateClickEmBotaoNumerico (char N)
    {
```

```
        if (this.Primeiro)
        {
            this.Display.setText ("" + N);
            this.Primeiro = false;
        }
        else
            this.Display.setText (this.Display.getText () + N);
    }

    private void TrateClickEmBotaoDeOperacao (char O)
    {
        if (!this.Display.getText ().equals (""))
        {
            this.TrateClickNoBotaoIgual ();

            this.Operando1 = new Double
                            (this.Display.getText ())
                            .doubleValue ();

            this.Operador = O;
        }
    }

    private void TrateClickNoBotaoIgual ()
    {
        if ((!(this.Display.getText ().equals (""))) && (this.Operador != ' '))
        {
            double Operando2 = new Double
                               (this.Display.getText ()).doubleValue (),

                   Resultado = 0;

            switch (this.Operador)
            {
                case '+':
                    Resultado = this.Operando1 + Operando2;
                    break;

                case '-':
                    Resultado = this.Operando1 - Operando2;
                    break;

                case '*':
                    Resultado = this.Operando1 * Operando2;
                    break;

                case '/':
                    Resultado = this.Operando1 / Operando2;
            }

            if (Math.round (Resultado) == Resultado)
                this.Display.setText ("" + Math.round (Resultado));
            else
                this.Display.setText ("" + Resultado);

            this.Operador = ' ';
        }

        this.Primeiro = true;
    }

    private void TrateClickNoBotaoDeLimpar ()
    {
        this.Display.setText ("");

        this.Operador = ' ';
        this.Primeiro = true;
    }
```

```
    private void TrateFechamentoDaJanela ()
    {
        System.exit (0);
    }

    public Calculadora ()
    {
        BorderLayout LayoutCalculadora = new BorderLayout ();

        this.setTitle ("Calculadora");
        this.setSize  (200, 200);
        this.setLayout (LayoutCalculadora);

        Panel      Botoes       = new Panel ();
        GridLayout LayoutBotoes = new GridLayout (4, 4);

        Botoes.setLayout (LayoutBotoes);

        Botoes.add (this.B7);
        Botoes.add (this.B8);
        Botoes.add (this.B9);
        Botoes.add (this.BA);
        Botoes.add (this.B4);
        Botoes.add (this.B5);
        Botoes.add (this.B6);
        Botoes.add (this.BS);
        Botoes.add (this.B1);
        Botoes.add (this.B2);
        Botoes.add (this.B3);
        Botoes.add (this.BM);
        Botoes.add (this.B0);
        Botoes.add (this.BI);
        Botoes.add (this.BL);
        Botoes.add (this.BD);

        this.add ("North",  this.Display);
        this.add ("Center", Botoes);

        this.B0.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('0');
                }
            }
        );

        this.B1.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('1');
                }
            }
        );

        this.B2.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('2');
                }
            }
```

```
        );

        this.B3.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('3');
                }
            }
        );

        this.B4.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('4');
                }
            }
        );

        this.B5.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('5');
                }
            }
        );

        this.B6.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('6');
                }
            }
        );

        this.B7.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('7');
                }
            }
        );

        this.B8.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('8');
                }
            }
        );

        this.B9.addMouseListener
```

```
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoNumerico ('9');
                }
            }
        );

        this.BA.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoDeOperacao ('+');
                }
            }
        );

        this.BS.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoDeOperacao ('-');
                }
            }
        );

        this.BM.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoDeOperacao ('*');
                }
            }
        );

        this.BD.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickEmBotaoDeOperacao ('/');
                }
            }
        );

        this.BI.addMouseListener
        (
            new MouseAdapter ()
            {
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickNoBotaoIgual ();
                }
            }
        );

        this.BL.addMouseListener
        (
            new MouseAdapter ()
            {
```

```
                public void mouseReleased (MouseEvent e)
                {
                    TrateClickNoBotaoDeLimpar ();
                }
            }
        );

        this.addWindowListener
        (
            new WindowAdapter ()
            {
                public void windowClosing (WindowEvent e)
                {
                    TrateFechamentoDaJanela ();
                }
            }
         );

        this.show ();
    }
}
```

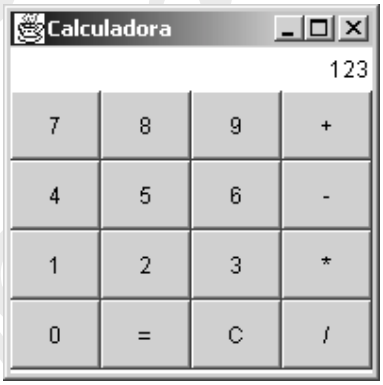
[C:\ExplsJava\Expl_08\TesteDeCalculadora.java]

```
class TesteDeCalculadora
{
    public static void main (String Args [])
    {
        Calculadora calculadora = new Calculadora();
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_08> javac TesteDeCalculadora.java
C:\ExplsJava\Expl_08> javaw TesteDeCalculadora
```

Isto fará com que se abra na tela a seguinte janela:



O usuário poderá interagir com ela de forma semelhante àquela que emprega quando usa uma calculadora simples.

Em vez de criar uma instância de MouseAdapter para cada um dos botões da calculadora, poder-se-ia criar apenas uma, com a capacidade de tratar eventos dirigidos a qualquer um dos botões.

[C:\ExplsJava\Expl_09\Calculadora.java]

```
import java.awt.*;
import java.awt.event.*;

class Calculadora extends Frame
{
    private Label  Display = new Label ("", Label.RIGHT);

    private Button B0      = new Button ("0"),
                   B1      = new Button ("1"),
                   B2      = new Button ("2"),
                   B3      = new Button ("3"),
                   B4      = new Button ("4"),
                   B5      = new Button ("5"),
                   B6      = new Button ("6"),
                   B7      = new Button ("7"),
                   B8      = new Button ("8"),
                   B9      = new Button ("9"),
                   BA      = new Button ("+"),
                   BS      = new Button ("-"),
                   BM      = new Button ("*"),
                   BD      = new Button ("/"),
                   BI      = new Button ("="),
                   BL      = new Button ("C");

    private boolean Primeiro = true;

    private double  Operando1;
    private char    Operador = ' ';

    private void TrateClickEmBotaoNumerico (char N)
    {
        if (this.Primeiro)
        {
            this.Display.setText ("" + N);
            this.Primeiro = false;
        }
        else
            this.Display.setText (this.Display.getText () + N);
    }

    private void TrateClickEmBotaoDeOperacao (char O)
    {
        if (!this.Display.getText ().equals (""))
        {
            this.TrateClickNoBotaoIgual ();

            this.Operando1 = new Double
                             (this.Display.getText ())
                             .doubleValue ();

            this.Operador = O;
```

```
        }
    }

    private void TrateClickNoBotaoIgual ()
    {
        if ((!(this.Display.getText ().equals (""))) && (this.Operador != '
'))
        {
            double Operando2 = new Double
                              (this.Display.getText ()).doubleValue (),

                   Resultado = 0;

            switch (this.Operador)
            {
                case '+':

                    Resultado = this.Operando1 + Operando2;
                    break;

                case '-':
                    Resultado = this.Operando1 - Operando2;
                    break;

                case '*':
                    Resultado = this.Operando1 * Operando2;
                    break;

                case '/':
                    Resultado = this.Operando1 / Operando2;
            }

            if (Math.round (Resultado) == Resultado)
                this.Display.setText ("" + Math.round (Resultado));
            else
                this.Display.setText ("" + Resultado);

            this.Operador = ' ';
        }

        this.Primeiro = true;
    }

    private void TrateClickNoBotaoDeLimpar ()
    {
        this.Display.setText ("");

        this.Operador = ' ';
        this.Primeiro = true;
    }

    private void TrateFechamentoDaJanela ()
    {
        System.exit (0);
    }

    public Calculadora ()
    {
        BorderLayout LayoutCalculadora = new BorderLayout ();

        this.setTitle ("Calculadora");
        this.setSize  (200, 200);
        this.setLayout (LayoutCalculadora);
```

```
        Panel      Botoes       = new Panel ();
        GridLayout LayoutBotoes = new GridLayout (4, 4);

        Botoes.setLayout (LayoutBotoes);

        Botoes.add (this.B7);
        Botoes.add (this.B8);
        Botoes.add (this.B9);
        Botoes.add (this.BA);
        Botoes.add (this.B4);
        Botoes.add (this.B5);
        Botoes.add (this.B6);
        Botoes.add (this.BS);
        Botoes.add (this.B1);
        Botoes.add (this.B2);
        Botoes.add (this.B3);
        Botoes.add (this.BM);
        Botoes.add (this.B0);
        Botoes.add (this.BI);
        Botoes.add (this.BL);
        Botoes.add (this.BD);

        this.add ("North",  this.Display);
        this.add ("Center", Botoes);

        MouseAdapter TratadorDeClicks;
        TratadorDeClicks = new MouseAdapter ()
        {
            public void mouseReleased (MouseEvent e)
            {
                if      (e.getComponent() == B0)
                    TrateClickEmBotaoNumerico ('0');
                else if (e.getComponent() == B1)
                    TrateClickEmBotaoNumerico ('1');
                else if (e.getComponent() == B2)
                    TrateClickEmBotaoNumerico ('2');
                else if (e.getComponent() == B3)
                    TrateClickEmBotaoNumerico ('3');
                else if (e.getComponent() == B4)
                    TrateClickEmBotaoNumerico ('4');
                else if (e.getComponent() == B5)
                    TrateClickEmBotaoNumerico ('5');
                else if (e.getComponent() == B6)
                    TrateClickEmBotaoNumerico ('6');
                else if (e.getComponent() == B7)
                    TrateClickEmBotaoNumerico ('7');
                else if (e.getComponent() == B8)
                    TrateClickEmBotaoNumerico ('8');
                else if (e.getComponent() == B9)
                    TrateClickEmBotaoNumerico ('9');

                else if (e.getComponent() == BA)
                    TrateClickEmBotaoDeOperacao ('+');
                else if (e.getComponent() == BS)
                    TrateClickEmBotaoDeOperacao ('-');
                else if (e.getComponent() == BM)
                    TrateClickEmBotaoDeOperacao ('*');
                else if (e.getComponent() == BD)
                    TrateClickEmBotaoDeOperacao ('/');

                else if (e.getComponent() == BI)
                    TrateClickNoBotaoIgual ();
```

```
                else if (e.getComponent() == BL)
                    TrateClickNoBotaoDeLimpar ();
            }
        };

        this.B0.addMouseListener (TratadorDeClicks);
        this.B1.addMouseListener (TratadorDeClicks);
        this.B2.addMouseListener (TratadorDeClicks);
        this.B3.addMouseListener (TratadorDeClicks);
        this.B4.addMouseListener (TratadorDeClicks);
        this.B5.addMouseListener (TratadorDeClicks);
        this.B6.addMouseListener (TratadorDeClicks);
        this.B7.addMouseListener (TratadorDeClicks);
        this.B8.addMouseListener (TratadorDeClicks);
        this.B9.addMouseListener (TratadorDeClicks);

        this.BA.addMouseListener (TratadorDeClicks);
        this.BS.addMouseListener (TratadorDeClicks);
        this.BM.addMouseListener (TratadorDeClicks);
        this.BD.addMouseListener (TratadorDeClicks);

        this.BI.addMouseListener (TratadorDeClicks);
        this.BL.addMouseListener (TratadorDeClicks);

        this.addWindowListener
        (
            new WindowAdapter ()
            {
                public void windowClosing (WindowEvent e)
                {
                    System.exit (0);
                }
            }
         );

        this.show ();
    }
}
```

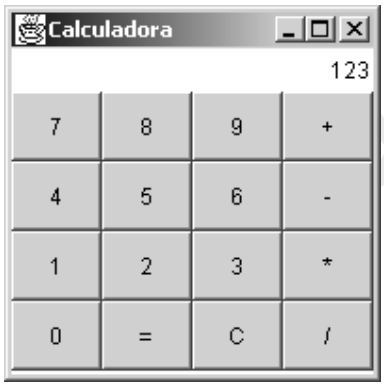
[C:\ExplsJava\Expl_09\TesteDeCalculadora.java]

```
class TesteDeCalculadora
{
   public static void main (String Args [])
   {
      Calculadora calculadora = new Calculadora();
   }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_09> javac TesteDeCalculadora.java
C:\ExplsJava\Expl_09> javaw TesteDeCalculadora
```

Isto fará com que se abra na tela a seguinte janela:



O usuário poderá interagir com ela de forma semelhante àquela que emprega quando usa uma calculadora simples.

Em vez de empregar instâncias de uma classe anônima derivada de um Adaper para o tratamento de eventos, poder-se-ia empregar instâncias de uma classe nomeada interna derivada desta mesma família de classes.

[C:\ExplsJava\Expl_10\Calculadora.java]

```
import java.awt.*;
import java.awt.event.*;

class Calculadora extends Frame
{
    private Label  Display = new Label ("", Label.RIGHT);

    private Button B0      = new Button ("0"),
                   B1      = new Button ("1"),
                   B2      = new Button ("2"),
                   B3      = new Button ("3"),
                   B4      = new Button ("4"),
                   B5      = new Button ("5"),
                   B6      = new Button ("6"),
                   B7      = new Button ("7"),
                   B8      = new Button ("8"),
                   B9      = new Button ("9"),
                   BA      = new Button ("+"),
                   BS      = new Button ("-"),
                   BM      = new Button ("*"),
                   BD      = new Button ("/"),
                   BI      = new Button ("="),
                   BL      = new Button ("C");

    private boolean Primeiro = true;

    private double Operando1;
    private char   Operador = ' ';

    protected class TratadorDeClicks extends MouseAdapter
    {
```

```
        private void TrateClickEmBotaoNumerico (char N)
        {
            if (Primeiro)
            {
                Display.setText ("" + N);
                Primeiro = false;
            }
            else
                Display.setText (Display.getText () + N);
        }

        private void TrateClickEmBotaoDeOperacao (char O)
        {
            if (!Display.getText ().equals (""))
            {
                this.TrateClickNoBotaoIgual ();

                Operando1 = new Double
                          (Display.getText ())
                          .doubleValue ();

                Operador  = O;
            }
        }

        private void TrateClickNoBotaoIgual ()
        {
            if ((!(Display.getText ().equals (""))) && (Operador != ' '))
            {
                double Operando2 = new Double
                                 (Display.getText ()).doubleValue (),

                       Resultado = 0;

                switch (Operador)
                {
                    case '+':
                        Resultado = Operando1 + Operando2;
                        break;

                    case '-':
                        Resultado = Operando1 - Operando2;
                        break;

                    case '*':
                        Resultado = Operando1 * Operando2;
                        break;

                    case '/':
                        Resultado = Operando1 / Operando2;
                }

                if (Math.round (Resultado) == Resultado)
                    Display.setText ("" + Math.round (Resultado));
                else
                    Display.setText ("" + Resultado);

                Operador = ' ';
            }

            Primeiro = true;
        }
```

```
        private void TrateClickNoBotaoDeLimpar ()
        {
            Display.setText ("");

            Operador = ' ';
            Primeiro = true;
        }

        public void mouseReleased (MouseEvent e)
        {
            if      (e.getComponent() == B0)
                this.TrateClickEmBotaoNumerico ('0');
            else if (e.getComponent() == B1)
                this.TrateClickEmBotaoNumerico ('1');
            else if (e.getComponent() == B2)
                this.TrateClickEmBotaoNumerico ('2');
            else if (e.getComponent() == B3)
                this.TrateClickEmBotaoNumerico ('3');
            else if (e.getComponent() == B4)
                this.TrateClickEmBotaoNumerico ('4');
            else if (e.getComponent() == B5)
                this.TrateClickEmBotaoNumerico ('5');
            else if (e.getComponent() == B6)
                this.TrateClickEmBotaoNumerico ('6');
            else if (e.getComponent() == B7)
                this.TrateClickEmBotaoNumerico ('7');
            else if (e.getComponent() == B8)
                this.TrateClickEmBotaoNumerico ('8');
            else if (e.getComponent() == B9)
                this.TrateClickEmBotaoNumerico ('9');

            else if (e.getComponent() == BA)
                this.TrateClickEmBotaoDeOperacao ('+');
            else if (e.getComponent() == BS)
                this.TrateClickEmBotaoDeOperacao ('-');
            else if (e.getComponent() == BM)
                this.TrateClickEmBotaoDeOperacao ('*');
            else if (e.getComponent() == BD)
                this.TrateClickEmBotaoDeOperacao ('/');

            else if (e.getComponent() == BI)
                this.TrateClickNoBotaoIgual ();
            else if (e.getComponent() == BL)
                this.TrateClickNoBotaoDeLimpar ();
        }
    }

    protected class TratadorDeFechamento extends WindowAdapter
    {
        public void windowClosing (WindowEvent e)
        {
            System.exit (0);
        }
    }

    public Calculadora ()
    {
        BorderLayout LayoutCalculadora = new BorderLayout ();

        this.setTitle ("Calculadora");
        this.setSize  (200, 200);
        this.setLayout (LayoutCalculadora);
```

```
        Panel      Botoes       = new Panel ();
        GridLayout LayoutBotoes = new GridLayout (4, 4);

        Botoes.setLayout (LayoutBotoes);

        Botoes.add (this.B7);
        Botoes.add (this.B8);
        Botoes.add (this.B9);
        Botoes.add (this.BA);
        Botoes.add (this.B4);
        Botoes.add (this.B5);
        Botoes.add (this.B6);
        Botoes.add (this.BS);
        Botoes.add (this.B1);
        Botoes.add (this.B2);
        Botoes.add (this.B3);
        Botoes.add (this.BM);
        Botoes.add (this.B0);
        Botoes.add (this.BI);
        Botoes.add (this.BL);
        Botoes.add (this.BD);

        this.add ("North",  this.Display);
        this.add ("Center", Botoes);

        TratadorDeClicks TC_Calculadora;
        TC_Calculadora = new TratadorDeClicks ();

        this.B0.addMouseListener (TC_Calculadora);
        this.B1.addMouseListener (TC_Calculadora);
        this.B2.addMouseListener (TC_Calculadora);
        this.B3.addMouseListener (TC_Calculadora);
        this.B4.addMouseListener (TC_Calculadora);
        this.B5.addMouseListener (TC_Calculadora);
        this.B6.addMouseListener (TC_Calculadora);
        this.B7.addMouseListener (TC_Calculadora);
        this.B8.addMouseListener (TC_Calculadora);
        this.B9.addMouseListener (TC_Calculadora);

        this.BA.addMouseListener (TC_Calculadora);
        this.BS.addMouseListener (TC_Calculadora);
        this.BM.addMouseListener (TC_Calculadora);
        this.BD.addMouseListener (TC_Calculadora);

        this.BI.addMouseListener (TC_Calculadora);
        this.BL.addMouseListener (TC_Calculadora);

        TratadorDeFechamento TF_Calculadora;
        TF_Calculadora = new TratadorDeFechamento ();

        this.addWindowListener (TF_Calculadora);

        this.show ();
    }
}
```

[C:\ExplsJava\Expl_10\TesteDeCalculadora.java]

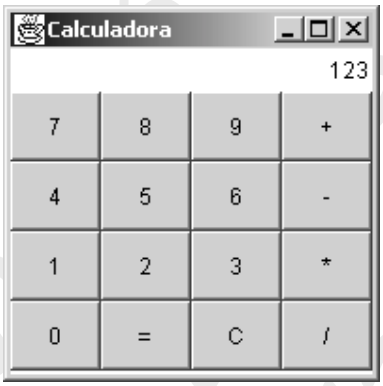
```
class TesteDeCalculadora
{
```

```
    public static void main (String Args [])
    {
        Calculadora calculadora = new Calculadora();
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_10> javac TesteDeCalculadora.java
C:\ExplsJava\Expl_10> javaw TesteDeCalculadora
```

Isto fará com que se abra na tela a seguinte janela:



O usuário poderá interagir com ela de forma semelhante àquela que emprega quando usa uma calculadora simples.

Mais sobre Detectores de Evento

Não é obrigatório que se faça uso de classes anônimas para detectar eventos. Se desejarmos, podemos fazer nossas janelas (classes que derivam da classe Frame) se comportarem como detectoras de eventos.

Para tanto, tudo o que precisamos fazer é fazê-las implementar interfaces dos detectores de evento desejados e registrá-las como detectoras de evento de seus componentes de devem ser sensíveis a eventos. Veja abaixo quais são essas interfaces:

1. **ActionListener:**

   - **void actionPerformed (ActionEvent e):**

     É invocado sempre que o usuário age sobre um componente;

2. **AdjustmentListener:**

- **void adjustmentValueChanged (AdjustmentEvent e):**
  É invocado sempre que o componente ajustável (ScrollBar's, por exemplo) muda;

3. **AWTEventListener:**

- **void eventDispatched (AWTEvent e):**
  É invocado sempre que um evento é despachado no AWT;

4. **ComponentListener:**

- **void componentHidden (ComponentEvent e):**
  É invocado sempre que o componente torna-se invisível;
- **void componentMoved (ComponentEvent e):**
  É invocado sempre que a posição do componente muda;
- **void componentResized (ComponentEvent e):**
  É invocado sempre que o componente muda de tamanho;
- **void componentShown (ComponentEvent e):**
  É invocado sempre que o componete torna-se visível;

5. **ContainerListener:**

- **void componentAdded (ContainerEvent e):**
  É invocado sempre que o componente é adicionado a um container;
- **void componentRemoved (ContainerEvent e):**
  É invocado sempre que o componente é removido de um container.

6. **DragGestureListener:**

- **void dragGestureRecognized (DragGestureEvent e):**
  É invocado sempre que é detectado o ato de arrastar o mouse com um botão pressionado;

7. **DragSourceListener:**

- **void dragDropEnd (DragSourceDropEvent e):**
  É invocado sempre que uma operação de arrastar-largar se completa;

- **void dragEnter (DragSourceDragEvent e):**
  É invocado sempre que, durante uma operação de arrastar-largar, o cursor do mouse entra sobre um componente;

- **void dragExit (DragSourceEvent e):**
  É invocado sempre que, durante uma operação de arrastar-largar, o cursor do mouse sai de cima de um componente;

- **void dragOver (DragSourceDragEvent e):**
  É invocado sempre que, durante uma operação de arrastar-largar, o cursor do mouse se move sobre um componente;

- **void dropActionChanged (DragSourceDragEvent e):**
  É invocado sempre que, durante uma operação de arrastar-largar, o usuário executa uma ação paralela (tipo apertar uma tecla ou outro botão do mouse);

8. DropTargetListener:

- **void dragEnter (DropTargetDragEvent e):**
  É invocado sempre que, durante uma operação de arrastar-largar, o cursor do mouse entra sobre um DropTarget;

- **void dragExit (DropTargetEvent e):**
  É invocado sempre que, durante uma operação de arrastar-largar, o cursor do mouse sai de cima de um DropTarget;

- **void dragOver (DropTargetDragEvent e):**
  É invocado sempre que, durante uma operação de arrastar-largar, o cursor do mouse se move sobre um DropTarget;

- **void drop (DropSourceDropEvent e):**
  É invocado sempre que, durante uma operação de arrastar-largar, estando o cursor so mouse sobre um DropTarget, o usuário termina a operação (larga);

- **void dropActionChanged (DropTargetDragEvent e):**
  É invocado sempre que, durante uma operação de arrastar-largar, o usuário executa

uma ação paralela (tipo apertar uma tecla ou outro botão do mouse) sobre um DropTarget;

9. FocusListener:

- **void focusGained (FocusEvent e):**
  É invocado sempre que o componente ganha o foco;
- **void focusLost (FocusEvent e):**
  É invocado sempre que o componente perde o foco;

10. InputMethodListener:

- **void caretPositionChanged (InputMethodEvent e):**
  É invocado sempre que o caret (cursor em componente que aceita digitação) muda de posição dentro de um componente;
- **void inputMethodTextChanged (InputMethodEvent e):**
  É invocado sempre que o texto de um componente que aceita digitação muda;

11. ItemListener:

- **void itemStateChanged (ItemEvent e):**
  É invocado sempre que um item de um componente de seleção é selecionado ou passa ao estado de não selecionado;

12. KeyListener:

- **void keyPressed (KeyEvent e):**
  É invocado sempre que uma tecla foi pressionada;
- **void keyReleased (KeyEvent e):**
  É invocado sempre que uma tecla foi liberada;
- **void keyTyped (KeyEvent e):**
  É invocada sempre que uma tecla foi digitada;

13. MouseListener:

- **void mouseClicked (MouseEvent e):**
  É invocado sempre que o usuário clica com o mouse sobre o componente;

- **void mouseEntered (MouseEvent e):**
  É invocado sempre que o usuário leva o cursor do mouse sobre o componente;

- **void mouseExited (MouseEvent e):**
  É invocado sempre que o usuário retira o cursor do mouse de sobre o componente;

- **void mousePressed (MouseEvent e):**
  É invocado sempre que o usuário pressiona o botão do mouse sobre o componente;

- **void mouseReleased (MouseEvent e):**
  É invocado sempre que o usuário libera o botão do mouse que mantinha pressionado sobre o componente;

14. MouseInputListener:

- **void mouseClicked (MouseEvent e):**
  É invocado sempre que o usuário clica com o mouse sobre o componente;

- **void mouseDragged (MouseEvent e):**
  É invocado sempre que um botão do mouse é pressionado sobre o componente e então o mouse é arrastado;

- **void mouseEntered (MouseEvent e):**
  É invocado sempre que o usuário leva o cursor do mouse sobre o componente;

- **void mouseExited (MouseEvent e):**
  É invocado sempre que o usuário retira o cursor do mouse de sobre o componente;

- **void mouseMoved (MouseEvent e):**
  É invocado sempre que o mouse se move por sobre o componente (sem nenhum botão pressionado);

- **void mousePressed (MouseEvent e):**
  É invocado sempre que o usuário pressiona o botão do mouse sobre o componente;

- **void mouseReleased (MouseEvent e):**
  É invocado sempre que o usuário libera o botão do mouse que mantinha pressionado sobre o componente;

15. MouseMotionListener:

- **void mouseMoved (MouseEvent e):**
  É invocado sempre que o mouse se move por sobre o componente (sem nenhum botão pressionado);
- **void mouseDragged (MouseEvent e):**
  É invocado sempre que um botão do mouse é pressionado sobre o componente e então o mouse é arrastado;

16. TextListener:

- **void textValueChanged (textEvent e):**
  É invocado sempre que o valor de um texto muda;

17. WindowListener:

- **void windowActivated (windowEvent e):**
  É invocado sempre que a janela é ativada;
- **void windowClosed (windowEvent e):**
  É invocado sempre que a janela é fechada;
- **void windowClosing (windowEvent e):**
  É invocado sempre que é solicitado o fechamento da janela;
- **void windowDeactivated (windowEvent e):**
  É invocado sempre que a janela é desativada;
- **void windowDeiconified (windowEvent e):**
  É invocado sempre que a janela é desiconificada;
- **void windowIconified (windowEvent e):**
  É invocado sempre que a janela é iconificada;

- **void windowOpened (windowEvent e):**

  É invocado sempre que a janela é aberta.

[C:\ExplsJava\Expl_11\Calculadora.java]

```
import java.awt.*;
import java.awt.event.*;

class Calculadora extends    Frame
                  implements MouseListener,
                             WindowListener
{
    private Label  Display = new Label ("", Label.RIGHT);

    private Button B0      = new Button ("0"),
                   B1      = new Button ("1"),
                   B2      = new Button ("2"),
                   B3      = new Button ("3"),
                   B4      = new Button ("4"),
                   B5      = new Button ("5"),
                   B6      = new Button ("6"),
                   B7      = new Button ("7"),
                   B8      = new Button ("8"),
                   B9      = new Button ("9"),
                   BA      = new Button ("+"),
                   BS      = new Button ("-"),
                   BM      = new Button ("*"),
                   BD      = new Button ("/"),
                   BI      = new Button ("="),
                   BL      = new Button ("C");

    private boolean Primeiro = true;

    private double  Operando1;
    private char    Operador = ' ';

    private void TrateClickEmBotaoNumerico (char N)
    {
        if (this.Primeiro)
        {
            this.Display.setText ("" + N);
            this.Primeiro = false;
        }
        else
            this.Display.setText (this.Display.getText () + N);
    }

    private void TrateClickEmBotaoDeOperacao (char O)
    {
        if (!this.Display.getText ().equals (""))
        {
            this.TrateClickNoBotaoIgual ();

            this.Operando1 = new Double
                             (this.Display.getText ())
                             .doubleValue ();

            this.Operador = O;
```

```
        }
    }

    private void TrateClickNoBotaoIgual ()
    {
        if ((!(this.Display.getText ().equals (""))) && (this.Operador != '
'))
        {
            double Operando2 = new Double
                              (this.Display.getText ()).doubleValue (),

                   Resultado = 0;

            switch (this.Operador)
            {
                case '+':
                    Resultado = this.Operando1 + Operando2;
                    break;

                case '-':
                    Resultado = this.Operando1 - Operando2;
                    break;

                case '*':
                    Resultado = this.Operando1 * Operando2;
                    break;

                case '/':
                    Resultado = this.Operando1 / Operando2;
            }

            if (Math.round (Resultado) == Resultado)
                this.Display.setText ("" + Math.round (Resultado));
            else
                this.Display.setText ("" + Resultado);

            this.Operador = ' ';
        }

        this.Primeiro = true;
    }

    private void TrateClickNoBotaoDeLimpar ()
    {
        this.Display.setText ("");

        this.Operador = ' ';
        this.Primeiro = true;
    }

    public Calculadora ()
    {
        BorderLayout LayoutCalculadora = new BorderLayout ();

        this.setTitle ("Calculadora");
        this.setSize  (200, 200);
        this.setLayout (LayoutCalculadora);

        Panel     Botoes       = new Panel ();
        GridLayout LayoutBotoes = new GridLayout (4, 4);

        Botoes.setLayout (LayoutBotoes);
```

```
        Botoes.add (this.B7);
        Botoes.add (this.B8);
        Botoes.add (this.B9);
        Botoes.add (this.BA);
        Botoes.add (this.B4);
        Botoes.add (this.B5);
        Botoes.add (this.B6);
        Botoes.add (this.BS);
        Botoes.add (this.B1);
        Botoes.add (this.B2);
        Botoes.add (this.B3);
        Botoes.add (this.BM);
        Botoes.add (this.B0);
        Botoes.add (this.BI);
        Botoes.add (this.BL);
        Botoes.add (this.BD);

        this.add ("North",  this.Display);
        this.add ("Center", Botoes);

        this.B0.addMouseListener (this);
        this.B1.addMouseListener (this);
        this.B2.addMouseListener (this);
        this.B3.addMouseListener (this);
        this.B4.addMouseListener (this);
        this.B5.addMouseListener (this);
        this.B6.addMouseListener (this);
        this.B7.addMouseListener (this);
        this.B8.addMouseListener (this);
        this.B9.addMouseListener (this);

        this.BA.addMouseListener (this);
        this.BS.addMouseListener (this);
        this.BM.addMouseListener (this);
        this.BD.addMouseListener (this);

        this.BI.addMouseListener (this);
        this.BL.addMouseListener (this);

        this.addWindowListener (this);

        this.show ();
    }

    public void mouseReleased (MouseEvent e)
    {
        if      (e.getComponent() == this.B0)
            this.TrateClickEmBotaoNumerico ('0');
        else if (e.getComponent() == this.B1)
            this.TrateClickEmBotaoNumerico ('1');
        else if (e.getComponent() == this.B2)
            this.TrateClickEmBotaoNumerico ('2');
        else if (e.getComponent() == this.B3)
            this.TrateClickEmBotaoNumerico ('3');
        else if (e.getComponent() == this.B4)
            this.TrateClickEmBotaoNumerico ('4');
        else if (e.getComponent() == this.B5)
            this.TrateClickEmBotaoNumerico ('5');
        else if (e.getComponent() == this.B6)
            this.TrateClickEmBotaoNumerico ('6');
        else if (e.getComponent() == this.B7)
            this.TrateClickEmBotaoNumerico ('7');
        else if (e.getComponent() == this.B8)
```

```
            this.TrateClickEmBotaoNumerico ('8');
        else if (e.getComponent() == this.B9)
            this.TrateClickEmBotaoNumerico ('9');

        else if (e.getComponent() == this.BA)
            this.TrateClickEmBotaoDeOperacao ('+');
        else if (e.getComponent() == this.BS)
            this.TrateClickEmBotaoDeOperacao ('-');
        else if (e.getComponent() == this.BM)
            this.TrateClickEmBotaoDeOperacao ('*');
        else if (e.getComponent() == this.BD)
            this.TrateClickEmBotaoDeOperacao ('/');

        else if (e.getComponent() == this.BI)
            this.TrateClickNoBotaoIgual ();
        else if (e.getComponent() == this.BL)
            this.TrateClickNoBotaoDeLimpar ();
    }

    public void mousePressed (MouseEvent e)
    {}

    public void mouseClicked (MouseEvent e)
    {}

    public void mouseEntered (MouseEvent e)
    {}

    public void mouseExited (MouseEvent e)
    {}

    public void windowClosing (WindowEvent e)
    {
        System.exit (0);
    }

    public void windowOpened (WindowEvent e)
    {}

    public void windowClosed (WindowEvent e)
    {}

    public void windowActivated (WindowEvent e)
    {}

    public void windowDeactivated (WindowEvent e)
    {}

    public void windowIconified (WindowEvent e)
    {}

    public void windowDeiconified (WindowEvent e)
    {}
}
```

[C:\ExplsJava\Expl_11\TesteDeCalculadora.java]

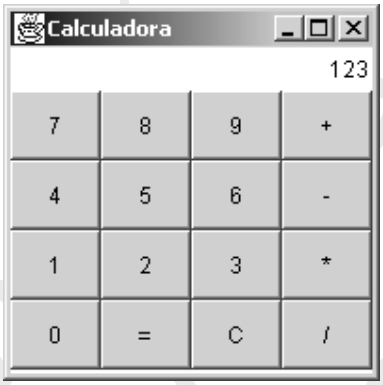
```
class TesteDeCalculadora
{
    public static void main (String Args [])
```

```
    {
        Calculadora calculadora = new Calculadora();
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_11> javac TesteDeCalculadora.java
C:\ExplsJava\Expl_11> javaw TesteDeCalculadora
```

Isto fará com que se abra na tela a seguinte janela:



O usuário poderá interagir com ela de forma semelhante àquela que emprega quando usa uma calculadora simples.

Lidando com o Foco

Quando o usuário clica em um componente de uma GUI, o ítem clicado se torna, em um certo sentido, selecionado, ou, como costumamos dizer, tem o foco.

É possível ainda que se deseje fazer que um componente ganhe o foco via programação (não em virtude de um click do usuário).

Sendo C um componente, veja abaixo a forma geral do método que transfere o foco para C:

```
C.requestFocus ();
```

Despacho de Eventos

Ocasionalmente pode ser útil que um programa gere seus próprios eventos. Embora possa parecer estranho, isto torna o projeto de um programa muito mais simples, já que evita a replicação de código.

Veja abaixo a lista dos tipos de eventos, acompanhados da lista de seus campos, de seus construtores e de seus métodos:

1. **ComponentEvent:**

   - **Campos:**

     - **static int COMPONENT_FIRST:**
       Número do primeiro identificador de evento de componente;

     - **static int COMPONENT_HIDDEN:**
       Representa o evento de ocultação de componente;

     - **static int COMPONENT_LAST:**
       Número do último identificador de evento de componente;

     - **static int COMPONENT_MOVED:**
       Representa o evento de movimentação de componente;

     - **static int COMPONENT_RESIZED:**
       Representa o evento de redimensionamento de componente;

     - **static int COMPONENT_SHOWN:**
       Representa o evento de exibição de componente;

   - **Construtores:**

     - **ComponentEvent (Component c, int i):**
       Constroi um ComponentEvent identificado pelo identificador i (um dos campos acima) dirigido ao componente c;

   - **Métodos:**

     - **void consume ():**
       Consome o evento de forma a impedir seu processamento;

     - **int getID ():**
       Retorna a identificação do evento (um dos campos acima);

- **Component getComponent ():**

  Retorna o componente ao qual se destina o evento;

- **boolean isConsumed ():**

  Retorna true se o evento foi consumido, e false, caso contrário;

2. ContainerEvent:

- **Campos:**

  - **static int COMPONENT_ADDED:**

    Representa o evento de adição de componente a um container;

  - **static int COMPONENT_REMOVED:**

    Representa o evento de remoção de componente de um container;

  - **static int CONTAINER_FIRST:**

    Número do primeiro identificador de evento de container;

  - **static int CONTAINER_LAST:**

    Número do último identificador de evento de container;

- **Construtores:**

  - **ContainerEvent (Component c, int i, Component a):**

    Constroi um ContainerEvent identificado pelo identificador i (um dos campos acima) dirigido ao componente c comunicando a adição do componente a;

- **Métodos:**

  - **void consume ():**

    Consome o evento de forma a impedir seu processamento;

  - **Component getChild ():**

    Retorna o container ao qual se destina o evento;

  - **Container getContainer ():**

    Retorna o componente adicionado/removido no/do container;

- **int getID ():**

  Retorna a identificação do evento (um dos campos acima);

- **boolean isConsumed ():**

  Retorna true se o evento foi consumido, e false, caso contrário;

3. **FocusEvent:**

- **Campos:**

  - **static int FOCUS_FIRST:**

    Número do primeiro identificador de evento de foco;

  - **static int FOCUS_GAINED:**

    Representa o evento de ganho de foco;

  - **static int FOCUS_LAST:**

    Número do último identificador de evento de foco;

  - **static int FOCUS_LOST:**

    Representa o evento de perda de foco;

- **Construtores:**

  - **FocusEvent (Component c, int i):**

    Constroi um FocusEvent identificado pelo identificador i (um dos campos acima) dirigido ao componente c para representar o ganho/perda definitiva de foco;

  - **FocusEvent (Component c, int i, boolean t):**

    Constroi um FocusEvent identificado pelo identificador i (um dos campos acima) dirigido ao componente c para representar o ganho/perda de foco; tal ganho/perda é temporária, caso t valha true, ou é definitiva, caso contrário;

- **Métodos:**

  - **void consume ():**

    Consome o evento de forma a impedir seu processamento;

- **Component getComponent ():**
  Retorna o componente ao qual se destina o evento;
- **int getID ():**
  Retorna a identificação do evento (um dos campos acima);
- **boolean isConsumed ():**
  Retorna true se o evento foi consumido, e false, caso contrário;
- **boolean isTemporary ():**
  Retorna true, caso o evento se refira a uma mudança temporária de foco, ou false, caso contrário;

4. KeyEvent:

- **Campos:**
  - **static int KEY_FIRST:**
    Número do primeiro identificador de evento de foco;
  - **static int KEY_LAST:**
    Número do último identificador de evento de foco;
  - **static int KEY_PRESSED:**
    Representa o evento de pressionamento de tecla;
  - **static int KEY_RELEASED:**
    Representa o evento de liberação de tecla;
  - **static int KEY_TYPED:**
    Representa o evento de digitação de tecla;
- **Construtores:**
  - **KeyEvent (Component c, int i, long t, int m, int codt):**
    Constroi um KeyEvent identificado pelo identificador i (um dos campos acima) dirigido ao componente c, gerado no instante de tempo t, relacionado à tecla de código codt, modificada pelo modificador m (shift, ctrl, etc);

- **KeyEvent (Component c, int i, long t, int m, int codt, char chart):**
  Constroi um KeyEvent identificado pelo identificador i (um dos campos acima) dirigido ao componente c, gerado no instante de tempo t, relacionado à tecla de código codt, modificada pelo modificador m (shift, ctrl, etc), resultando no caractere UNICODE chart;

- **Métodos:**

  - **void consume ():**
    Consome o evento de forma a impedir seu processamento;

  - **Component getComponent ():**
    Retorna o componente ao qual se destina o evento;

  - **int getID ():**
    Retorna a identificação do evento (um dos campos acima);

  - **char getKeyChar ():**
    Retorna o caractere UNICODE associado à tecla deste evento;

  - **int getKeyCode ():**
    Retorna o código do caractere UNICODE associado à tecla deste evento;

  - **static String getKeyModifiersText (int m):**
    Retorna um String que descreve a(s) tecla(s) de modificação representadas pelo modificador m (“Shift”, “Ctrl+Shift”, etc);

  - **static String getKeyText(int codt):**
    Retorna um String que descreve a tecla de código codt (“HOME”, “F1”, etc);

  - **int getModifiers ():**
    Retorna os modificadores deste evento;

  - **long getWhen ():**
    Retorna o timestamp do momento no qual ocorreu o evento;

- **boolean isActionKey ():**
  Retorna true se a tecla associada ao evento for uma tecla de ação, e false, caso contrário;

- **boolean isAltDown ():**
  Retorna true se a tecla ALT estiver pressionada por ocasião da ocorrência deste evento, e false, caso contrário;

- **boolean isAltGraphDown ():**
  Retorna true se a tecla ALT-GRAPH estiver pressionada por ocasião da ocorrência deste evento, e false, caso contrário;

- **boolean isConsumed ():**
  Retorna true se o evento foi consumido, e false, caso contrário;

- **boolean isControlDown ():**
  Retorna true se a tecla CTRL estiver pressionada por ocasião da ocorrência deste evento, e false, caso contrário;

- **boolean isMetaDown ():**
  Retorna true se a tecla META estiver pressionada por ocasião da ocorrência deste evento, e false, caso contrário;

- **boolean isShiftDown ():**
  Retorna true se a tecla SHIFT estiver pressionada por ocasião da ocorrência deste evento, e false, caso contrário;

- **void setKeyChar (char chart):**
  Ajusta o KeyEvent para indicar o caractere lógico chart;

- **void setKeyCode (int codt):**
  Ajusta o KeyEvent para indicar o caractere lógico de código UNICODE codt;

- **void setModifiers (int m):**
  Ajusta o KeyEvent para indicar o uso dos modificadores expressos por m;

5. MouseEvent:

- **Campos:**

  - **static int MOUSE_CLICKED:**
    Representa o evento de click de um botão do mouse;

  - **static int MOUSE_DRAGGED:**
    Representa o evento de movimentação do mouse com um de seus botões pressionado;

  - **static int MOUSE_ENTERED:**
    Representa o evento de posicionamento do mouse sobre um componente;

  - **static int MOUSE_EXITED:**
    Representa o evento de retirada do mouse de sobre um componente;

  - **static int MOUSE_FIRST:**
    Número do primeiro identificador de evento de mouse;

  - **static int MOUSE_LAST:**
    Número do último identificador de evento de foco;

  - **static int MOUSE_MOVED:**
    Representa o evento de movimentação do mouse sem o pressionamento de nenhum de seus botões;

  - **static int MOUSE_PRESSED:**
    Representa o evento de pressionamento de um dos botões do mouse;

  - **static int MOUSE_RELEASED:**
    Representa o evento de liberação do botão do mouse que se encontrava pressionado;

- **Construtores:**

  - **MouseEvent (Component c, int i, long t, int m, int x, int y, int cc, boolean apm):**
    Constroi um MouseEvent identificado pelo identificador i (um dos campos acima) dirigido ao componente c, gerado no instante de tempo t, modificado pelo

modificador m (shift, ctrl, etc), na posição (x,y), indicando cc clicks do mouse e com apm especificando se o MouseEvent é ou não um acionador de um popup menu;

- **Métodos:**

  - **void consume ():**
    Consome o evento de forma a impedir seu processamento;

  - **int getID ():**
    Retorna a identificação do evento (um dos campos acima);

  - **int getClickCount ():**
    Retorna a quantidade de clicks associada e este evento;

  - **Component getComponent ():**
    Retorna o componente ao qual se destina o evento;

  - **int getModifiers ():**
    Retorna os modificadores deste evento;

  - **Point getPoint ():**
    Retorna uma instância da classe Point representanto a posição (x,y) (relativa ao componente associado ao evento) na qual o MouseEvent ocorreu;

  - **long getWhen ():**
    Retorna o timestamp do momento no qual ocorreu o evento;

  - **int getX ():**
    Retorna a coordenada x da posição (relativa ao componente associado ao evento) na qual o MouseEvent ocorreu;

  - **int getY ():**
    Retorna a coordenada y da posição (relativa ao componente associado ao evento) na qual o MouseEvent ocorreu;

- **boolean isAltDown ():**
  Retorna true se a tecla ALT estiver pressionada por ocasião da ocorrência deste evento, e false, caso contrário;

- **boolean isAltGraphDown ():**
  Retorna true se a tecla ALT-GRAPH estiver pressionada por ocasião da ocorrência deste evento, e false, caso contrário;

- **boolean isConsumed ():**
  Retorna true se o evento foi consumido, e false, caso contrário;

- **boolean isControlDown ():**
  Retorna true se a tecla CTRL estiver pressionada por ocasião da ocorrência deste evento, e false, caso contrário;

- **boolean isMetaDown ():**
  Retorna true se a tecla META estiver pressionada por ocasião da ocorrência deste evento, e false, caso contrário;

- **boolean isShiftDown ():**
  Retorna true se a tecla SHIFT estiver pressionada por ocasião da ocorrência deste evento, e false, caso contrário;

- **boolean isPopupTrigger ():**
  Retorna true, se o MouseEvent é um evento acionador de um popup menu, e false, caso contrário;

- **void translatePoint (int x, int y):**
  Traduz as coordenadas do MouseEvent pela adição do deslocamento (offset) (x,y);

6. WindowEvent:

- **Campos:**

  - **static int WINDOW_ACTIVATED:**
    Representa o evento de ativação de janela;

- **static int WINDOW_CLOSED:**
  Representa o evento de fechamento de janela;
- **static int WINDOW_CLOSING:**
  Representa o evento de pedido de fechamento de janela;
- **static int WINDOW_DEACTIVATED:**
  Representa o evento de desativação de janela;
- **static int WINDOW_DEICONIFIED:**
  Representa o evento de desiconificação de janela;
- **static int WINDOW_FIRST:**
  Número do primeiro identificador de evento de janela;
- **static int WINDOW_ICONIFIED:**
  Representa o evento de desiconificação de janela;
- **static int WINDOW_LAST:**
  Número do último identificador de evento de janela;
- **static int WINDOW_OPENED:**
  Representa o evento de abertura de janela;

- **Construtores:**
  - **WindowEvent (Window w, int i):**
    Constroi um WindowEvent identificado pelo identificador i (um dos campos acima) dirigido à janela w;
- **Métodos:**
  - **void consume ():**
    Consome o evento de forma a impedir seu processamento;
  - **int getID ():**
    Retorna a identificação do evento (um dos campos acima);

- **Window getWindow ():**

  Retorna a janela à qual se destina o evento;

- **boolean isConsumed ():**

  Retorna true se o evento foi consumido, e false, caso contrário;

Sendo C um compontente AWT e E um AWTEvent, veja abaixo a forma geral do comando que despacha um evento:

C.deliverEvent (E);

[C:\ExplsJava\Expl_12\Calculadora.java]

```
import java.awt.*;
import java.awt.event.*;
import java.util.Date;

class Calculadora extends   Frame
                  implements WindowListener,
                             MouseListener,
                             KeyListener
{
    private Label  Display = new Label ("", Label.RIGHT);
    private Button Botao [];

    private boolean Primeiro = true;

    private double  Operando1;
    private char    Operador = ' ';

    private void TrateClickEmBotaoNumerico (char N)
    {
        if (this.Primeiro)
        {
            this.Display.setText ("" + N);
            this.Primeiro = false;
        }
        else
            this.Display.setText (this.Display.getText () + N);
    }

    private void TrateClickEmBotaoDeOperacao (char O)
    {
        if (!this.Display.getText ().equals (""))
        {
            this.TrateClickNoBotaoIgual ();

            this.Operando1 = new Double
                             (this.Display.getText ())
                             .doubleValue ();

            this.Operador = O;
        }
    }

    private void TrateClickNoBotaoIgual ()
    {
        if ((!(this.Display.getText ().equals (""))) && (this.Operador != ' '))
```

```
        {
            double Operando2 = new Double
                              (this.Display.getText ()).doubleValue (),

                   Resultado = 0;

            switch (this.Operador)
            {
                case '+':
                    Resultado = this.Operando1 + Operando2;
                    break;

                case '-':
                    Resultado = this.Operando1 - Operando2;
                    break;

                case '*':
                    Resultado = this.Operando1 * Operando2;
                    break;

                case '/':
                    Resultado = this.Operando1 / Operando2;
            }

            if (Math.round (Resultado) == Resultado)
                this.Display.setText ("" + Math.round (Resultado));
            else
                this.Display.setText ("" + Resultado);

            this.Operador = ' ';
        }

        this.Primeiro = true;
    }

    private void TrateClickNoBotaoDeLimpar ()
    {
        this.Display.setText ("");

        this.Operador = ' ';
        this.Primeiro = true;
    }

    public Calculadora ()
    {
        this.setTitle ("Calculadora");
        this.setSize  (200, 200);

        BorderLayout LayoutCalculadora = new BorderLayout ();
        this.setLayout (LayoutCalculadora);

        this.addWindowListener (this);

        Panel      Botoes       = new Panel ();
        GridLayout LayoutBotoes = new GridLayout (4, 4);

        Botoes.setLayout (LayoutBotoes);

        String Rotulo [] = {"7", "8", "9", "+",
                            "4", "5", "6", "-",
                            "1", "2", "3", "*",
                            "0", "=", "C", "/"};

        this.Botao = new Button [Rotulo.length];

        for (int I = 0; I < this.Botao.length; I++)
        {
            this.Botao [I] = new Button (Rotulo [I]);
```

```
            this.Botao [I].addMouseListener (this);
            this.Botao [I].addKeyListener   (this);

            Botoes.add (this.Botao [I]);
        }

        this.add ("North",  this.Display);
        this.add ("Center", Botoes);

        this.show ();
    }

    public void windowClosing (WindowEvent e)
    {
        System.exit (0);
    }

    public void windowOpened (WindowEvent e)
    {}

    public void windowClosed (WindowEvent e)
    {}

    public void windowActivated (WindowEvent e)
    {}

    public void windowDeactivated (WindowEvent e)
    {}

    public void windowIconified (WindowEvent e)
    {}

    public void windowDeiconified (WindowEvent e)
    {}

    public void mouseReleased (MouseEvent e)
    {
        char RotuloDoBotao = ((Button)e.getComponent()).getLabel().charAt(0);

        switch (RotuloDoBotao)
        {
            case '0':
            case '1':
            case '2':
            case '3':
            case '4':
            case '5':
            case '6':
            case '7':
            case '8':
            case '9': this.TrateClickEmBotaoNumerico (RotuloDoBotao);
                      break;

            case '+':
            case '-':
            case '*':
            case '/': TrateClickEmBotaoDeOperacao (RotuloDoBotao);
                      break;

            case '=': TrateClickNoBotaoIgual ();
                      break;


            case 'C': TrateClickNoBotaoDeLimpar ();
        }
    }

    public void mousePressed (MouseEvent e)
    {}
```

```
    public void mouseClicked (MouseEvent e)
    {}

    public void mouseEntered (MouseEvent e)
    {}

    public void mouseExited (MouseEvent e)
    {}

    public void keyTyped (KeyEvent e)
    {
        if ((int)e.getKeyChar () == KeyEvent.VK_ENTER)
            e.setKeyChar ('=');
        else
            if ((int)e.getKeyChar () == KeyEvent.VK_ESCAPE)
                e.setKeyChar ('C');

        e.setKeyChar (Character.toUpperCase (e.getKeyChar ()));

        Date D = new Date ();

        for (int I = 0; I < Botao.length; I++)
            if (e.getKeyChar () == this.Botao [I].getLabel ().charAt (0))
            {
                this.Botao [I].dispatchEvent (new MouseEvent (
                                              this.Botao [I],
                                              MouseEvent.MOUSE_Released,
                                              D.getTime (),
                                              0, 0, 0, 1, false));

                break;
            }
    }

    public void keyPressed (KeyEvent e)
    {}

    public void keyReleased (KeyEvent e)
    {}
}
```

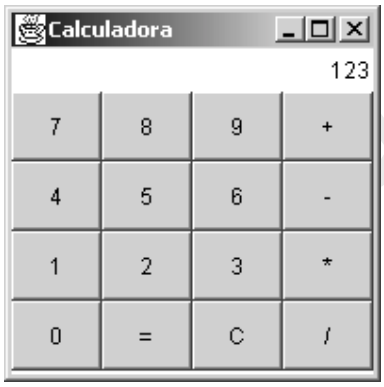
[C:\ExplsJava\Expl_12\TesteDeCalculadora.java]

```
class TesteDeCalculadora
{
    public static void main (String Args [])
    {
        Calculadora calculadora = new Calculadora();
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_12> javac TesteDeCalculadora.java
C:\ExplsJava\Expl_12> javaw TesteDeCalculadora
```

Isto fará com que se abra na tela a seguinte janela:



O usuário poderá interagir com ela de forma semelhante àquela que emprega quando usa uma calculadora simples.

Applets (O Pacote java.applet)

Uma das principais usos de Java é a criação de miniaplicações, ou applets. O pacote java.applet é uma biblioteca que contém diversas classes úteis para o desenvolvimento de applets.

java.applet.Applet

Applet's podem ser vistas como um pequeno programa que se concentra na solução de um problema bem específico. Elas devem ser pequenas, rápidas, facilmente transferíveis através de recursos de rede.

Applet's são programas Java compilados em ByteCodes que são executados em um browser WWW que ofereça suporte a Java. Por serem transferidas através da WWW e executadas na máquina do usuário, elas são, por natureza inseguras.

Em virtude disso, tanto a SUN quanto os fabricantes de browsers da WWW colocaram algumas restrições sobre a funcionalidade das applets:

- Applet's não têm acesso ao sistema de arquivos do usuário local nem para leitura nem para escrita ;
- Applet's não podem ter funções qualificadas com o modificador native, já que o código escrito em outras linguagens não pode ser verificado.

- Applet's não podem contactar nenhum servidor da rede exceto aquele de onde foram descarregadas.

Tecnicamente uma Applet é uma subclasse de java.awt.panel e, como tal, pode conter componentes do Abstract Window Toolkit e do Java Language Package, além de poderem entrar na composição de outras applets e até mesmo de aplicações completas.

O ciclo de vida de uma applets é composto por quatro fases diferentes, a saber: carga, ativação, parada, destruição. Cada uma dessas fases é ativada por um método apropriado.

Além das estruturas e do comportamento herdado da classe java.awt.Panel, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **Applet ():**
    Constroi uma instância da classe Applet;
- **Métodos:**
  - **void destroy ():**
    É chamada pelo browser para informar à Applet que ela está na iminência de ser destruida; deve implementar todo o comportamento necessário para desalocar quaisquer recursos que tenha alocado;
  - **AppletContext getAppletContext ():**
    Retorna o contexto da Applet;
  - **String getAppletInfo ():**
    Retorna informações a respeito desta Applet;
  - **AudioClip getAudioClip (URL U):**
    Retorna o AudioClip especificado pela URL U;
  - **AudioClip getAudioClip (URL U, String N):**
    Retorna o AudioClip especificado pela URL U e nome N;
  - **URL getCodeBase ():**
    Retorna a URL base;

- **URL getDocumentBase ():**

  Retorna a URL do documento;

- **Image getImage (URL U):**

  Retorna a Image especificada pela URL U;

- **Image getImage (URL U, String N):**

  Retorna a Image especificada pela URL U e nome N;

- **String getParameter (String N):**

  Retorna o parâmetro de nome N;

- **String[][] getParameterInfo ():**

  Retorna informações sobre os parâmetros;

- **void init ():**

  É chamada pelo browser para informar à Applet que ela acaba de ser carregada; deve implementar todo o comportamento necessário à sua iniciação;

- **boolean isActive ():**

  Verifica ser esta Applet está ativa;

- **static newAudioClip (URL U):**

  Retorna o AudioClip especificado pela URL U;

- **void play (URL U):**

  Reproduz o AudioClip especificado pela URL U;

- **void play (URL U, String N):**

  Reproduz o AudioClip especificado pela URL U e nome N;

- **void resize (Dimension D):**

  Redimensiona esta Applet com a dimensão dada;

- **void resize (int L, int A):**

  Redimensiona esta Applet com a largura L e altura A;

- **void showStatus (String E):**

  Faz com o estado E seja exibido na janela de exibição de estado;

- **void start ():**

  É chamada pelo browser para informar à Applet que ela deve rodar; deve implementar todo o comportamento necessário ao inicio de sua execução;

- **void stop ():**

  É chamada pelo browser para informar à Applet que ela deve parar de rodar; deve implementar todo o comportamento necessário para parar sua execução;

Para executar uma Applet, é preciso (1) os ByteCodes resultantes da compilação; e (2) um arquivo HTML contendo as informações básicas da applet e seus parâmetros.

Sendo Applet.class o nome do arquivo compilado da applet que deseja ativar, Local o local onde ela se encontra, Nome o nome da instância da applet, L e A, respectivamente, a largura e a altura com as quais deseja vê-la exibida, $Nome_i$ os nomes de seus parâmetros e $Valor_i$ os valores de seus parâmetros, temos que a estrutura básica de um arquivo HTML para ativar uma applet tem a seguinte estrutura básica:

```
<html>
        <applet codebase="Local" Code="Applet.class" Name="Nome" Width=L Heigth=A>
                <param name="Nome1" value="Valor1">
                <param name="Nome2" value="Valor2">
                …
                <param name="Nomen" value="Valorn">
        </applet>
</html>
```

[C:\ExplsJava\Expl_13\Calculadora.java]

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.util.Date;

public class Calculadora extends    Frame
                         implements MouseListener,
                                    KeyListener
{
    private Label  Display = new Label ("", Label.RIGHT);
    private Button Botao [];

    private boolean Primeiro = true;
```

```
    private double  Operando1;
    private char    Operador = ' ';

    private void TrateClickEmBotaoNumerico (char N)
    {
        if (this.Primeiro)
        {
            this.Display.setText ("" + N);
            this.Primeiro = false;
        }
        else
            this.Display.setText (this.Display.getText () + N);
    }

    private void TrateClickEmBotaoDeOperacao (char O)
    {
        if (!this.Display.getText ().equals (""))
        {
            this.TrateClickNoBotaoIgual ();

            this.Operando1 = new Double
                           (this.Display.getText ())
                           .doubleValue ();

            this.Operador = O;
        }
    }

    private void TrateClickNoBotaoIgual ()
    {
        if ((!(this.Display.getText ().equals (""))) && (this.Operador != ' '))
        {
            double Operando2 = new Double
                             (this.Display.getText ()).doubleValue (),

                   Resultado = 0;

            switch (this.Operador)
            {
                case '+':
                    Resultado = this.Operando1 + Operando2;
                    break;

                case '-':
                    Resultado = this.Operando1 - Operando2;
                    break;

                case '*':
                    Resultado = this.Operando1 * Operando2;
                    break;

                case '/':
                    Resultado = this.Operando1 / Operando2;
            }

            if (Math.round (Resultado) == Resultado)
                this.Display.setText ("" + Math.round (Resultado));
            else
                this.Display.setText ("" + Resultado);

            this.Operador = ' ';
        }

        this.Primeiro = true;
    }

    private void TrateClickNoBotaoDeLimpar ()
    {
```

```
        this.Display.setText ("");

        this.Operador = ' ';
        this.Primeiro = true;
    }

    public void start ()
    {
        BorderLayout LayoutCalculadora = new BorderLayout ();
        this.setLayout (LayoutCalculadora);

        Panel      Botoes       = new Panel ();
        GridLayout LayoutBotoes = new GridLayout (4, 4);

        Botoes.setLayout (LayoutBotoes);

        String Rotulo [] = {"7", "8", "9", "+",
                            "4", "5", "6", "-",
                            "1", "2", "3", "*",
                            "0", "=", "C", "/"};

        this.Botao = new Button [Rotulo.length];

        for (int I = 0; I < this.Botao.length; I++)
        {
            this.Botao [I] = new Button (Rotulo [I]);

            this.Botao [I].addMouseListener (this);

            this.Botao [I].addKeyListener   (this);

            Botoes.add (this.Botao [I]);
        }

        this.add ("North",  this.Display);
        this.add ("Center", Botoes);
    }

    public void mouseReleased (MouseEvent e)
    {
        char RotuloDoBotao = ((Button)e.getComponent()).getLabel().charAt(0);

        switch (RotuloDoBotao)
        {
            case '0':
            case '1':
            case '2':
            case '3':
            case '4':
            case '5':
            case '6':
            case '7':
            case '8':
            case '9': this.TrateClickEmBotaoNumerico (RotuloDoBotao);
                      break;

            case '+':
            case '-':
            case '*':
            case '/': TrateClickEmBotaoDeOperacao (RotuloDoBotao);
                      break;

            case '=': TrateClickNoBotaoIgual ();
                      break;

            case 'C': TrateClickNoBotaoDeLimpar ();
        }
    }
```

```
    public void mousePressed (MouseEvent e)
    {}

    public void mouseClicked (MouseEvent e)
    {}

    public void mouseEntered (MouseEvent e)
    {}

    public void mouseExited (MouseEvent e)
    {}

    public void keyTyped (KeyEvent e)
    {
        if ((int)e.getKeyChar () == KeyEvent.VK_ENTER)
            e.setKeyChar ('=');
        else
            if ((int)e.getKeyChar () == KeyEvent.VK_ESCAPE)
                e.setKeyChar ('C');

        e.setKeyChar (Character.toUpperCase (e.getKeyChar ()));

        Date D = new Date ();

        for (int I = 0; I < Botao.length; I++)
            if (e.getKeyChar () == this.Botao [I].getLabel ().charAt (0))
            {
                this.Botao [I].dispatchEvent (new MouseEvent (
                                              this.Botao [I],
                                              MouseEvent.MOUSE_CLICKED,
                                              D.getTime (),
                                              0, 0, 0, 1, false));
                break;
            }
    }

    public void keyPressed (KeyEvent e)
    {}

    public void keyReleased (KeyEvent e)
    {}
}
```

[C:\ExplsJava\Expl_13\Calculadora.html]

```
<html>
    <head>
        <title>Exemplo de Applet</title>
    </head>

    <body bgcolor="#000000" text="ffffff">
        <div align="center">
            <h1>Calculadora</h1>

            <applet codebase="." code="Calculadora.class" width=350 height=400>
            </applet>
        </div>
    </body>
</html>
```

Para compilar este programa, daremos o seguinte comando:

C:\ExplsJava\Expl_13> javac Calculadora.java

Para executar este programa, basta carregar em um browser o arquivo C:\ExplsJava\Expl_13\Calculadora.html.

Isto fará com que seja exibida no browser uma página como a que vemos abaixo:

O usuário poderá interagir com ela de forma idêntica àquela que emprega quando usa uma calculadora em uma janela.

Uma applet pode, eventualmente, abrir janelas independentes do browser. Tais janelas são chamadas de applet windows e estão sujeitas às mesmas restrições que as applets.

[C:\ExplsJava\Expl_14\Calculadora.java]

```
import java.awt.*;
import java.awt.event.*;
import java.util.Date;

public class Calculadora extends    Frame
                         implements WindowListener,
                                    MouseListener,
                                    KeyListener
{
    private Label  Display = new Label ("", Label.RIGHT);
    private Button Botao [];

    private boolean Primeiro = true;
```

```
    private double  Operando1;
    private char    Operador = ' ';

    private void TrateClickEmBotaoNumerico (char N)
    {
        if (this.Primeiro)
        {
            this.Display.setText ("" + N);
            this.Primeiro = false;
        }
        else
            this.Display.setText (this.Display.getText () + N);
    }

    private void TrateClickEmBotaoDeOperacao (char O)
    {
        if (!this.Display.getText ().equals (""))
        {
            this.TrateClickNoBotaoIgual ();

            this.Operando1 = new Double
                           (this.Display.getText ())
                           .doubleValue ();

            this.Operador = O;
        }
    }

    private void TrateClickNoBotaoIgual ()
    {
        if ((!(this.Display.getText ().equals (""))) && (this.Operador != ' '))
        {
            double Operando2 = new Double
                             (this.Display.getText ()).doubleValue (),

                   Resultado = 0;

            switch (this.Operador)
            {
                case '+':
                    Resultado = this.Operando1 + Operando2;
                    break;

                case '-':
                    Resultado = this.Operando1 - Operando2;
                    break;

                case '*':
                    Resultado = this.Operando1 * Operando2;
                    break;

                case '/':
                    Resultado = this.Operando1 / Operando2;
            }

            if (Math.round (Resultado) == Resultado)
                this.Display.setText ("" + Math.round (Resultado));
            else
                this.Display.setText ("" + Resultado);

            this.Operador = ' ';
        }

        this.Primeiro = true;
    }

    private void TrateClickNoBotaoDeLimpar ()
    {
```

```
        this.Display.setText ("");


        this.Operador = ' ';
        this.Primeiro = true;
    }

    public Calculadora ()
    {
        this.setTitle ("Calculadora");
        this.setSize  (200, 200);

        BorderLayout LayoutCalculadora = new BorderLayout ();
        this.setLayout (LayoutCalculadora);

        this.addWindowListener (this);

        Panel      Botoes       = new Panel ();
        GridLayout LayoutBotoes = new GridLayout (4, 4);

        Botoes.setLayout (LayoutBotoes);

        String Rotulo [] = {"7", "8", "9", "+",
                            "4", "5", "6", "-",
                            "1", "2", "3", "*",
                            "0", "=", "C", "/"};

        this.Botao = new Button [Rotulo.length];

        for (int I = 0; I < this.Botao.length; I++)
        {
            this.Botao [I] = new Button (Rotulo [I]);

            this.Botao [I].addMouseListener (this);
            this.Botao [I].addKeyListener   (this);

            Botoes.add (this.Botao [I]);
        }

        this.add ("North",  this.Display);
        this.add ("Center", Botoes);

        this.show ();
    }

    public void windowClosing (WindowEvent e)
    {
        this.hide    ();
        this.dispose ();
    }

    public void windowOpened (WindowEvent e)
    {}

    public void windowClosed (WindowEvent e)
    {}

    public void windowActivated (WindowEvent e)
    {}

    public void windowDeactivated (WindowEvent e)
    {}

    public void windowIconified (WindowEvent e)
    {}

    public void windowDeiconified (WindowEvent e)
    {}
```

```
    public void mouseReleased (MouseEvent e)
    {
        char RotuloDoBotao = ((Button)e.getComponent()).getLabel().charAt(0);

        switch (RotuloDoBotao)
        {
            case '0':
            case '1':
            case '2':
            case '3':
            case '4':
            case '5':
            case '6':
            case '7':
            case '8':
            case '9': this.TrateClickEmBotaoNumerico (RotuloDoBotao);
                      break;

            case '+':
            case '-':
            case '*':
            case '/': TrateClickEmBotaoDeOperacao (RotuloDoBotao);
                      break;

            case '=': TrateClickNoBotaoIgual ();
                      break;

            case 'C': TrateClickNoBotaoDeLimpar ();
        }
    }

    public void mousePressed (MouseEvent e)
    {}

    public void mouseClicked (MouseEvent e)
    {}

    public void mouseEntered (MouseEvent e)
    {}

    public void mouseExited (MouseEvent e)
    {}

    public void keyTyped (KeyEvent e)
    {
        if ((int)e.getKeyChar () == KeyEvent.VK_ENTER)
            e.setKeyChar ('=');
        else
            if ((int)e.getKeyChar () == KeyEvent.VK_ESCAPE)
                e.setKeyChar ('C');

        e.setKeyChar (Character.toUpperCase (e.getKeyChar ()));

        Date D = new Date ();

        for (int I = 0; I < Botao.length; I++)
            if (e.getKeyChar () == this.Botao [I].getLabel ().charAt (0))
            {
                this.Botao [I].dispatchEvent (new MouseEvent (
                                              this.Botao [I],
                                              MouseEvent.MOUSE_CLICKED,
                                              D.getTime (),
                                              0, 0, 0, 1, false));
                break;
            }
    }

    public void keyPressed (KeyEvent e)
    {}
```

```
    public void keyReleased (KeyEvent e)
    {}
}
```

[C:\ExplsJava\Expl_14\Acionador.java]

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class Acionador extends    Applet
                       implements MouseListener
{
    private Button Calculos = new Button ("Calculos");

    public void start ()
    {
        GridLayout LayoutAcionador = new GridLayout (1, 1);
        this.setLayout (LayoutAcionador);

        Calculos.addMouseListener (this);
        this.add (Calculos);
    }

    public void mouseClicked (MouseEvent e)
    {
        Calculadora calculadora = new Calculadora ();
    }

    public void mouseReleased (MouseEvent e)
    {}

    public void mousePressed (MouseEvent e)
    {}

    public void mouseEntered (MouseEvent e)
    {}

    public void mouseExited (MouseEvent e)
    {}
}
```

[C:\ExplsJava\Expl_14\Calculadora.html]

```
<html>
    <head>
        <title>Exemplo de Applet</title>
    </head>

    <body bgcolor="#000000" text="ffffff">
        <div align="center">
            <h1>Calculadora</h1>
            <h3>Para acionar uma calculadora, clique no botão a seguir:<h3>
            <applet codebase="." code="Acionador.class" width=80 height=20>
            </applet>
        </div>
    </body>
```

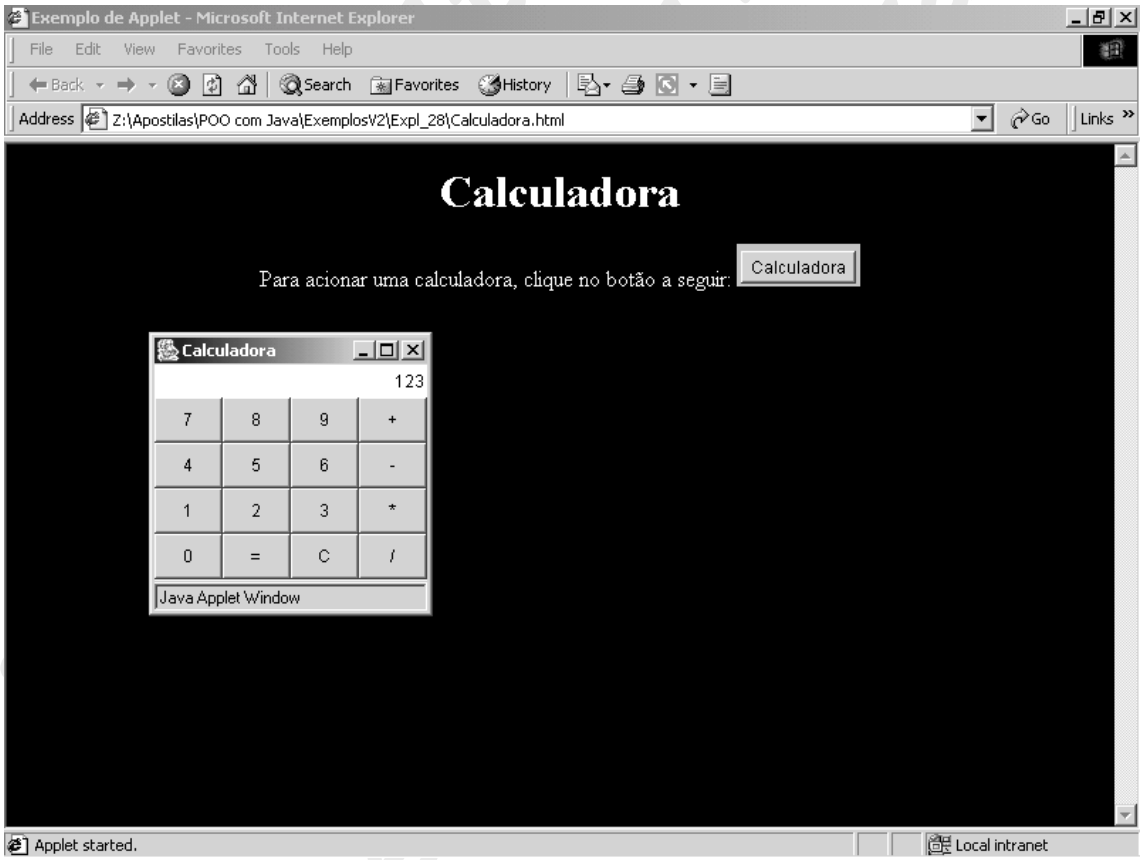
```
</html>
```

Para compilar este programa, daremos o seguinte comando:

```
C:\ExplsJava\Expl_14> javac Calculadora.java
C:\ExplsJava\Expl_14> javac Acionador.java
```

Para executar este programa, basta carregar em um browser o arquivo C:\ExplsJava\Expl_14\Calculadora.html.

Isto fará com que seja exibida no browser uma página como a que vemos abaixo; podemos também observar na figura a applet window aberta ao clicar no botão rotulado Calculadora na página exibida no browser:



O usuário poderá interagir com ela de forma idêntica àquela que emprega quando usa uma calculadora em uma janela.

Classes de Apoio

java.applet.AppletContext

Esta interface pode ser vista como o contexto de uma Applet, i.e., o documento contendo a Applet e outras Applet's do mesmo documento. Os métodos aqui especificados servem para permitir que uma Applet se informe a respeito de seu ambiente.

Além das estruturas e do comportamento herdado da classe java.awt.Panel, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Métodos:**
  - **Applet getApplet (String N):**
    Retorna a Applet do contexto desta Applet que tem o nome N;
  - **Enumeration getApplets ():**
    Retorna uma enumeração com todas as Applet's do contexto desta Applet;
  - **AudioClip getAudioClip (URL U):**
    Retorna o AudioClip especificado pela URL U;
  - **Image getImage (URL U):**
    Retorna a Image especificada pela URL U;
  - **void showDocument (URL U):**
    Substitui a página WEB que está sendo mostrada no browser pela especificada pela URL U;
  - **void showDocument (URL U, String N):**
    Substitui a página WEB que está sendo mostrada no browser pela especificada pela URL U e nome N;
  - **void showStatus (String E):**
    Faz com o estado E seja exibido na janela de exibição de estado.

java.awt.Image

Esta classe especifica um componente gráfico no qual é possível carregar e mostrar imagens gravadas em arquivos no padrão GIF ou JPEG. Para carregar uma Image faz-se uso da função

getImage da classe Applet e da classe AppletContext. Para exibir uma Image, faz-se uso da função drawImage da classe Graphics.

java.awt.AudioClip

Um AudioClip é uma classe que especifica um componente no qual é possível carregar e tocar sons gravadas em arquivos no padrão AU. Para carregar um AudioClip faz-se uso da função getAudioClip da classe Applet e da classe AppletContext. Para reproduzir um AudioClip, faz-se uso dos métodos da própria classe AudioClip.

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Métodos:**
  - **void play ():**
    Reproduz este AudioClip;
  - **void loop ():**
    Reproduz continuamente este AudioClip;
  - **void stop ():**
    Interrompe a reprodução deste AudioClip.

[C:\ExplsJava\Expl_15\ImgSom.java]

```
import java.awt.*;
import java.applet.*;

public class ImgSom extends Applet
{
    private Image     Imagem;
    private AudioClip Som;

    public String [][] getParameterInfo ()
    {
        String [][] R = {{"Imagem", "string", "A imagem a ser mostrada"},
                         {"Som"   , "string", "O som a ser tocado"}};

        return R;
    }

    public void init ()
    {
        String NomArqImg = this.getParameter ("Imagem");
        String NomArqSom = this.getParameter ("Som");

        this.Imagem = this.getImage     (this.getDocumentBase (), NomArqImg);
```

```
        this.Som    = this.getAudioClip (this.getDocumentBase (), NomArqSom);
    }

    public void paint (Graphics G)
    {
        G.drawImage (this.Imagem, 0, 0, this);
    }

    public void start ()
    {
        this.repaint  ();
        this.Som.loop ();
    }

    public void stop ()
    {
        this.Som.stop ();
    }
}
```

[C:\ExplsJava\Expl_15\ImgSom.html]

```
<html>
    <head>
        <title>Exemplo de Som e Imagem</title>
    </head>

    <body bgcolor="#000000" text="ffffff">
        <div align="center">
            <h1>Paraíso Tropical</h1>

            <applet codebase="." code="ImgSom.class" width=800 height=600>
                <param name="Imagem" value="Praia.jpg">
                <param name="Som"    value="Moderna.au">
            </applet>
        </div>
    </body>
</html>
```

Para compilar este programa, daremos o seguinte comando:

C:\ExplsJava\Expl_15> javac ImgSom.java

Para executar este programa, basta carregar em um browser o arquivo C:\ExplsJava\Expl_15\ImgSom.html.

Isto fará com que seja exibida no browser uma página como a que vemos abaixo:

Além de ver a janela acima mostrada na tela, o usuário poderá ouvir a reprodução de uma trilha sonora.

Threads

Programas convencionais são monothreaded, i.e., seguem um único fluxo de execução. Multithreading representa a possiblidade de diversos fluxos de execução coexistirem dentro de um mesmo programa, i.e., permite um paralelismo dentro do programa.

Veja abaixo a lista dos campos da classe Thread, de seus construtores e de seus métodos:

- **Campos:**

  - **static int MAX_PRIORITY:**

    Representa a máxima prioridade que pode ser atribuída a uma thread;

  - **static int MIN_PRIORITY:**

    Representa a mínima prioridade que pode ser atribuída a uma thread;

- **static int NORM_PRIORITY:**

  Representa a prioridade que normalmente é atribuída a uma thread;

- **Construtores:**

  - **Thread ():**

    Constroi uma instância da classe Thread;

  - **Thread (Runnable R):**

    Constroi uma instância da classe Thread que executará o método run de R;

  - **Thread (Runnable R, String N):**

    Constroi uma instância da classe Thread chamada N e que executará o método run de R;

  - **Thread (String N):**

    Constroi uma instância da classe Thread chamada N;

  - **Thread (ThreadGroup G, Runnable R):**

    Constroi uma instância da classe Thread vinculada ao grupo G e que executará o método run de R;

  - **Thread (ThreadGroup G, Runnable R, String N):**

    Constroi uma instância da classe Thread chamada N, vinculada ao grupo G e que executará o método run de R;

  - **Thread (ThreadGroup G, String N):**

    Constroi uma instância da classe Thread chamada N e vinculada ao grupo G;

- **Métodos:**

  - **static int activeCount ():**

    Retorna a quantidade de threads ativas no grupo da Thread corrente;

  - **void checkAccess ():**

    Verifica se a Thread corrente tem permissão para acessar a Thread à qual se refere e lança a exceção SecurityException em caso negativo.

- **static Thread currentThread ():**

  Retorna a Thread corrente;

- **void destroy ():**

  Destroi a Thread à qual se refere sem preliminares nem posliminares;

- **static void dumpStack ():**

  Escreve um stack trace da Thread corrente;

- **static int enumerate (Thread T []):**

  Copia em T cada Thread ativa do grupo da Thread corrente e de todos os seus subgrupos;

- **String getName ():**

  Retorna o nome da Thread;

- **int getPriority ():**

  Retorna a prioridade da Thread;

- **ThreadGroup getThreadGroup ():**

  Retorna o grupo da Thread;

- **void interrupt ():**

  Interrompe esta Thread;

- **static boolean interrupted ():**

  Verifica se a Thread corrente foi interrompida;

- **boolean isAlive ():**

  Verifica se a Thread à qual se refere sem preliminares nem posliminares;

- **boolean isDeamon ():**

  Verifica se a Thread é uma thread deamon;

- **boolean isInterrupted ():**

  Verdifica se a Thread foi interrompida;

- **void join ():**

  Espera a morte da Thread;

- **void join (long m):**

  Espera no máximo m milisegundos pela morte da Thread;

- **void join (long M, int N):**

  Espera no máximo M milisegundos mais N nanosegundos pela morte da Thread;

- **void run ():**

  Se o construtor da Thread recebeu um objeto que implementa a interface Runnable, então executa o método run daquele objeto, caso contrário não faz nada e retorna;

- **void setDeamon (boolean D):**

  Torna a Thread uma deamon thread ou uma thread de usuário;

- **void setName (String N):**

  Torna N o nome da Thread;

- **void setPriority (int P):**

  Torna P a nova prioridade da Thread;

- **static void sleep (long M):**

  Faz com que a thread corrente cesse sua exeução por M milisegundos;

- **static void sleep (long M, int N):**

  Faz com que a thread corrente cesse sua exeução por M milisegundos mais N nanosegundos;

- **void start ():**

  Faz com que a Thread entre em execução (a JVM ativa o método run da Thread);

- **static void yield ():**

  Faz com que a thread corrente deixe temporariamente de ser executada, deixando outra thread executar;

Sincronismo

Muitos sistemas operacionais modernos têm as primitivas básicas para criar e rodar threads mas a maioria deles não é thread-safe. Java foi escrita a partir do zero tendo threads em mente, por isso todas as classes da biblioteca de Java são thread-safe.

Demora um pouco para se acostumar com threads; conceber um programa com várias linhas de execução em andamento, possivelmente que se influenciam mutuamente e que precisam sincronizar-se requer o desenvolvimento de uma nova forma de pensar.

Veja abaixo as facilidade que a linguagem Java oferece para a sincronização de threads:

1. Para se ter garantia de que somente uma thread está em um dado momento executando um certo método, basta qualificá-lo com o modificador synchronized;

2. Para se ter garantia de acesso exclusivo a um objeto, basta utilizá-lo em um bloco que exija acesso sincronizado àquele objeto; sendo Obj um objeto, veja abaixo a forma geral de um trecho de código durante o qual é garantido o acesso exclusivo ao objeto Obj:

```
synchronized (Obj)
{
    …
}
```

3. Para se ter garantia de acesso exclusivo às variáveis privativas da classe do objeto cujo método está correntemente em execução, basta fazê-lo em um bloco que exija acesso sincronizado àquela classe; veja abaixo a forma geral de um trecho de código cuja execução tem a garantia de acesso exclusivo a tais variáveis:

```
synchronized (getClass ())
{
    …
}
```

4. Para se ter garantia de acesso variáveis públicas de uma classe diferente daquela do método que está correntemente em execução, basta fazê-lo em um bloco que exija acesso sincronizado àquela classe; sendo Classe o identificador de uma classe, veja abaixo a forma geral de um trecho de código cuja execução tem a garantia de acesso exclusivo a tais variáveis:

```
synchronized (Class.for.Name ("Classe"))
{
    …
}
```

Scheduler

Basicamente, existem três tipos de schedulers de threads, o preemptivo, o preemptivo com prioridades e o não preemptivo. As diferenças desses três tipos de schedulers consistem do seguinte:

- O primeiro tipo de scheduler dá às threads uma fração de tempo sempre do mesmo tamanho para executar. Neste tipo de scheduler as threads podem simplesmente rodar livremente e sem terem que se preocupar com o tempo, já que no momento apropriado elas serão interrompidas a fim de destinar a CPU à execução de outras threads que eventualmente aguardam.
- O segundo tipo de scheduler dá às threads uma fração de tempo de tamanho proporcional à suas prioridades para executar. Pelas mesmas razões, neste de scheduler as threads também podem simplesmente rodar livremente e sem terem que se preocupar com o tempo.
- O terceiro tipo de scheduler dá a cada thread o direito de ser executada por um tempo arbitrariamente grande (só para quando explicitamente abrir mão do seu direito de ser executada). Passa-se então a executar a segunda, a terceira e assim por diante.

Possíveis Implementações

Uma primeira possibilidade seria implementar uma classe derivada da classe Thread. Neste caso, faz-se o seguinte: cria-se uma classe para representar a thread desejada, diz-se que a mesma extende a classe Thread com a expressão extends Thread.

É preciso agora implementar um método público de nome run, sem retorno e sem parâmetros, que funcionará como função principal da Thread, i.e., a função que é executada quando a Thread entra em operação. Feito isso, pode-se agora criar instâncias da classe em questão e controlá-la através dos métodos que herda da classe Thread.

[C:\ExplsJava\Expl_16\ExcecaoDeFila.java]

```
public class ExcecaoDeFila extends Exception
{
    public static final int CAPACIDADE_INVALIDA = 0,
                            FILA_CHEIA          = 1,
                            FILA_VAZIA          = 2,
                            EXCECAO_INVALIDA    = 3;


    private static final int MENOR_ID = CAPACIDADE_INVALIDA,
                             MAIOR_ID = FILA_VAZIA;

    public static final String Mensagem [] =
    {
         "Capacidade deve ser um numero natural positivo",
         "Esgotou-se a capacidade da agenda",
         "Nao ha contatos registrados na agenda",
         "O nome deste contato nao consta na agenda",
         "O nome deste contato ja consta na agenda",
         "So solicite telefones entre zero e a quantidade existente",
         "So solicite nomes entre zero e a quantidade existente",
         "Indicacao de excecao invalida"
    };

    private int Tipo;

    private ExcecaoDeFila (int T)
    {
        super (ExcecaoDeFila.Mensagem [T]);
        this.Tipo = T;
    }

    public static void indique (int T) throws ExcecaoDeFila
    {
        if (T<ExcecaoDeFila.MENOR_ID || T>ExcecaoDeFila.MAIOR_ID)
            throw new ExcecaoDeFila (ExcecaoDeFila.EXCECAO_INVALIDA);

        throw new ExcecaoDeFila (T);

    }

    public int seuTipo ()
    {
        return this.Tipo;
    }
}
```

[C:\ExplsJava\Expl_16\Fila.java]

```
class Fila
{
    private Object Elem [];

    private int OndeGuardar = 0,
                DeOndeTirar = 0,
                QtosElems   = 0;
```

```
    public Fila (int T) throws ExcecaoDeFila
    {
        if (T <= 0)
            ExcecaoDeFila.indique (ExcecaoDeFila.CAPACIDADE_INVALIDA);

        this.Elem = new Object [T];
    }

    public Fila () throws ExcecaoDeFila
    {
        this (25);
    }

    public int tamanho ()
    {
        return this.Elem.length;
    }

    public boolean cheia ()
    {
        return this.QtosElems == this.Elem.length;
    }

    public boolean vazia ()
    {
        return this.QtosElems == 0;
    }

    public void enfileire (Object O) throws ExcecaoDeFila
    {
        if (this.cheia ())
            ExcecaoDeFila.indique (ExcecaoDeFila.FILA_CHEIA);

        this.Elem [this.OndeGuardar] = O;

        this.OndeGuardar = (this.OndeGuardar + 1) % this.Elem.length;
        this.QtosElems++;
    }

    public Object elementoDesenfileirado () throws ExcecaoDeFila
    {
        if (this.vazia ())
            ExcecaoDeFila.indique (ExcecaoDeFila.FILA_VAZIA);

        Object R = this.Elem [this.DeOndeTirar];

        this.DeOndeTirar = (this.DeOndeTirar + 1) % this.Elem.length;
        this.QtosElems--;

        return R;
    }
}
```

[C:\ExplsJava\Expl_16\Semaforo.java]

```
class Semaforo
{
    private int QtdDeRecursos;
    private int QtdDeProcsEsperando = 0;
```

```
    Semaforo (int S)
    {
        this.QtdDeRecursos = S;
    }

    public synchronized void P ()
    {
        if (this.QtdDeRecursos > 0)
            this.QtdDeRecursos--;
        else
        {
            this.QtdDeProcsEsperando++;

            try
            {
                this.wait ();
            }
            catch (InterruptedException E)
            {}
        }
    }

    public synchronized void V ()
    {
        if (this.QtdDeProcsEsperando > 0)
        {
            this.notify ();
            this.QtdDeProcsEsperando--;
        }
        else
            this.QtdDeRecursos++;
    }
}
```

[C:\ExplsJava\Expl_16\Produtor.java]

```
class Produtor extends Thread
{
    private final int TpoMaxProd = 1000;

    private Fila     Deposito;
    private Semaforo MutEx, Livre, Ocupado;

    private boolean Fim = false;

    private void produza ()
    {
        Integer Produto = new Integer ((int)(Math.random () * 1000));

        try
        {
            this.sleep ((int)(Math.random () * this.TpoMaxProd));
        }
        catch (InterruptedException E)
        {}

        this.MutEx.P ();
        try
        {
            this.Deposito.enfileire (Produto);
```

```
        }
        catch (ExcecaoDeFila E)
        {}
        this.MutEx.V ();

        System.out.println ("PRODUZIDO: " + Produto);
    }

    public Produtor (Fila     D, Semaforo M,
                     Semaforo L, Semaforo O)
    {
        this.Deposito = D;
        this.MutEx    = M;
        this.Livre    = L;
        this.Ocupado  = O;
    }

    public void Stop ()
    {
        this.Fim = true;
    }

    public void run ()
    {
        for (;!this.Fim;)
        {
            this.Livre.P   ();
            this.produza   ();
            this.Ocupado.V ();
        }
    }
}
```

[C:\ExplsJava\Expl_16\Consumidor.java]

```
class Consumidor extends Thread
{
    private final int TpoMaxCons = 2000;

    private Fila     Deposito;
    private Semaforo MutEx, Livre, Ocupado;

    private boolean Fim = false;

    private void consuma ()
    {
        try
        {
            this.sleep ((int)(Math.random () * this.TpoMaxCons));
        }
        catch (InterruptedException E)
        {}

        Integer Produto = null;
        this.MutEx.P ();
        try
        {
            Produto = (Integer)this.Deposito.elementoDesenfileirado ();
        }
        catch (ExcecaoDeFila E)
```

```
        {}
        this.MutEx.V ();

        System.out.println ("CONSUMIDO: " + Produto);
    }

    public Consumidor (Fila     D, Semaforo M,
                       Semaforo L, Semaforo O)
    {
        this.Deposito = D;
        this.MutEx    = M;
        this.Livre    = L;
        this.Ocupado  = O;
    }

    public void Stop ()
    {
        this.Fim = true;
    }

    public void run ()
    {
        for (;!this.Fim;)
        {
            this.Ocupado.P ();
            this.consuma   ();
            this.Livre.V   ();
        }
    }
}
```

[C:\ExplsJava\Expl_16\TesteDePeC.java]

```
class TesteDePeC
{
    private static final int TAMANHO_DO_DEPOSITO = 15;

    private static Semaforo MutEx   = new Semaforo (1),
                            Livre   = new Semaforo (TAMANHO_DO_DEPOSITO),
                            Ocupado = new Semaforo (0);

    private static Fila       Deposito;
    private static Produtor   Prod;
    private static Consumidor Cons1, Cons2;

    public static void main (String Args [])
    {
        try
        {
            TesteDePeC.Deposito = new Fila (TAMANHO_DO_DEPOSITO);
        }
        catch (ExcecaoDeFila E)
        {}

        TesteDePeC.Prod  = new Produtor   (Deposito, MutEx,
                                           Livre,    Ocupado);

        TesteDePeC.Cons1 = new Consumidor (Deposito, MutEx,
                                           Livre,    Ocupado);
```

```
        TesteDePeC.Cons2 = new Consumidor (Deposito, MutEx,
                                           Livre,    Ocupado);

        Prod .start ();
        Cons1.start ();
        Cons2.start ();
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_16> javac Principal.java
C:\ExplsJava\Expl_16> java Principal
```

Isto poderá produzir no console a seguinte saída:

```
PRODUZIDO: 179
CONSUMIDO: 179
PRODUZIDO: 200
CONSUMIDO: 200
PRODUZIDO: 127
CONSUMIDO: 127
PRODUZIDO: 749
CONSUMIDO: 749
PRODUZIDO: 582
PRODUZIDO: 982
CONSUMIDO: 582
CONSUMIDO: 982
PRODUZIDO: 565
CONSUMIDO: 565
PRODUZIDO: 822
CONSUMIDO: 822
PRODUZIDO: 995
PRODUZIDO: 905
CONSUMIDO: 995
PRODUZIDO: 983
...
```

Uma outra possibilidade seria implementar uma classe que implementa a interface Runnable. Neste caso, faz-se o seguinte: cria-se uma classe para representar a thread desejada, diz-se que a mesma implementa a interface Runnable com a expressão implements Runnable.

É preciso agora implementar um método público de nome run, sem retorno e sem parâmetros, que funcionará como função principal da Thread, i.e., a função que é executada quando a Thread entra em operação.

Isso feito, deve-se declarar e instanciar um objeto da classe em questão, armazenando no mesmo uma nova instância da classe. Em seguida, deve-se declarar e instanciar um objeto da classe Thread, tomando-se o cuidado de passar para seu construtor o objeto inicialmente declarado e instanciado. Basta agora controlar a thread via o objeto da classe Thread .

[C:\ExplsJava\Expl_17\ExcecaoDeFila.java]

```
public class ExcecaoDeFila extends Exception
{
    public static final int CAPACIDADE_INVALIDA = 0,
                            FILA_CHEIA          = 1,
                            FILA_VAZIA          = 2,
                            EXCECAO_INVALIDA    = 3;


    private static final int MENOR_ID = CAPACIDADE_INVALIDA,
                             MAIOR_ID = FILA_VAZIA;

    public static final String Mensagem [] =
    {
         "Capacidade deve ser um numero natural positivo",
         "Esgotou-se a capacidade da agenda",
         "Nao ha contatos registrados na agenda",
         "O nome deste contato nao consta na agenda",
         "O nome deste contato ja consta na agenda",
         "So solicite telefones entre zero e a quantidade existente",
         "So solicite nomes entre zero e a quantidade existente",
         "Indicacao de excecao invalida"
    };

    private int Tipo;

    private ExcecaoDeFila (int T)
    {
        super (ExcecaoDeFila.Mensagem [T]);
        this.Tipo = T;
    }

    public static void indique (int T) throws ExcecaoDeFila
    {
        if (T<ExcecaoDeFila.MENOR_ID || T>ExcecaoDeFila.MAIOR_ID)
            throw new ExcecaoDeFila (ExcecaoDeFila.EXCECAO_INVALIDA);

        throw new ExcecaoDeFila (T);

    }

    public int seuTipo ()
    {
        return this.Tipo;
    }
}
```

[C:\ExplsJava\Expl_17\Fila.java]

```
class Fila
{
    private Object Elem [];

    private int OndeGuardar = 0,
                DeOndeTirar = 0,
                QtosElems   = 0;

    public Fila (int T) throws ExcecaoDeFila
    {
        if (T <= 0)
            ExcecaoDeFila.indique (ExcecaoDeFila.CAPACIDADE_INVALIDA);

        this.Elem = new Object [T];
    }

    public Fila () throws ExcecaoDeFila
    {
        this (25);
    }

    public int tamanho ()
    {
        return this.Elem.length;
    }

    public boolean cheia ()
    {
        return this.QtosElems == this.Elem.length;
    }

    public boolean vazia ()
    {
        return this.QtosElems == 0;
    }

    public void enfileire (Object O) throws ExcecaoDeFila
    {
        if (this.cheia ())
            ExcecaoDeFila.indique (ExcecaoDeFila.FILA_CHEIA);

        this.Elem [this.OndeGuardar] = O;

        this.OndeGuardar = (this.OndeGuardar + 1) % this.Elem.length;
        this.QtosElems++;
    }

    public Object elementoDesenfileirado () throws ExcecaoDeFila
    {
        if (this.vazia ())
            ExcecaoDeFila.indique (ExcecaoDeFila.FILA_VAZIA);

        Object R = this.Elem [this.DeOndeTirar];

        this.DeOndeTirar = (this.DeOndeTirar + 1) % this.Elem.length;
        this.QtosElems--;

        return R;
    }
}
```

[C:\ExplsJava\Expl_17\Semaforo.java]

```
class Semaforo
{
    private int QtdDeRecursos;
    private int QtdDeProcsEsperando = 0;

    Semaforo (int S)
    {
        this.QtdDeRecursos = S;
    }

    public synchronized void P ()
    {
        if (this.QtdDeRecursos > 0)
            this.QtdDeRecursos--;
        else
        {
            this.QtdDeProcsEsperando++;

            try
            {
                this.wait ();
            }
            catch (InterruptedException E)
            {}
        }
    }

    public synchronized void V ()
    {
        if (this.QtdDeProcsEsperando > 0)
        {
            this.notify ();
            this.QtdDeProcsEsperando--;
        }
        else
            this.QtdDeRecursos++;
    }
}
```

[C:\ExplsJava\Expl_17\Produtor.java]

```
class Produtor implements Runnable
{
    private final int TpoMaxProd = 1000;

    private Fila     Deposito;
    private Semaforo MutEx, Livre, Ocupado;

    private Thread  Processo;
    private boolean Fim = false;

    private void produza ()
    {
        Integer Produto = new Integer ((int)(Math.random () * 1000));

        try
```

```
        {
            this.Processo.sleep ((int)(Math.random () * this.TpoMaxProd));
        }
        catch (InterruptedException E)
        {}

        this.MutEx.P ();
        try
        {
            this.Deposito.enfileire (Produto);
        }
        catch (ExcecaoDeFila E)
        {}
        this.MutEx.V ();

        System.out.println ("PRODUZIDO: " + Produto);
    }

    public Produtor (Fila     D, Semaforo M,
                     Semaforo L, Semaforo O)
    {
        this.Deposito = D;
        this.MutEx    = M;
        this.Livre    = L;
        this.Ocupado  = O;
        this.Processo = new Thread (this);
    }

    public void start ()
    {
        this.Processo.start ();
    }

    public void stop ()
    {
        this.Fim = true;
    }

    public void run ()
    {
        for (;!this.Fim;)
        {
            this.Livre.P   ();
            this.produza   ();
            this.Ocupado.V ();
        }
    }
}
```

[C:\ExplsJava\Expl_17\Consumidor.java]

```
class Consumidor implements Runnable
{
    private final int TpoMaxCons = 2000;

    private Fila     Deposito;
    private Semaforo MutEx, Livre, Ocupado;

    private Thread  Processo;
    private boolean Fim = false;
```

```
    private void consuma ()
    {
        try
        {
            this.Processo.sleep ((int)(Math.random () * this.TpoMaxCons));
        }
        catch (InterruptedException E)
        {}

        Integer Produto = null;
        this.MutEx.P ();
        try
        {
            Produto = (Integer)this.Deposito.elementoDesenfileirado ();
        }
        catch (ExcecaoDeFila E)
        {}
        this.MutEx.V ();

        System.out.println ("CONSUMIDO: " + Produto);
    }

    public Consumidor (Fila     D, Semaforo M,
                       Semaforo L, Semaforo O)
    {
        this.Deposito = D;
        this.MutEx    = M;
        this.Livre    = L;
        this.Ocupado  = O;
        this.Processo = new Thread (this);
    }

    public void start ()
    {
        this.Processo.start ();
    }

    public void stop ()
    {
        this.Fim = true;
    }

    public void run ()
    {
        for (;!this.Fim;)
        {
            this.Ocupado.P ();
            this.consuma   ();
            this.Livre.V   ();
        }
    }
}
```

[C:\ExplsJava\Expl_17\TesteDePeC.java]

```
class TesteDePeC
{
    private static final int TAMANHO_DO_DEPOSITO = 15;
```

```
    private static Semaforo MutEx   = new Semaforo (1),
                            Livre   = new Semaforo (TAMANHO_DO_DEPOSITO),
                            Ocupado = new Semaforo (0);

    private static Fila       Deposito;
    private static Produtor   Prod;
    private static Consumidor Cons1, Cons2;

    public static void main (String Args [])
    {
        try
        {
            TesteDePeC.Deposito = new Fila (TAMANHO_DO_DEPOSITO);
        }
        catch (ExcecaoDeFila E)
        {}

        TesteDePeC.Prod  = new Produtor   (Deposito, MutEx,
                                           Livre,    Ocupado);

        TesteDePeC.Cons1 = new Consumidor (Deposito, MutEx,
                                           Livre,    Ocupado);

        TesteDePeC.Cons2 = new Consumidor (Deposito, MutEx,
                                           Livre,    Ocupado);

        Prod .start ();
        Cons1.start ();
        Cons2.start ();
    }
}
```

Para compilar e executar este programa, daremos os seguintes comandos:

```
C:\ExplsJava\Expl_17> javac Principal.java
C:\ExplsJava\Expl_17> java Principal
```

Isto poderia produzir no console a seguinte saída:

```
PRODUZIDO: 179
CONSUMIDO: 179
PRODUZIDO: 200
CONSUMIDO: 200
PRODUZIDO: 127
CONSUMIDO: 127
PRODUZIDO: 749
CONSUMIDO: 749
PRODUZIDO: 582
PRODUZIDO: 982
CONSUMIDO: 582
CONSUMIDO: 982
PRODUZIDO: 565
CONSUMIDO: 565
PRODUZIDO: 822
```

```
CONSUMIDO: 822
PRODUZIDO: 995
PRODUZIDO: 905
CONSUMIDO: 995
PRODUZIDO: 983
. . .
```

Applet's e Thread's

Usar threads em applets é uma boa prática de programação pois isso leva à construção de applets bem mais bem comportadas. Com threads pode-se criar applets que têm seu próprio fluxo de execução e que rodam sem interfacear com outras partes do sistema.

Troca de Informações via Rede (O Pacote java.net)

Aplicações desenvolvidas em Java podem trocar informações através da rede fazendo uso de sockets que funcionam como uma espécie de ponte entre a aplicação e a rede.

A Classe java.net.InetAddress

Instâncias da classe InetAddress representam endereços de máquinas na internet.

- **Métodos:**
  - **byte[] getAddress ():**
    Obtem o endereço IP da máquina associada a este InetAddress na forma de um vetor de bytes;
  - **static InetAddress getAllByName (String N):**
    Cria uma instância da classe InetAddress para representar a máquina cujo nome foi dado; essa instância contem informações como nome, endereços IP, etc;
  - **static InetAddress getByName (String N):**
    Cria uma instância da classe InetAddress para representar a máquina cujo nome foi dado; essa instância contem informações como nome, endereço IP, etc;

- **String getHostAddress ():**
  Obtem o endereço IP da máquina associada a este InetAddress na forma de um String;
- **String getHostName ():**
  Obtem o nome da máquina associada a este InetAddress;
- **static InetAddress getLocalHost ():**
  Cria uma instância da classe InetAddress para representar a máquina local; essa instância contem informações como nome, endereço IP, etc;
- **boolean IsMulticastAddress ():**
  Verifica se este InetAddress se refere a um IP de difusão;
- **String toString ():**
  Retorna este InetAddress na forma de uma instância da classe String.

A Classe java.net.ServerSocket

ServerSockets esperam a vinda de um pedido através da rede, realizam alguma operação baseada naquele pedido e então possivelmente retornam um resultado ao requerente.

- **Construtores:**
  - **ServerSocket (int P):**
    Constroi uma instância da classe ServerSocket; P é o número da porta na qual esperar pedidos;
  - **ServerSocket (int P, int TF):**
    Constroi uma instância da classe ServerSocket; P é o número da porta na qual esperar pedidos; TF é o tamanho máximo da fila de pedidos (se um pedido chegar e a fila de pedidos estiver cheia, a conexão será recusada).
  - **ServerSocket (int P, int TF, InetAddress M):**
    Constroi uma instância da classe ServerSocket; P é o número da porta da máquina M na qual esperar pedidos; TF é o tamanho máximo da fila de pedidos (se um pedido chegar e a fila de pedidos estiver cheia, a conexão será recusada);
- **Métodos:**

- **Socket accept ():**

  Espera um pedido de conexão e o aceita;

- **void close ():**

  Fecha este ServerSocket, não mais aceitando pedidos de conexão;

- **InetAddress getInetAddress ():**

  Obtem o endereço local deste ServerSocket;

- **int getLocalPort ():**

  Obtem a porta na qual este ServerSocket está esperando pedidos de conexão;

- **int getSoTimeout ():**

  Obtem o tempo de timeout estabelecido pelo sistema operacional (zero significa tempo infinito);

- **void setSoTimeout (int T):**

  Ajusta o tempo de timeout estabelecido pelo sistema operacional (zero significa tempo infinito);

- **String toString ():**

  Retorna o endereço local deste ServerSocket e a porta na qual ele espera pedidos de conexão na forma de uma instância da classe String.

A Classe java.net.Socket

Esta classe implementa sockets de clientes, ou simplesmente sockets.

- **Construtores:**

  - **Socket ():**

    Constroi uma instância da classe Socket não conectada;

  - **Socket (InetAddress S, int P):**

    Constroi uma instância da classe Socket e a conecta na porta P do servidor S;

- **Socket (InetAddress S, int PS, InetAddress L, int PL):**
  Constroi uma instância da classe Socket e a conecta na porta PS do servidor S e à porta PL da máquina local L;
- **Socket (String N, int P):**
  Constroi uma instância da classe Socket e a conecta na porta P do servidor de nome N;
- **Socket (String S, int P, InetAddress L, int PL):**
  Constroi uma instância da classe Socket e a conecta na porta PS do servidor S e à porta PL da máquina local L;

- **Métodos:**
  - **void close ():**
    Fecha este Socket, que passará a não mais aceitar transmissões ou recepções de dados;
  - **InetAddress getInetAddress ():**
    Obtem o endereço ao qual este Socket está conectado;
  - **InputStream getInputStream ():**
    Obtem um InputStream para este Socket;
  - **InetAddress getLocalAddress ():**
    Obtem o endereço local deste Socket;
  - **int getLocalPort ():**
    Obtem a porta que este Socket utiliza a fim de transmitir e receber dados;
  - **OutputStream getOutputStream ():**
    Obtem um OutputStream para este Socket;
  - **int getPort ():**
    Obtem a porta à qual este Socket está conectado;
  - **int getReceivedBufferSize ():**
    Obtem o tamanho do buffer de recepção de dados;

- **int getSendBufferSize ():**

  Obtem o tamanho do buffer de transmissão de dados;

- **int getSoTimeout ():**

  Obtem o tempo de timeout estabelecido pelo sistema operacional (zero significa tempo infinito);

- **boolean getTcpNoDelay ():**

  Verifica se está não está habilitado retardo TCP;

- **void setReceivedBufferSize (int T):**

  Ajusta o tamanho do buffer de recepção de dados;

- **void setSendBufferSize (int T):**

  Ajusta o tamanho do buffer de transmissão de dados;

- **void setSoTimeout (int T):**

  Obtem o tempo de timeout estabelecido pelo sistema operacional (zero significa tempo infinito);

- **void setTcpNoDelay (boolean B):**

  Habilita e desabilita retardo TCP;

- **String toString ():**

  Retorna este Socket na forma de uma instância da classe String.

[C:\ExplsJava\Expl_18\Prg_Recepcao\Recepcao.java]

```
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;

class Janela extends    Frame
             implements WindowListener,
                        MouseListener
{
    private Button   OK       = new Button ("OK");
    private TextArea Mensagem = new TextArea ();

    public Janela (String M)
    {
        this.setTitle ("Nova Mensagem");
        this.setSize  (400, 200);
```

```
        BorderLayout LayoutJanela = new BorderLayout ();
        this.setLayout (LayoutJanela);

        this.add ("Center", this.Mensagem);

        Panel Botoes = new Panel ();

        GridLayout LayoutBotoes = new GridLayout (1,5);
        Botoes.setLayout (LayoutBotoes);

        Botoes.add (new Label ());
        Botoes.add (new Label ());
        Botoes.add (this.OK);
        Botoes.add (new Label ());
        Botoes.add (new Label ());

        this.add ("South", Botoes);

        this.addWindowListener   (this);
        this.OK.addMouseListener (this);

        this.Mensagem.setText     (M);
        this.Mensagem.setEditable (false);

        this.show ();
    }



    public void windowClosing (WindowEvent e)
    {
        this.hide    ();
        this.dispose ();
    }

    public void windowOpened (WindowEvent e)
    {}

    public void windowClosed (WindowEvent e)
    {}

    public void windowActivated (WindowEvent e)
    {}

    public void windowDeactivated (WindowEvent e)
    {}

    public void windowIconified (WindowEvent e)
    {}

    public void windowDeiconified (WindowEvent e)
    {}



    public void mouseClicked (MouseEvent e)
    {}

    public void mousePressed (MouseEvent e)
    {}

    public void mouseReleased (MouseEvent e)
    {
        this.hide    ();
        this.dispose ();
    }
```

```
    public void mouseEntered (MouseEvent e)
    {}

    public void mouseExited (MouseEvent e)
    {}
}

class Recepcao
{
    public static void main (String Args [])
    {
        try
        {
            ServerSocket CanalDePedidoDeConexao = new ServerSocket (10000);

            for (;;)
            {
                Socket CanalDeRecepcaoDaMensagem =
                       CanalDePedidoDeConexao.accept ();

                BufferedReader EntradaDaMensagem = new BufferedReader
                                                   (new InputStreamReader
                                                   (CanalDeRecepcaoDaMensagem.
                                                    getInputStream ()));

                String TextoDaMensagem = "";

                for (;;)
                {
                     TextoDaMensagem = TextoDaMensagem +
                                       EntradaDaMensagem.readLine ();

                    if (EntradaDaMensagem.ready ())
                        TextoDaMensagem = TextoDaMensagem + '\n';
                    else
                        break;
                }

                EntradaDaMensagem.close ();
                EntradaDaMensagem = null;

                CanalDeRecepcaoDaMensagem.close ();
                CanalDeRecepcaoDaMensagem = null;

                Janela janela = new Janela (TextoDaMensagem);
            }
        }
        catch (IOException E)
        {
            System.err.println ("Impossivel criar canal de pedido de comunicacao.");
            System.err.println ();
        }
    }
}
```

[C:\ExplsJava\Expl_18\PrgEnvio\Envio.java]

```
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;

class Janela extends    Frame
             implements WindowListener,
                        MouseListener
```

```
{
    private String   Endereco;

    private Button OK      = new Button ("OK"),
                   Cancela = new Button ("Cancela");

    private TextArea Mensagem = new TextArea ();

    public Janela (String Endereco)
    {
        this.Endereco = Endereco;

        this.setTitle ("Entre sua mensagem");
        this.setSize  (400, 200);

        BorderLayout LayoutJanela = new BorderLayout ();
        this.setLayout (LayoutJanela);

        this.add ("Center", this.Mensagem);

        Panel Botoes = new Panel ();

        GridLayout LayoutBotoes = new GridLayout (1,7);
        Botoes.setLayout (LayoutBotoes);

        Botoes.add (new Label ());
        Botoes.add (new Label ());
        Botoes.add (this.OK);
        Botoes.add (new Label ());
        Botoes.add (this.Cancela);
        Botoes.add (new Label ());
        Botoes.add (new Label ());

        this.add ("South", Botoes);

        this   .addWindowListener (this);
        OK     .addMouseListener  (this);
        Cancela.addMouseListener  (this);

        this.show ();
    }



    public void windowClosing (WindowEvent e)
    {
        System.exit (0);
    }

    public void windowOpened (WindowEvent e)
    {}

    public void windowClosed (WindowEvent e)
    {}

    public void windowActivated (WindowEvent e)
    {}

    public void windowDeactivated (WindowEvent e)
    {}

    public void windowIconified (WindowEvent e)
    {}

    public void windowDeiconified (WindowEvent e)
    {}
```

```
    private void TrataClickNoBotaoOK ()
    {
        if (!Mensagem.getText().equals(""))
        {
            try
            {
                Socket CanalDeTransmissaoDaMensagem = new Socket (Endereco, 10000);

                PrintWriter SaidaDaMensagem = new PrintWriter (
                                              CanalDeTransmissaoDaMensagem.
                                              getOutputStream());

                SaidaDaMensagem.print (Mensagem.getText());

                SaidaDaMensagem.close ();
                SaidaDaMensagem = null;

                CanalDeTransmissaoDaMensagem.close ();
                CanalDeTransmissaoDaMensagem = null;
            }
            catch (UnknownHostException E)
            {
                System.err.println ("Servidor " + Endereco + " desconhecido.");
                System.err.println ();
            }
            catch (IOException E)
            {
                System.err.println ("Impossivel criar um canal de comunicacao.");
                System.err.println ();
            }
        }

        System.exit (0);
    }

    private void TrataClickNoBotaoCancela ()
    {
        System.exit (0);
    }

    public void mouseClicked (MouseEvent e)
    {}

    public void mousePressed (MouseEvent e)
    {}

    public void mouseReleased (MouseEvent e)
    {
        if (e.getComponent() == OK)
            TrataClickNoBotaoOK ();
        else if (e.getComponent() == Cancela)
            TrataClickNoBotaoCancela ();
    }

    public void mouseEntered (MouseEvent e)
    {}

    public void mouseExited (MouseEvent e)
    {}
}

class Envio
{
    public static void main (String Args [])
    {
        if (Args.length != 1)
        {
```

```
            System.err.println ("Uso correto: java Envio IPouNOME");
            return;
        }

        Janela janela = new Janela (Args [0]);
    }
}
```

Para compilar o programa de recepção de mensagens, daremos o seguinte comando:

C:\ExplsJava\Expl_18\Prg_Recepcao> javac Recepcao.java

O programa de recepção deve então ser executado nas máquinas que devem estar habilitadas a receber mensagens através do seguinte comando:

C:\ExplsJava\Expl_18\Prg_Recepcao> javaw Recepcao

Nenhuma saída será produzida pela execução do comando acima.

Para compilar o programa de envio de mensagens, daremos o seguinte comando:

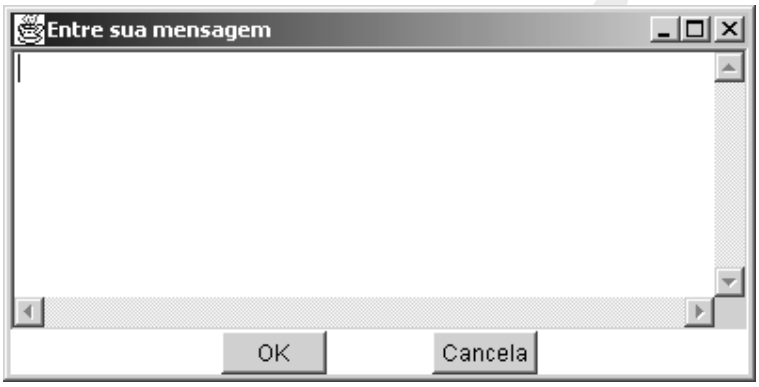
C:\ExplsJava\Expl_18\Prg_Envio> javac Envio.java

O programa de envio pode então ser executado em qualquer máquina que esteja em rede com máquinas nas quais foi executado o programa de recepção para mandar uma mensagem para qualquer uma destas máquinas. Isso pode ser conseguido através do seguinte comando:

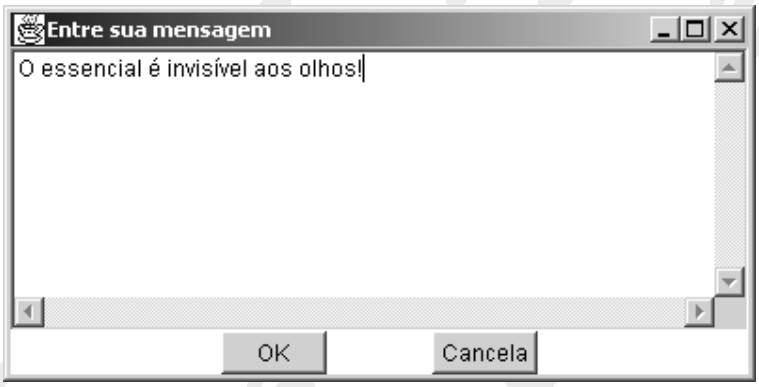
C:\ExplsJava\Expl_18\Prg_Envio> javaw Envio Cronos

Cronos é o nome máquina à qual se destina a mensagem e pode ser substituído por qualquer outro nome desde que, naturalmente, exista uma máquina com aquele nome em rede com a máquina que envia a mensagem e, ainda, desde que esta máquina esteja executando o programa de recepção. Se preferirmos, podemos, em vez de usar um nome de máquina, usar o endereço IP dela.

Veja abaixo a janela que é aberta pelo programa de envio.

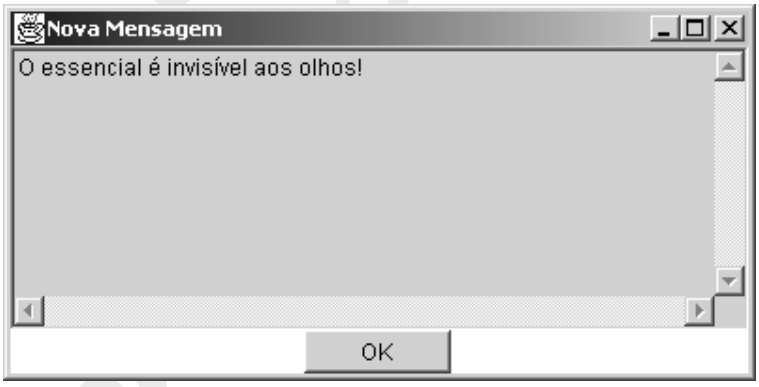


Basta entrar com o texto da mensagem na caixa de texto onde se encontra o cursor, como mostra a figura abaixo.



Para enviar a mensagem, é só acionar o botão OK. Para desistir do envio da mensagem, basta acionar o botão Cancela (ou o botão X).

Após o acionamento do botão OK, na tela da máquina receptora, será aberta a janela abaixo.



Nesta janela, o texto da mensagem pode ser lido, mas não pode ser alterado. Após sua leitura, basta acionar o botão OK e a janela desaparecerá da tela da máquina receptora.

Veja abaixo mais um exemplo de programa para ambiente distribuído, neste caso, um cliente SMTP. É interessante notar como o cliente SMTP segue as especificações do protocolo SMTP a fim de se comunicar com o servidor SMTP.

[C:\ExplsJava\Expl_19\ClienteSMTP\ExcecaoDeSMTP.java]

```
package ClienteSMTP;

public class ExcecaoDeSMTP extends RuntimeException
{
    public static final int SERVIDOR_DESCONHECIDO         =  0,
                            SERVIDOR_INCOMUNICAVEL        =  1,
                            CANAL_DE_ENTRADA_INDISPONIVEL =  2,
                            CANAL_DE_SAIDA_INDISPONIVEL   =  3,
                            INSUCESSO_NA_RECEPCAO         =  4,
                            SERVIDOR_INDISPONIVEL         =  5,
                            CLIENTE_RECUSADO              =  6,
                            REMETENTE_RECUSADO            =  7,
                            DESTINATARIO_RECUSADO         =  8,
                            SERVICO_INDISPONIVEL          =  9,
                            MENSAGEM_RECUSADA             = 10,
                            INSUCESSO_NA_DESPEDIDA        = 11,
                            INSUCESSO_NA_DESCONEXAO       = 12,
                            EXCECAO_INVALIDA              = 13;


    private static final int MENOR_ID = SERVIDOR_DESCONHECIDO,
                             MAIOR_ID = INSUCESSO_NA_DESCONEXAO;

    private static final String Mensagem [] =
    {
         "Servidor desconhecido",
         "Servidor incomunicavel",
         "Canal de entrada indisponivel",
         "Canal de saida indisponivel",
         "Insucesso na recepcao",
         "Servidor indisponivel",
         "Cliente recusado",
         "Remetente recusado",
         "Destinatario recusado",
         "Servico indisponivel",

         "Mensagem recusada",
         "Insucesso na despedida",
         "Insucesso na desconexao",
         "Indicacao de excecao invalida"
    };

    private int Tipo;

    private ExcecaoDeSMTP (int T)
    {
        super (ExcecaoDeSMTP.Mensagem [T]);
        this.Tipo = T;
    }
```

```
    public static void indique (int T) throws ExcecaoDeSMTP
    {
        if (T<ExcecaoDeSMTP.MENOR_ID || T>ExcecaoDeSMTP.MAIOR_ID)
            throw new ExcecaoDeSMTP (ExcecaoDeSMTP.EXCECAO_INVALIDA);

        throw new ExcecaoDeSMTP (T);
    }

    public int seuTipo ()
    {
        return this.Tipo;
    }
}
```

[C:\ExplsJava\Expl_19\ClienteSMTP\ClienteDeSMTP.java]

```
package ClienteSMTP;

import java.net.*;
import java.io.*;

public class ClienteSMTP
{
    private Socket         CanalDeComunicacao = null;
    private BufferedReader Entrada            = null;
    private PrintWriter    Saida              = null;

    public ClienteSMTP (String Servidor, int Porta) throws ExcecaoDeSMTP
    {
        try
        {
            CanalDeComunicacao = new Socket(Servidor, Porta);
        }
        catch (UnknownHostException E)
        {
            ExcecaoDeSMTP.indique (ExcecaoDeSMTP.SERVIDOR_DESCONHECIDO);
        }
        catch (IOException E)
        {
            ExcecaoDeSMTP.indique (ExcecaoDeSMTP.SERVIDOR_INCOMUNICAVEL);
        }

        try
        {
            Entrada = new BufferedReader (
                      new InputStreamReader (
                      CanalDeComunicacao.
                      getInputStream ()));
        }
        catch (IOException E)
        {
            ExcecaoDeSMTP.indique (ExcecaoDeSMTP.CANAL_DE_ENTRADA_INDISPONIVEL);
        }

        try
        {
            Saida = new PrintWriter (
                    new OutputStreamWriter (
                    CanalDeComunicacao.
                    getOutputStream()));
        }
        catch (IOException E)
        {
            ExcecaoDeSMTP.indique (ExcecaoDeSMTP.CANAL_DE_SAIDA_INDISPONIVEL);
```

```
        }
    }

    public void finalize () throws ExcecaoDeSMTP
    {
        try
        {
            Entrada.close();
            Entrada = null;

            Saida.close();
            Saida = null;

            CanalDeComunicacao.close();
            CanalDeComunicacao = null;
        }
        catch (IOException E)
        {
            ExcecaoDeSMTP.indique (ExcecaoDeSMTP.INSUCESSO_NA_DESCONEXAO);
        }
    }

    private int ObtenhaResposta () throws ExcecaoDeSMTP
    {
        int R = 0;

        try
        {
            R = Integer.parseInt(Entrada.readLine().substring(0,3));
        }
        catch (IOException E)
        {
            ExcecaoDeSMTP.indique (ExcecaoDeSMTP.INSUCESSO_NA_RECEPCAO);
        }

        return R;
    }

    public void EnvieMensagem (String NomeRemetente,
                               String E_mailRemetente,
                               String NomeDestinatario,
                               String E_mailDestinatario,
                               String Assunto,
                               String Mensagem)
                               throws ExcecaoDeSMTP
    {

            int Resposta;

            Resposta = this.ObtenhaResposta ();
            if (Resposta != 220)
                ExcecaoDeSMTP.indique (ExcecaoDeSMTP.SERVIDOR_INDISPONIVEL);

            String NomeDoMeuComputador = null;
            try
            {
                NomeDoMeuComputador = InetAddress.getLocalHost().getHostName();
            }
            catch (UnknownHostException E)
            {}

            Saida.println("HELO " + NomeDoMeuComputador);
            Saida.flush();

            Resposta = this.ObtenhaResposta ();
            if (Resposta != 250)
                ExcecaoDeSMTP.indique (ExcecaoDeSMTP.CLIENTE_RECUSADO);

            Saida.println("MAIL FROM: " + E_mailRemetente);
```

```
            Saida.flush();


            Resposta = this.ObtenhaResposta ();
            if (Resposta != 250)
                ExcecaoDeSMTP.indique (ExcecaoDeSMTP.REMETENTE_RECUSADO);

            Saida.println("RCPT TO: " + E_mailDestinatario);
            Saida.flush();

            Resposta = this.ObtenhaResposta (); // 250/251 DestinatarioRecusado
            if (Resposta != 250 && Resposta != 251)
                ExcecaoDeSMTP.indique (ExcecaoDeSMTP.DESTINATARIO_RECUSADO);

            Saida.println("DATA");
            Saida.flush();

            Resposta = this.ObtenhaResposta ();
            if (Resposta != 354)
                ExcecaoDeSMTP.indique (ExcecaoDeSMTP.SERVICO_INDISPONIVEL);

            Saida.println("FROM: " + NomeRemetente + " <" + E_mailRemetente + ">");
            Saida.flush();
            Saida.println("TO: " + NomeDestinatario + " <" + E_mailDestinatario + ">");
            Saida.flush();
            Saida.println("SUBJECT: " + Assunto);
            Saida.flush();
            Saida.println(Mensagem);
            Saida.flush();
            Saida.println(".");
            Saida.flush();
            Resposta = this.ObtenhaResposta ();
            if (Resposta != 250)
                ExcecaoDeSMTP.indique (ExcecaoDeSMTP.MENSAGEM_RECUSADA);

            Saida.println("QUIT");
            Saida.flush();
            Resposta = this.ObtenhaResposta ();
            if (Resposta != 221)
                ExcecaoDeSMTP.indique (ExcecaoDeSMTP.INSUCESSO_NA_DESPEDIDA);
    }
}
```

[C:\ExplsJava\Expl_19\E_mail.java]

```
import java.awt.*;
import java.awt.event.*;
import ClienteSMTP.*;

class Janela extends    Frame
             implements WindowListener,
                        MouseListener
{
    private Button OK     = new Button ("OK"),
                   Cancela = new Button ("Cancela");

    private TextField NomeDoServidor       = new TextField (),
                      NomeDoRemetente      = new TextField (),
                      E_mailDoRemetente    = new TextField (),
                      NomeDoDestinatario   = new TextField (),
                      E_mailDoDestinatario = new TextField (),
                      TextoDoAssunto       = new TextField ();

    private TextArea TextoDaMensagem = new TextArea ();
```

```
    public Janela ()
    {
        this.setTitle ("Correio Eletrônico");
        this.setSize  (800, 600);

        Panel Servidor = new Panel ();
        GridLayout LayoutServidor = new GridLayout (2,1);
        Servidor.setLayout (LayoutServidor);

        Servidor.add (new Label ("Servidor SMTP:"));
        Servidor.add (this.NomeDoServidor);

        Panel Remetente = new Panel ();
        GridLayout LayoutRemetente = new GridLayout (2,2);
        Remetente.setLayout (LayoutRemetente);

        Remetente.add (new Label ("Nome do Remetente:"));
        Remetente.add (new Label ("E-mail do Remetente:"));
        Remetente.add (this.NomeDoRemetente);
        Remetente.add (this.E_mailDoRemetente);

        Panel Destinatario = new Panel ();
        GridLayout LayoutDestinatario = new GridLayout (2,2);
        Destinatario.setLayout (LayoutDestinatario);

        Destinatario.add (new Label ("Nome do Destinatario:"));
        Destinatario.add (new Label ("E-mail do Destinatario:"));
        Destinatario.add (this.NomeDoDestinatario);
        Destinatario.add (this.E_mailDoDestinatario);

        Panel Assunto = new Panel ();
        GridLayout LayoutAssunto = new GridLayout (2,1);
        Assunto.setLayout (LayoutAssunto);

        Assunto.add (new Label ("Assunto:"));
        Assunto.add (this.TextoDoAssunto);

        Panel Infos = new Panel ();
        GridLayout LayoutInfos = new GridLayout (5,1);
        Infos.setLayout (LayoutInfos);

        Infos.add (Servidor);
        Infos.add (Remetente);
        Infos.add (Destinatario);
        Infos.add (Assunto);
        Infos.add (new Label ("Texto da Mensagem:"));

        BorderLayout LayoutJanela = new BorderLayout ();
        this.setLayout (LayoutJanela);

        this.add ("North", Infos);

        this.add ("Center", this.TextoDaMensagem);

        Panel Botoes = new Panel ();

        GridLayout LayoutBotoes = new GridLayout (1,7);
        Botoes.setLayout (LayoutBotoes);

        Botoes.add (new Label ());
        Botoes.add (new Label ());
        Botoes.add (this.OK);
```

```
        Botoes.add (new Label ());
        Botoes.add (this.Cancela);
        Botoes.add (new Label ());
        Botoes.add (new Label ());

        this.add ("South", Botoes);

        this        .addWindowListener (this);
        this.OK     .addMouseListener  (this);
        this.Cancela.addMouseListener  (this);

        this.show ();
    }



    public void windowClosing (WindowEvent e)
    {
        System.exit (0);
    }

    public void windowOpened (WindowEvent e)
    {}

    public void windowClosed (WindowEvent e)
    {}

    public void windowActivated (WindowEvent e)
    {}

    public void windowDeactivated (WindowEvent e)
    {}

    public void windowIconified (WindowEvent e)
    {}

    public void windowDeiconified (WindowEvent e)
    {}



    private void TrataClickNoBotaoOK ()
    {
        if (!this.TextoDaMensagem.getText().equals(""))
        {
            ClienteSMTP CliSMTP = new ClienteSMTP (NomeDoServidor.getText(),
25);

            CliSMTP.EnvieMensagem (this.NomeDoRemetente     .getText(),
                                   this.E_mailDoRemetente   .getText(),
                                   this.NomeDoDestinatario  .getText(),
                                   this.E_mailDoDestinatario.getText(),
                                   this.TextoDoAssunto      .getText(),
                                   this.TextoDaMensagem     .getText());
        }

        System.exit (0);
    }

    private void TrataClickNoBotaoCancela ()
    {
```

```
        System.exit (0);
    }

    public void mouseReleased (MouseEvent e)
    {}

    public void mousePressed (MouseEvent e)
    {}

    public void mouseClicked (MouseEvent e)
    {
        if (e.getComponent() == this.OK)
            this.TrataClickNoBotaoOK ();
        else if (e.getComponent() == this.Cancela)
            this.TrataClickNoBotaoCancela ();
    }

    public void mouseEntered (MouseEvent e)
    {}

    public void mouseExited (MouseEvent e)
    {}
}

class E_mail
{
    public static void main (String Args [])
    {
        Janela janela = new Janela ();
    }
}
```

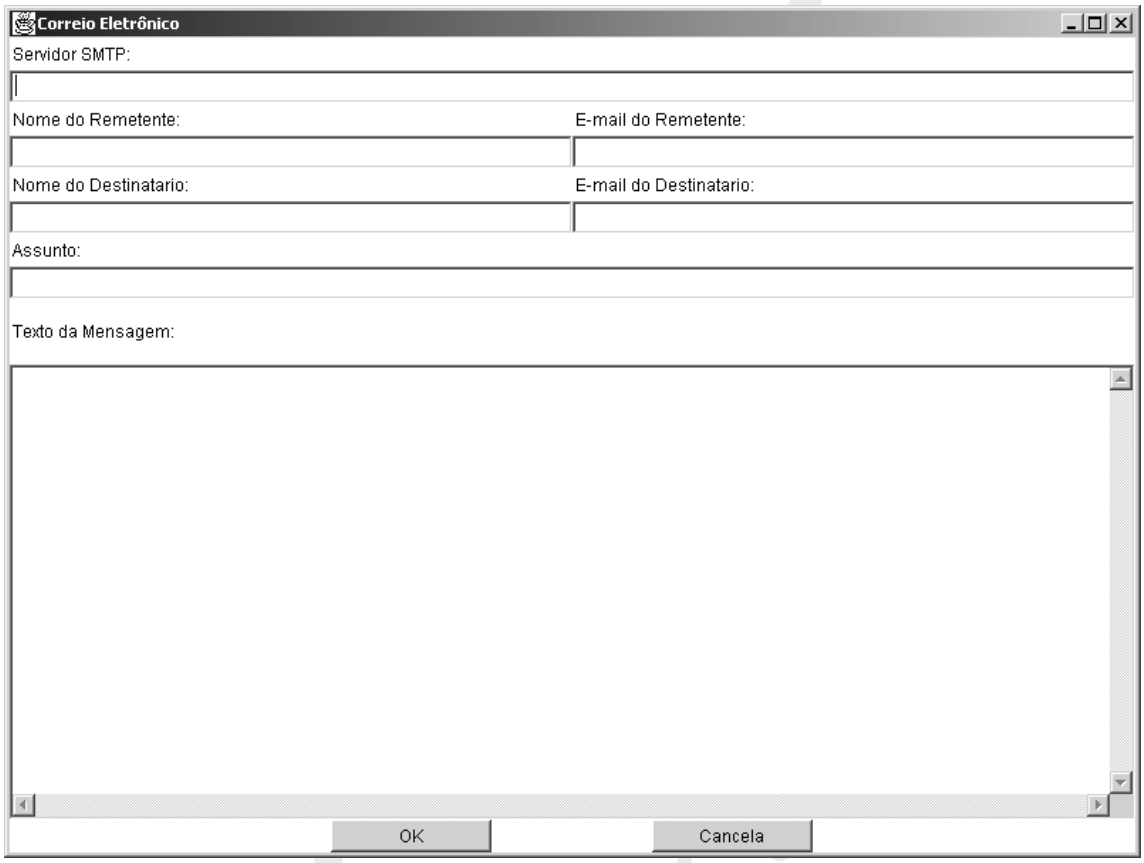
Para compilar o programa cliente SMTP, daremos o seguinte comando:

C:\ExplsJava\Expl_19> javac -classpath . E_mail.java

O programa de envio pode então ser executado em qualquer máquina que esteja em rede com máquinas servidoras de SMTP. Isso pode ser conseguido através do seguinte comando:

C:\ExplsJava\Expl_19> javaw -classpath . E_mail

Veja abaixo a janela que é aberta pelo programa cliente de SMTP:



Para redigir sua mensagem, basta preencher cada um dos campos (servidor SMTP, nome do remetente, e-mail do remetente, nome do destinatário, e-mail do destinatário, assunto e texto da mensagem) da janela acima.

Para enviar a mensagem, é só acionar o botão OK. Para desistir do envio da mensagem, basta acionar o botão Cancela (ou o botão X).

Classes Utilitárias

A Classe java.lang.System

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Campos:**

- **static PrintStream err:**
  Representa uma instância da classe java.io.PrintStream associada ao dispositivo padrão de saída de erros;
- **static InputStream in:**
  Representa uma instância da classe java.io.InputStream associada ao dispositivo padrão de entrada;
- **static PrintStream out:**
  Representa uma instância da classe java.io.PrintStream associada ao dispositivo padrão de saída;

- **Métodos:**
  - **static void arraycopy (Object O, int PO, Object D, int PD, int Q):}**
    Copia Q elementos, a partir da posição PO, do vetor O para o vetor D, a partir da posição PD;
  - **static long currentTimeMillis ():**
    Resulta na quantidade de milissegundos que se passaram desde a 00:00:00h do dia 01/jan/1970;
  - **static void exit (int T):**
    Termina a Java Virtual Machine em execução no momento, voltanto para o sistema operacional a condição de término T (0 indica término normal e outros valores término anormal);
  - **static Properties getProperties ():**
    Retorna as propriedades do sistema em uma instância da classe Properties;
  - **static String getProperty (String N):**
    Retorna o valor da propriedade do sistema de nome N;
  - **static String getProperty (String N, String D):**
    Retorna o valor da propriedade do sistema de nome N (no caso dela não ser encontrada, retorna o valor default D);

- **static void gc ():**
  Força o sistema a fazer coleta de lixo (garbage collection);
- **static void loadLibrary (String N):**
  Carrega a DLL de nome N;
- **static void runFinalization ():**
  Executa os métodos static void finalize () de todos os objetos com finalização pendente;
- **static void setErr (PrintStream E):**
  Associa E ao dispositivo padrão de saída de erro;
- **static void setIn (InputStream I):**
  Associa I ao dispositivo padrão de entrada;
- **static void setOut (PrintStream O):**
  Associa O ao dispositivo padrão de saída;
- **static void setProperties (Properties P):**
  Ajusta as propriedades do sistema a partir de uma instância da classe Properties;
- **static void setProperty (String N, String V):**
  Ajusta a propriedade do sistema de nome N fazendo-a assumir o valor V.

A Classe java.lang.Runtime

Toda aplicação Java em execução tem uma instância da class Runtime que permite que a aplicação se comunique com o ambiente no qual executa. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Métodos:**
  - **void exit (int T):**
    Termina a Java Virtual Machine em execução no momento, voltanto para o sistema operacional a condição de término T (0 indica término normal e outros valores término anormal);

- **long freeMemory ():**
  Retorna a quantidade de memória livre disponível para o sistema;
- **void gc ():**
  Força o sistema a fazer coleta de lixo (garbage collection);
- **static Runtime getRuntime ():**
  Retorna uma instância da classe Runtime que representa o Runtime System Environment;
- **static void loadLibrary (String N):**
  Carrega a DLL de nome N;
- **static void runFinalization ():**
  Executa os métodos static void finalize () de todos os objetos com finalização pendente;
- **long totalMemory ():**
  Retorna a quantidade total de memória do sistema;
- **void traceInstructions (boolean T):**
  Habilita ou desabilita, conforme o valor de T, seguir passo a passo a execução das instruções;
- **void traceMethodCalls (boolean T):**
  Habilita ou desabilita, conforme o valor de T, seguir passo a passo a execução de métodos.

O Pacote java.util

A Classe java.util.Date

Esta classe é utilizada para representar datas e horários de uma maneira que independa do sistema. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

- **Date ():**

  Constroi uma instância da classe Date com a data e horário atual do sistema;

- **Date (long L):**

  Constroi uma instância da classe Date a partir de um valor UTC;

- **Métodos:**

  - **boolean after (Date D):**

    Verifica se este Date é posterior a D;

  - **boolean before (Date D):**

    Verifica se este Date é anterior a D;

  - **int compareTo (Date D):**

    Compara este Date com D; retorna um número negativo no caso do primeiro ser menor que o segundo; retorna zero, no caso de ambos serem iguais; e retorna um número positivo no caso do primeiro ser maior que o segundo;

  - **long getTime ():**

    Resulta na quantidade de milissegundos que se passaram desde a 00:00:00h do dia 01/jan/1970 até o instante representado por este Date;

  - **void setTime (long L):**

    Ajusta este Date para representar o ponto no tempo que fica L milisegundos após 00:00:00h do dia 01/jan/1970.

A Classe java.util.Random

Esta classe implementa um gerador de números pseudo-aleatórios e pode ser usada para gerar uma seqüência de números aparentemente aleatórios. Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

  - **Random ():**

    Constroi uma nova instância da classe Random (um novo gerador de números aleatórios);

- **Random (long S):**

  Constroi uma nova instância da classe Random (um novo gerador de números aleatórios) baseado na semente S;

- **Métodos:**

  - **boolean nextBoolean ():**

    Retorna o próximo valor lógico, pseudo-aleatório e de distribuição uniforme, da seqüência deste gerador de números aleatórios;

  - **void nextBytes (byte V []):**

    Gera bytes aleatórios e os coloca no vetor V;

  - **double nextDouble ():**

    Retorna o próximo valor real de dupla precisão, pseudo-aleatório e de distribuição uniforme, da seqüência deste gerador de números aleatórios;

  - **double nextFloat ():**

    Retorna o próximo valor real, pseudo-aleatório e de distribuição uniforme, da seqüência deste gerador de números aleatórios;

  - **double nextGaussian ():**

    Retorna o próximo valor real de dupla precisão, pseudo-aleatório e de distribuição de Gauss, da seqüência deste gerador de números aleatórios;

  - **int nextInt ():**

    Retorna o próximo valor inteiro, pseudo-aleatório e de distribuição uniforme, da seqüência deste gerador de números aleatórios;

  - **int nextInt (int N):**

    Retorna o próximo valor inteiro R ($0 \leq R < N$), pseudo-aleatório e de distribuição uniforme, da seqüência deste gerador de números aleatórios;

  - **long nextLong ():**

    Retorna o próximo valor inteiro longo, pseudo-aleatório e de distribuição uniforme, da seqüência deste gerador de números aleatórios;

- **void setSeed (long S):**

  Torna S a nova semente do gerador de números pseudo-aleatórios.

A Classe java.util.Stack

Esta classe implementa uma pilha (LIFO) de objetos.

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**

  - **Stack ():**

    Constroi uma instância vazia da classe Stack;

- **Métodos:**

  - **boolean empty ():**

    Verifica se esta Stack está vazia;

  - **Object peek ():**

    Retorna o objeto que se encontra no topo desta Stack sem desempilhá-lo;

  - **Object pop ():**

    Desempilha e retorna o Object que se encontra no topo da pilha;

  - **void push (Object O):**

    Empilha o Object O;

  - **int search (Object O):**

    Procura o Object O nesta Stack e retorna sua ordem desempilhamento (1 para o primeiro a ser desempilhado, 2 para o segundo a ser desempilhado, etc). Retorna –1 no caso de não encontrá-lo nesta Stack.

A Classe java.util.Hashtable

Derivada da classe Dictionary, a classe Hashtable implementa uma tabela de hash. Tabelas de hash são estruturas de dados projetadas para rapidamente localizar e recuperar informações dada uma chave. Tanto a chave, quanto a informação a ela associada, podem ser objetos de

qualquer tipo, mas é obrigatório que o objeto usado como chave implemente os métodos hashCode e equals.

Instâncias desta classe têm dois parâmetros que afetam seu desempenho: (1) capacidade inicial de armazenamento (quantidade de posições que existem inicialmente em uma Hashtable); e (2) taxa máxima de ocupação (valor percentual que estabelece um limite à porcentagem de posições ocupadas em uma Hashtable).

À medida que a taxa de ocupação de uma Hashtable aumenta, a eficiência do algorítmo de Hash diminui. Assim, podemos facilmente concluir que nunca é bom trabalhar com uma Hashtable cheia ou quase cheia. Por isso, toda Hashtable tem uma taxa máxima de ocupação. Toda vez que a taxa de ocupação de uma Hashtable atinge a taxa máxima de ocupação, a Hashtable cresce em tamanho, melhorando assim a referida taxa de ocupação.

Veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **Hashtable ():**
    Constroi uma instância vazia da classe Hashtable com valores padrão para capacidade inicial de armazenamento e taxa máxima de ocupação;
  - **Hashtable (int C):**
    Constroi uma instância vazia da classe Hashtable com capacidade inicial de armazenamento igual a C e com taxa máxima de ocupação padrão;
  - **Hashtable (int C, float T):**
    Constroi uma instância vazia da classe Hashtable com capacidade inicial de armazenamento igual a C e taxa máxima de ocupação igual a T;
- **Métodos:**
  - **void clear ():**
    Remove todos os elementos desta Hashtable;

- **boolean contains (Object V):**
  Verifica se esta Hashtable contém o Object V (não como chave, como valor associado a uma chave);
- **boolean containsKey (Object C):**
  Verifica se esta Hashtable contém o Object C (como chave);
- **boolean containsValue (Object V):**
  Verifica se esta Hashtable contém o Object V (não como chave, como valor associado a uma chave);
- **Object get (Object C):**
  Retorna o valor associado à chave C nesta Hashtable;
- **boolean isEmpty ():**
  Verifica se esta Hashtable está vazia;
- **void put (Object C, Object V):**
  Armazena um par da forma (chave, valor) nesta Hashtable (C é a chave e V é o valor);
- **void remove (Object C):**
  Remove desta Hashtable o elemento cuja chave é C;
- **int size ():**
  Retorna a quantidade de pares (chave, valor) se encontram armazenados nesta Hashtable.

A Classe java.util.Properties

Derivada da classe Hashtable, esta classe pode ser vista como uma tabela de hash persistente, i.e., que pode ser armazenada e recuperada. Além das estruturas e do comportamento herdado da classe java.util.Hashtable, veja abaixo a interface que a classe especifica para comunicação com ela própria e com suas instâncias:

- **Construtores:**
  - **Properties ():**
    Constroi uma instância vazia da classe Properties;

- **Properties (Properties D):**
  Constroi uma instância vazia da classe Properties tendo D como lista de propriedades default (lista de propriedades onde deve-se procurar no caso de uma propriedade ser procurada e não encontrada na lista de propriedades ora criada);

- **Métodos:**
  - **String getProperty (String C):**
    Retorna a propriedade associada à chave C nesta Properties;
  - **String getProperty (String C, String D):**
    Retorna a propriedade associada à chave C nesta Properties (retorna D no caso da chave C não ser encontrada);
  - **void list (PrintStream PS):**
    Escreve esta Properties em PS;
  - **void list (PrintWriter PW):**
    Escreve esta Properties em PW;
  - **void load (InputStream IS):**
    Lê esta Properties de IS;
  - **void setProperty (String P, String V):**
    Armazena um par da forma (propriedade, valor) nesta Properties (P é a propriedade e V é o valor);
  - **void Store (OutputStream OS, String C):**
    Escreve a lista de propriedades no *stream* OS (o String C deve ser entendido como um comentário a cerca da lista de propriedades esclarecendo sua natureza).

Métodos Nativos

Métodos nativos são métodos escritos em linguagem C. Podemos introduzir um método nativo em uma classe qualificando-o com o modificador native. Métodos nativos não têm corpo (têm apenas o cabeçalho seguido por um ponto e vírgula (;)).

Uma vez que não são muito recomendáveis, tais métodos não serão abordados com maiores detalhes neste texto.

Anexo I

Hierarquia de Classes

- class java.lang.Object
  - class javax.swing.AbstractAction (implementa javax.swing.Action, java.lang.Cloneable, java.io.Serializable)
    - class javax.swing.plaf.basic.BasicDesktopPaneUI.CloseAction
    - class javax.swing.plaf.basic.BasicDesktopPaneUI.MaximizeAction
    - class javax.swing.plaf.basic.BasicDesktopPaneUI.MinimizeAction
    - class javax.swing.plaf.basic.BasicDesktopPaneUI.NavigateAction
    - class javax.swing.plaf.basic.BasicFileChooserUI.ApproveSelectionAction
    - class javax.swing.plaf.basic.BasicFileChooserUI.CancelSelectionAction
    - class javax.swing.plaf.basic.BasicFileChooserUI.ChangeToParentDirectoryAction
    - class javax.swing.plaf.basic.BasicFileChooserUI.GoHomeAction
    - class javax.swing.plaf.basic.BasicFileChooserUI.NewFolderAction
    - class javax.swing.plaf.basic.BasicFileChooserUI.UpdateAction
    - class javax.swing.plaf.basic.BasicInternalFrameTitlePane.CloseAction
    - class javax.swing.plaf.basic.BasicInternalFrameTitlePane.IconifyAction
    - class javax.swing.plaf.basic.BasicInternalFrameTitlePane.MaximizeAction
    - class javax.swing.plaf.basic.BasicInternalFrameTitlePane.MoveAction
    - class javax.swing.plaf.basic.BasicInternalFrameTitlePane.RestoreAction
    - class javax.swing.plaf.basic.BasicInternalFrameTitlePane.SizeAction
    - class javax.swing.plaf.basic.BasicSliderUI.ActionScroller
    - class javax.swing.plaf.basic.BasicTreeUI.TreeCancelEditingAction
    - class javax.swing.plaf.basic.BasicTreeUI.TreeHomeAction
    - class javax.swing.plaf.basic.BasicTreeUI.TreeIncrementAction
    - class javax.swing.plaf.basic.BasicTreeUI.TreePageAction
    - class javax.swing.plaf.basic.BasicTreeUI.TreeToggleAction
    - class javax.swing.plaf.basic.BasicTreeUI.TreeTraverseAction
    - class javax.swing.plaf.metal.MetalFileChooserUI.DirectoryComboBoxAction
    - class javax.swing.text.TextAction
      - class javax.swing.text.DefaultEditorKit.BeepAction
      - class javax.swing.text.DefaultEditorKit.CopyAction
      - class javax.swing.text.DefaultEditorKit.CutAction
      - class javax.swing.text.DefaultEditorKit.DefaultKeyTypedAction
      - class javax.swing.text.DefaultEditorKit.InsertBreakAction
      - class javax.swing.text.DefaultEditorKit.InsertContentAction
      - class javax.swing.text.DefaultEditorKit.InsertTabAction
      - class javax.swing.text.DefaultEditorKit.PasteAction
      - class javax.swing.text.StyledEditorKit.StyledTextAction
        - class javax.swing.text.html.HTMLEditorKit.HTMLTextAction
          - class javax.swing.text.html.HTMLEditorKit.InsertHTMLTextAction
        - class javax.swing.text.StyledEditorKit.AlignmentAction
        - class javax.swing.text.StyledEditorKit.BoldAction
        - class javax.swing.text.StyledEditorKit.FontFamilyAction
        - class javax.swing.text.StyledEditorKit.FontSizeAction
        - class javax.swing.text.StyledEditorKit.ForegroundAction
        - class javax.swing.text.StyledEditorKit.ItalicAction

- class javax.swing.text.StyledEditorKit.UnderlineAction
- class javax.swing.border.AbstractBorder (implementa javax.swing.border.Border, java.io.Serializable)
  - class javax.swing.plaf.basic.BasicBorders.ButtonBorder (implementa javax.swing.plaf.UIResource)
    - class javax.swing.plaf.basic.BasicBorders.RadioButtonBorder
    - class javax.swing.plaf.basic.BasicBorders.ToggleButtonBorder
  - class javax.swing.plaf.basic.BasicBorders.FieldBorder (implementa javax.swing.plaf.UIResource)
  - class javax.swing.plaf.basic.BasicBorders.MarginBorder (implementa javax.swing.plaf.UIResource)
  - class javax.swing.plaf.basic.BasicBorders.MenuBarBorder (implementa javax.swing.plaf.UIResource)
  - class javax.swing.border.BevelBorder
    - class javax.swing.plaf.BorderUIResource.BevelBorderUIResource (implementa javax.swing.plaf.UIResource)
    - class javax.swing.border.SoftBevelBorder
  - class javax.swing.border.CompoundBorder
    - class javax.swing.plaf.BorderUIResource.CompoundBorderUIResource (implementa javax.swing.plaf.UIResource)
  - class javax.swing.border.EmptyBorder (implementa java.io.Serializable)
    - class javax.swing.plaf.BorderUIResource.EmptyBorderUIResource (implementa javax.swing.plaf.UIResource)
    - class javax.swing.border.MatteBorder
      - class javax.swing.plaf.BorderUIResource.MatteBorderUIResource (implementa javax.swing.plaf.UIResource)
  - class javax.swing.border.EtchedBorder
    - class javax.swing.plaf.BorderUIResource.EtchedBorderUIResource (implementa javax.swing.plaf.UIResource)
  - class javax.swing.border.LineBorder
    - class javax.swing.plaf.BorderUIResource.LineBorderUIResource (implementa javax.swing.plaf.UIResource)
  - class javax.swing.plaf.metal.MetalBorders.ButtonBorder (implementa javax.swing.plaf.UIResource)
    - class javax.swing.plaf.metal.MetalBorders.RolloverButtonBorder
  - class javax.swing.plaf.metal.MetalBorders.Flush3DBorder (implementa javax.swing.plaf.UIResource)
    - class javax.swing.plaf.metal.MetalBorders.TextFieldBorder
  - class javax.swing.plaf.metal.MetalBorders.InternalFrameBorder (implementa javax.swing.plaf.UIResource)
  - class javax.swing.plaf.metal.MetalBorders.MenuBarBorder (implementa javax.swing.plaf.UIResource)
  - class javax.swing.plaf.metal.MetalBorders.MenuItemBorder (implementa javax.swing.plaf.UIResource)
  - class javax.swing.plaf.metal.MetalBorders.PopupMenuBorder (implementa javax.swing.plaf.UIResource)
  - class javax.swing.plaf.metal.MetalBorders.ScrollPaneBorder (implementa javax.swing.plaf.UIResource)
  - class javax.swing.plaf.metal.MetalBorders.ToolBarBorder (implementa javax.swing.SwingConstants, javax.swing.plaf.UIResource)
  - class javax.swing.border.TitledBorder
    - class javax.swing.plaf.BorderUIResource.TitledBorderUIResource (implementa javax.swing.plaf.UIResource)
- class javax.swing.AbstractButton.ButtonChangeListener (implementa javax.swing.event.ChangeListener, java.io.Serializable)
- class java.util.AbstractCollection (implementa java.util.Collection)

- class java.util.AbstractList (implementa java.util.List)
  - class java.util.AbstractSequentialList
    - class java.util.LinkedList (implementa java.lang.Cloneable, java.util.List, java.io.Serializable)
  - class java.util.ArrayList (implementa java.lang.Cloneable, java.util.List, java.io.Serializable)
  - class java.util.Vector (implementa java.lang.Cloneable, java.util.List, java.io.Serializable)
    - class java.util.Stack
- class java.util.AbstractSet (implementa java.util.Set)
  - class java.util.HashSet (implementa java.lang.Cloneable, java.io.Serializable, java.util.Set)
  - class java.util.TreeSet (implementa java.lang.Cloneable, java.io.Serializable, java.util.SortedSet)

- class javax.swing.text.AbstractDocument (implementa javax.swing.text.Document, java.io.Serializable)
  - class javax.swing.text.DefaultStyledDocument (implementa javax.swing.text.StyledDocument)
    - class javax.swing.text.html.HTMLDocument
  - class javax.swing.text.PlainDocument
- class javax.swing.text.AbstractDocument.AbstractElement (implementa javax.swing.text.Element, javax.swing.text.MutableAttributeSet, java.io.Serializable, javax.swing.tree.TreeNode)
  - class javax.swing.text.AbstractDocument.BranchElement
    - class javax.swing.text.DefaultStyledDocument.SectionElement
    - class javax.swing.text.html.HTMLDocument.BlockElement
  - class javax.swing.text.AbstractDocument.LeafElement
    - class javax.swing.text.html.HTMLDocument.RunElement
- class javax.swing.tree.AbstractLayoutCache (implementa javax.swing.tree.RowMapper)
  - class javax.swing.tree.FixedHeightLayoutCache
  - class javax.swing.tree.VariableHeightLayoutCache
- class javax.swing.tree.AbstractLayoutCache.NodeDimensions
  - class javax.swing.plaf.basic.BasicTreeUI.NodeDimensionsHandler
- class javax.swing.AbstractListModel (implementa javax.swing.ListModel, java.io.Serializable)
  - class javax.swing.plaf.basic.BasicDirectoryModel (implementa java.beans.PropertyChangeListener)
  - class javax.swing.DefaultComboBoxModel (implementa javax.swing.MutableComboBoxModel, java.io.Serializable)
  - class javax.swing.DefaultListModel
  - class javax.swing.plaf.metal.MetalFileChooserUI.DirectoryComboBoxModel (implementa javax.swing.ComboBoxModel)
  - class javax.swing.plaf.metal.MetalFileChooserUI.FilterComboBoxModel (implementa javax.swing.ComboBoxModel, java.beans.PropertyChangeListener)
- class java.util.AbstractMap (implementa java.util.Map)
  - class java.util.HashMap (implementa java.lang.Cloneable, java.util.Map, java.io.Serializable)
  - class java.util.TreeMap (implementa java.lang.Cloneable, java.io.Serializable, java.util.SortedMap)
  - class java.util.WeakHashMap (implementa java.util.Map)
- class javax.swing.table.AbstractTableModel (implementa java.io.Serializable, javax.swing.table.TableModel)
  - class javax.swing.table.DefaultTableModel (implementa java.io.Serializable)
- class javax.swing.undo.AbstractUndoableEdit (implementa java.io.Serializable, javax.swing.undo.UndoableEdit)
  - class javax.swing.text.AbstractDocument.ElementEdit (implementa javax.swing.event.DocumentEvent.ElementChange)

- class javax.swing.undo.CompoundEdit
  - class javax.swing.text.AbstractDocument.DefaultDocumentEvent (implementa javax.swing.event.DocumentEvent)
  - class javax.swing.undo.UndoManager (implementa javax.swing.event.UndoableEditListener)
- class javax.swing.text.DefaultStyledDocument.AttributeUndoableEdit
- class javax.swing.undo.StateEdit

- class javax.swing.text.AbstractWriter
  - class javax.swing.text.html.HTMLWriter
  - class javax.swing.text.html.MinimalHTMLWriter
- class java.security.AccessControlContext
- class java.security.AccessController
- class javax.accessibility.AccessibleBundle
  - class javax.accessibility.AccessibleRole
  - class javax.accessibility.AccessibleState
- class javax.accessibility.AccessibleContext
  - class javax.swing.Box.AccessibleBox (implementa javax.accessibility.AccessibleComponent, java.io.Serializable)
  - class javax.swing.Box.Filler.AccessibleBoxFiller (implementa javax.accessibility.AccessibleComponent, java.io.Serializable)
  - class javax.swing.CellRendererPane.AccessibleCellRendererPane (implementa javax.accessibility.AccessibleComponent, java.io.Serializable)
  - class javax.swing.JApplet.AccessibleJApplet (implementa javax.accessibility.AccessibleComponent, java.io.Serializable)
  - class javax.swing.JComponent.AccessibleJComponent (implementa javax.accessibility.AccessibleComponent, java.io.Serializable)
    - class javax.swing.AbstractButton.AccessibleAbstractButton (implementa javax.accessibility.AccessibleAction, javax.accessibility.AccessibleValue)
      - class javax.swing.JButton.AccessibleJButton
      - class javax.swing.JMenuItem.AccessibleJMenuItem (implementa javax.swing.event.ChangeListener)
        - class javax.swing.JCheckBoxMenuItem.AccessibleJCheckBoxMenuItem
        - class javax.swing.JMenu.AccessibleJMenu (implementa javax.accessibility.AccessibleSelection)
        - class javax.swing.JRadioButtonMenuItem.AccessibleJRadioButtonMenuItem
      - class javax.swing.JToggleButton.AccessibleJToggleButton (implementa java.awt.event.ItemListener)
        - class javax.swing.JCheckBox.AccessibleJCheckBox
        - class javax.swing.JRadioButton.AccessibleJRadioButton
    - class javax.swing.JColorChooser.AccessibleJColorChooser
    - class javax.swing.JComboBox.AccessibleJComboBox (implementa javax.accessibility.AccessibleAction)
    - class javax.swing.JDesktopPane.AccessibleJDesktopPane
    - class javax.swing.JFileChooser.AccessibleJFileChooser
    - class javax.swing.JInternalFrame.AccessibleJInternalFrame (implementa javax.accessibility.AccessibleValue)
    - class javax.swing.JInternalFrame.JDesktopIcon.AccessibleJDesktopIcon (implementa javax.accessibility.AccessibleValue)
    - class javax.swing.JLabel.AccessibleJLabel
    - class javax.swing.JLayeredPane.AccessibleJLayeredPane

- class javax.swing.JList.AccessibleJList (implementa javax.accessibility.AccessibleSelection, javax.swing.event.ListDataListener, javax.swing.event.ListSelectionListener, java.beans.PropertyChangeListener)
- class javax.swing.JMenuBar.AccessibleJMenuBar (implementa javax.accessibility.AccessibleSelection)
- class javax.swing.JOptionPane.AccessibleJOptionPane
- class javax.swing.JPanel.AccessibleJPanel
- class javax.swing.JPopupMenu.AccessibleJPopupMenu
- class javax.swing.JProgressBar.AccessibleJProgressBar (implementa javax.accessibility.AccessibleValue)
- class javax.swing.JRootPane.AccessibleJRootPane
- class javax.swing.JScrollBar.AccessibleJScrollBar (implementa javax.accessibility.AccessibleValue)
- class javax.swing.JScrollPane.AccessibleJScrollPane (implementa javax.swing.event.ChangeListener)
- class javax.swing.JSeparator.AccessibleJSeparator
- class javax.swing.JSlider.AccessibleJSlider (implementa javax.accessibility.AccessibleValue)
- class javax.swing.JSplitPane.AccessibleJSplitPane (implementa javax.accessibility.AccessibleValue)
- class javax.swing.JTabbedPane.AccessibleJTabbedPane (implementa javax.accessibility.AccessibleSelection, javax.swing.event.ChangeListener)
- class javax.swing.JTable.AccessibleJTable (implementa javax.accessibility.AccessibleSelection, javax.swing.event.CellEditorListener, javax.swing.event.ListSelectionListener, java.beans.PropertyChangeListener, javax.swing.event.TableColumnModelListener, javax.swing.event.TableModelListener)
- class javax.swing.table.JTableHeader.AccessibleJTableHeader
- class javax.swing.text.JTextComponent.AccessibleJTextComponent (implementa javax.accessibility.AccessibleText, javax.swing.event.CaretListener, javax.swing.event.DocumentListener)
  - class javax.swing.JEditorPane.AccessibleJEditorPane
    - class javax.swing.JEditorPane.AccessibleJEditorPaneHTML
    - class javax.swing.JEditorPane.JEditorPaneAccessibleHypertextSupport (implementa javax.accessibility.AccessibleHypertext)
  - class javax.swing.JTextArea.AccessibleJTextArea
  - class javax.swing.JTextField.AccessibleJTextField
    - class javax.swing.JPasswordField.AccessibleJPasswordField
- class javax.swing.JToolBar.AccessibleJToolBar
- class javax.swing.JToolTip.AccessibleJToolTip
- class javax.swing.JTree.AccessibleJTree (implementa javax.accessibility.AccessibleSelection, javax.swing.event.TreeExpansionListener, javax.swing.event.TreeModelListener, javax.swing.event.TreeSelectionListener)
- class javax.swing.JViewport.AccessibleJViewport

- class javax.swing.JDialog.AccessibleJDialog (implementa javax.accessibility.AccessibleComponent, java.io.Serializable)
- class javax.swing.JFrame.AccessibleJFrame (implementa javax.accessibility.AccessibleComponent, java.io.Serializable)

        - class javax.swing.JList.AccessibleJList.AccessibleJListChild (implementa javax.accessibility.Accessible, javax.accessibility.AccessibleComponent)
        - class javax.swing.JTable.AccessibleJTable.AccessibleJTableCell (implementa javax.accessibility.Accessible, javax.accessibility.AccessibleComponent)
        - class javax.swing.table.JTableHeader.AccessibleJTableHeader.AccessibleJTableHeaderEntry (implementa javax.accessibility.Accessible, javax.accessibility.AccessibleComponent)
        - class javax.swing.JTree.AccessibleJTree.AccessibleJTreeNode (implementa javax.accessibility.Accessible, javax.accessibility.AccessibleAction, javax.accessibility.AccessibleComponent, javax.accessibility.AccessibleSelection)
        - class javax.swing.JWindow.AccessibleJWindow (implementa javax.accessibility.AccessibleComponent, java.io.Serializable)
- class javax.accessibility.AccessibleHyperlink (implementa javax.accessibility.AccessibleAction)
    - class javax.swing.JEditorPane.JEditorPaneAccessibleHypertextSupport.HTMLLink
- class java.lang.reflect.AccessibleObject
    - class java.lang.reflect.Constructor (implementa java.lang.reflect.Member)
    - class java.lang.reflect.Field (implementa java.lang.reflect.Member)
    - class java.lang.reflect.Method (implementa java.lang.reflect.Member)
- class javax.accessibility.AccessibleStateSet
- class java.rmi.activation.ActivationDesc (implementa java.io.Serializable)
- class java.rmi.activation.ActivationGroupDesc (implementa java.io.Serializable)
- class java.rmi.activation.ActivationGroupDesc.CommandEnvironment (implementa java.io.Serializable)
- class java.rmi.activation.ActivationGroupID (implementa java.io.Serializable)
- class java.rmi.activation.ActivationID (implementa java.io.Serializable)
- class java.util.zip.Adler32 (implementa java.util.zip.Checksum)
- class java.awt.geom.AffineTransform (implementa java.lang.Cloneable, java.io.Serializable)
- class java.awt.image.AffineTransformOp (implementa java.awt.image.BufferedImageOp, java.awt.image.RasterOp)
- class java.security.AlgorithmParameterGenerator
- class java.security.AlgorithmParameterGeneratorSpi
- class java.security.AlgorithmParameters
- class java.security.AlgorithmParametersSpi
- class java.awt.AlphaComposite (implementa java.awt.Composite)
- class org.omg.CosNaming.NamingContextPackage.AlreadyBoundHelper
- class org.omg.CosNaming.NamingContextPackage.AlreadyBoundHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.text.Annotation
- class org.omg.CORBA.Any (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CORBA.AnyHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.awt.geom.Area (implementa java.lang.Cloneable, java.awt.Shape)
- class java.lang.reflect.Array
- class java.util.Arrays
- class java.text.AttributedCharacterIterator.Attribute (implementa java.io.Serializable)
    - class java.awt.font.TextAttribute
- class java.text.AttributedString
- class javax.swing.text.html.parser.AttributeList (implementa javax.swing.text.html.parser.DTDConstants, java.io.Serializable)
- class java.util.jar.Attributes (implementa java.lang.Cloneable, java.util.Map)
- class java.util.jar.Attributes.Name
- class java.net.Authenticator
- class java.awt.AWTEventMulticaster (implementa java.awt.event.ActionListener, java.awt.event.AdjustmentListener, java.awt.event.ComponentListener, java.awt.event.ContainerListener, java.awt.event.FocusListener,

java.awt.event.InputMethodListener, java.awt.event.ItemListener, java.awt.event.KeyListener, java.awt.event.MouseListener, java.awt.event.MouseMotionListener, java.awt.event.TextListener, java.awt.event.WindowListener)

- class java.awt.image.BandCombineOp (implementa java.awt.image.RasterOp)
- class javax.swing.plaf.basic.BasicBorders
- class javax.swing.plaf.basic.BasicBorders.SplitPaneBorder (implementa javax.swing.border.Border, javax.swing.plaf.UIResource)
- class javax.swing.plaf.basic.BasicButtonListener (implementa javax.swing.event.ChangeListener, java.awt.event.FocusListener, java.awt.event.MouseListener, java.awt.event.MouseMotionListener, java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicColorChooserUI.PropertyHandler (implementa java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicComboBoxEditor (implementa javax.swing.ComboBoxEditor, java.awt.event.FocusListener)
    - class javax.swing.plaf.basic.BasicComboBoxEditor.UIResource (implementa javax.swing.plaf.UIResource)
    - class javax.swing.plaf.metal.MetalComboBoxEditor
        - class javax.swing.plaf.metal.MetalComboBoxEditor.UIResource (implementa javax.swing.plaf.UIResource)
- class javax.swing.plaf.basic.BasicComboBoxUI.ComboBoxLayoutManager (implementa java.awt.LayoutManager)
    - class javax.swing.plaf.metal.MetalComboBoxUI.MetalComboBoxLayoutManager
- class javax.swing.plaf.basic.BasicComboBoxUI.FocusHandler (implementa java.awt.event.FocusListener)
- class javax.swing.plaf.basic.BasicComboBoxUI.ItemHandler (implementa java.awt.event.ItemListener)
- class javax.swing.plaf.basic.BasicComboBoxUI.ListDataHandler (implementa javax.swing.event.ListDataListener)
- class javax.swing.plaf.basic.BasicComboBoxUI.PropertyChangeHandler (implementa java.beans.PropertyChangeListener)
    - class javax.swing.plaf.metal.MetalComboBoxUI.MetalPropertyChangeListener
- class javax.swing.plaf.basic.BasicComboPopup.ItemHandler (implementa java.awt.event.ItemListener)
- class javax.swing.plaf.basic.BasicComboPopup.ListDataHandler (implementa javax.swing.event.ListDataListener)
- class javax.swing.plaf.basic.BasicComboPopup.ListSelectionHandler (implementa javax.swing.event.ListSelectionListener)
- class javax.swing.plaf.basic.BasicComboPopup.PropertyChangeHandler (implementa java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicFileChooserUI.SelectionListener (implementa javax.swing.event.ListSelectionListener)
- class javax.swing.plaf.basic.BasicGraphicsUtils
- class javax.swing.plaf.basic.BasicIconFactory (implementa java.io.Serializable)
- class javax.swing.plaf.basic.BasicInternalFrameTitlePane.PropertyChangeHandler (implementa java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicInternalFrameTitlePane.TitlePaneLayout (implementa java.awt.LayoutManager)
- class javax.swing.plaf.basic.BasicInternalFrameUI.BasicInternalFrameListener (implementa javax.swing.event.InternalFrameListener)
- class javax.swing.plaf.basic.BasicInternalFrameUI.ComponentHandler (implementa java.awt.event.ComponentListener)
- class javax.swing.plaf.basic.BasicInternalFrameUI.GlassPaneDispatcher (implementa javax.swing.event.MouseInputListener)
- class javax.swing.plaf.basic.BasicInternalFrameUI.InternalFrameLayout (implementa java.awt.LayoutManager)

- class javax.swing.plaf.basic.BasicInternalFrameUI.InternalFramePropertyChangeListener (implementa java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicListUI.FocusHandler (implementa java.awt.event.FocusListener)
- class javax.swing.plaf.basic.BasicListUI.ListDataHandler (implementa javax.swing.event.ListDataListener)
- class javax.swing.plaf.basic.BasicListUI.ListSelectionHandler (implementa javax.swing.event.ListSelectionListener)
- class javax.swing.plaf.basic.BasicListUI.MouseInputHandler (implementa javax.swing.event.MouseInputListener)
- class javax.swing.plaf.basic.BasicListUI.PropertyChangeHandler (implementa java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicMenuItemUI.MouseInputHandler (implementa javax.swing.event.MouseInputListener)
- class javax.swing.plaf.basic.BasicMenuUI.ChangeHandler (implementa javax.swing.event.ChangeListener)
- class javax.swing.plaf.basic.BasicOptionPaneUI.ButtonActionListener (implementa java.awt.event.ActionListener)
- class javax.swing.plaf.basic.BasicOptionPaneUI.ButtonAreaLayout (implementa java.awt.LayoutManager)
- class javax.swing.plaf.basic.BasicOptionPaneUI.PropertyChangeHandler (implementa java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicProgressBarUI.ChangeHandler (implementa javax.swing.event.ChangeListener)
- class javax.swing.plaf.basic.BasicScrollBarUI.ModelListener (implementa javax.swing.event.ChangeListener)
- class javax.swing.plaf.basic.BasicScrollBarUI.PropertyChangeHandler (implementa java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicScrollBarUI.ScrollListener (implementa java.awt.event.ActionListener)
- class javax.swing.plaf.basic.BasicScrollPaneUI.HSBChangeListener (implementa javax.swing.event.ChangeListener)
- class javax.swing.plaf.basic.BasicScrollPaneUI.PropertyChangeHandler (implementa java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicScrollPaneUI.ViewportChangeHandler (implementa javax.swing.event.ChangeListener)
- class javax.swing.plaf.basic.BasicScrollPaneUI.VSBChangeListener (implementa javax.swing.event.ChangeListener)
- class javax.swing.plaf.basic.BasicSliderUI.ChangeHandler (implementa javax.swing.event.ChangeListener)
- class javax.swing.plaf.basic.BasicSliderUI.FocusHandler (implementa java.awt.event.FocusListener)
- class javax.swing.plaf.basic.BasicSliderUI.PropertyChangeHandler (implementa java.beans.PropertyChangeListener)
    - class javax.swing.plaf.metal.MetalSliderUI.MetalPropertyListener
- class javax.swing.plaf.basic.BasicSliderUI.ScrollListener (implementa java.awt.event.ActionListener)
- class javax.swing.plaf.basic.BasicSplitPaneDivider.DividerLayout (implementa java.awt.LayoutManager)
- class javax.swing.plaf.basic.BasicSplitPaneDivider.DragController
    - class javax.swing.plaf.basic.BasicSplitPaneDivider.VerticalDragController
- class javax.swing.plaf.basic.BasicSplitPaneUI.BasicHorizontalLayoutManager (implementa java.awt.LayoutManager2)
    - class javax.swing.plaf.basic.BasicSplitPaneUI.BasicVerticalLayoutManager
- class javax.swing.plaf.basic.BasicSplitPaneUI.KeyboardDownRightHandler (implementa java.awt.event.ActionListener)

- class javax.swing.plaf.basic.BasicSplitPaneUI.KeyboardEndHandler (implementa java.awt.event.ActionListener)
- class javax.swing.plaf.basic.BasicSplitPaneUI.KeyboardHomeHandler (implementa java.awt.event.ActionListener)
- class javax.swing.plaf.basic.BasicSplitPaneUI.KeyboardResizeToggleHandler (implementa java.awt.event.ActionListener)
- class javax.swing.plaf.basic.BasicSplitPaneUI.KeyboardUpLeftHandler (implementa java.awt.event.ActionListener)
- class javax.swing.plaf.basic.BasicSplitPaneUI.PropertyHandler (implementa java.beans.PropertyChangeListener)
- class java.awt.BasicStroke (implementa java.awt.Stroke)
- class javax.swing.plaf.basic.BasicTabbedPaneUI.PropertyChangeHandler (implementa java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicTabbedPaneUI.TabbedPaneLayout (implementa java.awt.LayoutManager)
    - class javax.swing.plaf.metal.MetalTabbedPaneUI.TabbedPaneLayout
- class javax.swing.plaf.basic.BasicTabbedPaneUI.TabSelectionHandler (implementa javax.swing.event.ChangeListener)
- class javax.swing.plaf.basic.BasicTableHeaderUI.MouseInputHandler (implementa javax.swing.event.MouseInputListener)
- class javax.swing.plaf.basic.BasicTableUI.FocusHandler (implementa java.awt.event.FocusListener)
- class javax.swing.plaf.basic.BasicTableUI.KeyHandler (implementa java.awt.event.KeyListener)
- class javax.swing.plaf.basic.BasicTableUI.MouseInputHandler (implementa javax.swing.event.MouseInputListener)
- class javax.swing.plaf.basic.BasicToolBarUI.DockingListener (implementa javax.swing.event.MouseInputListener)
    - class javax.swing.plaf.metal.MetalToolBarUI.MetalDockingListener
- class javax.swing.plaf.basic.BasicToolBarUI.PropertyListener (implementa java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicToolBarUI.ToolBarContListener (implementa java.awt.event.ContainerListener)
- class javax.swing.plaf.basic.BasicToolBarUI.ToolBarFocusListener (implementa java.awt.event.FocusListener)
- class javax.swing.plaf.basic.BasicTreeUI.CellEditorHandler (implementa javax.swing.event.CellEditorListener)
- class javax.swing.plaf.basic.BasicTreeUI.FocusHandler (implementa java.awt.event.FocusListener)
- class javax.swing.plaf.basic.BasicTreeUI.MouseInputHandler (implementa javax.swing.event.MouseInputListener)
- class javax.swing.plaf.basic.BasicTreeUI.PropertyChangeHandler (implementa java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicTreeUI.SelectionModelPropertyChangeHandler (implementa java.beans.PropertyChangeListener)
- class javax.swing.plaf.basic.BasicTreeUI.TreeExpansionHandler (implementa javax.swing.event.TreeExpansionListener)
- class javax.swing.plaf.basic.BasicTreeUI.TreeModelHandler (implementa javax.swing.event.TreeModelListener)
- class javax.swing.plaf.basic.BasicTreeUI.TreeSelectionHandler (implementa javax.swing.event.TreeSelectionListener)
- class java.beans.beancontext.BeanContextChildSupport (implementa java.beans.beancontext.BeanContextChild, java.beans.beancontext.BeanContextServicesListener, java.io.Serializable)

    - class java.beans.beancontext.BeanContextSupport (implementa java.beans.beancontext.BeanContext, java.beans.PropertyChangeListener, java.io.Serializable, java.beans.VetoableChangeListener)
        - class java.beans.beancontext.BeanContextServicesSupport (implementa java.beans.beancontext.BeanContextServices)
- class java.beans.beancontext.BeanContextServicesSupport.BCSSProxyServiceProvider (implementa java.beans.beancontext.BeanContextServiceProvider, java.beans.beancontext.BeanContextServiceRevokedListener)
- class java.beans.beancontext.BeanContextServicesSupport.BCSSServiceProvider (implementa java.io.Serializable)
- class java.beans.beancontext.BeanContextSupport.BCSChild (implementa java.io.Serializable)
    - class java.beans.beancontext.BeanContextServicesSupport.BCSSChild
- class java.beans.beancontext.BeanContextSupport.BCSIterator (implementa java.util.Iterator)
- class java.beans.Beans
- class org.omg.CosNaming.Binding (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CosNaming.BindingHelper
- class org.omg.CosNaming.BindingHolder (implementa org.omg.CORBA.portable.Streamable)
- class org.omg.CosNaming.BindingIteratorHelper
- class org.omg.CosNaming.BindingIteratorHolder (implementa org.omg.CORBA.portable.Streamable)
- class org.omg.CosNaming.BindingListHelper
- class org.omg.CosNaming.BindingListHolder (implementa org.omg.CORBA.portable.Streamable)
- class org.omg.CosNaming.BindingType (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CosNaming.BindingTypeHelper
- class org.omg.CosNaming.BindingTypeHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.util.BitSet (implementa java.lang.Cloneable, java.io.Serializable)
- class java.awt.print.Book (implementa java.awt.print.Pageable)
- class java.lang.Boolean (implementa java.io.Serializable)
- class org.omg.CORBA.BooleanHolder (implementa org.omg.CORBA.portable.Streamable)
- class javax.swing.BorderFactory
- class java.awt.BorderLayout (implementa java.awt.LayoutManager2, java.io.Serializable)
- class javax.swing.plaf.BorderUIResource (implementa javax.swing.border.Border, java.io.Serializable, javax.swing.plaf.UIResource)
- class javax.swing.BoxLayout (implementa java.awt.LayoutManager2, java.io.Serializable)
    - class javax.swing.plaf.basic.DefaultMenuLayout (implementa javax.swing.plaf.UIResource)
- class java.text.BreakIterator (implementa java.lang.Cloneable)
- class javax.swing.ButtonGroup (implementa java.io.Serializable)
- class org.omg.CORBA.ByteHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.util.Calendar (implementa java.lang.Cloneable, java.io.Serializable)
    - class java.util.GregorianCalendar
- class org.omg.CosNaming.NamingContextPackage.CannotProceedHelper
- class org.omg.CosNaming.NamingContextPackage.CannotProceedHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.awt.CardLayout (implementa java.awt.LayoutManager2, java.io.Serializable)
- class java.security.cert.Certificate
    - class java.security.cert.X509Certificate (implementa java.security.cert.X509Extension)
- class java.security.cert.CertificateFactory
- class java.security.cert.CertificateFactorySpi
- class java.lang.Character (implementa java.lang.Comparable, java.io.Serializable)
- class java.lang.Character.Subset
    - class java.lang.Character.UnicodeBlock
    - class java.awt.im.InputSubset

- class org.omg.CORBA.CharHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.awt.CheckboxGroup (implementa java.io.Serializable)
- class java.lang.Class (implementa java.io.Serializable)
- class java.lang.ClassLoader
    - class java.security.SecureClassLoader
        - class java.net.URLClassLoader
- class java.awt.datatransfer.Clipboard
- class java.security.CodeSource (implementa java.io.Serializable)
- class java.text.CollationElementIterator
- class java.text.CollationKey (implementa java.lang.Comparable)
- class java.text.Collator (implementa java.lang.Cloneable, java.util.Comparator)
    - class java.text.RuleBasedCollator
- class java.util.Collections
- class java.awt.Color (implementa java.awt.Paint, java.io.Serializable)
    - class javax.swing.plaf.ColorUIResource (implementa javax.swing.plaf.UIResource)
    - class java.awt.SystemColor (implementa java.io.Serializable)
- class javax.swing.colorchooser.ColorChooserComponentFactory
- class java.awt.image.ColorConvertOp (implementa java.awt.image.BufferedImageOp, java.awt.image.RasterOp)
- class java.awt.image.ColorModel (implementa java.awt.Transparency)
    - class java.awt.image.ComponentColorModel
    - class java.awt.image.IndexColorModel
    - class java.awt.image.PackedColorModel
        - class java.awt.image.DirectColorModel
- class java.awt.color.ColorSpace
    - class java.awt.color.ICC_ColorSpace
- class java.lang.Compiler
- class org.omg.CORBA.CompletionStatus (implementa org.omg.CORBA.portable.IDLEntity)
- class java.awt.Component (implementa java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - class javax.swing.Box.Filler (implementa javax.accessibility.Accessible)
    - class java.awt.Button
    - class java.awt.Canvas
    - class java.awt.Checkbox (implementa java.awt.ItemSelectable)
    - class java.awt.Choice (implementa java.awt.ItemSelectable)
    - class java.awt.Container
        - class javax.swing.plaf.basic.BasicSplitPaneDivider (implementa java.beans.PropertyChangeListener)
        - class javax.swing.Box (implementa javax.accessibility.Accessible)
        - class javax.swing.CellRendererPane (implementa javax.accessibility.Accessible)
        - class javax.swing.tree.DefaultTreeCellEditor.EditorContainer
        - class javax.swing.JComponent (implementa java.io.Serializable)
            - class javax.swing.AbstractButton (implementa java.awt.ItemSelectable, javax.swing.SwingConstants)
                - class javax.swing.JButton (implementa javax.accessibility.Accessible)
                    - class javax.swing.plaf.basic.BasicArrowButton (implementa javax.swing.SwingConstants)
                        - class javax.swing.plaf.metal.MetalScrollButton
                    - class javax.swing.plaf.metal.MetalComboBoxButton
                - class javax.swing.JMenuItem (implementa javax.accessibility.Accessible, javax.swing.MenuElement)

        - class javax.swing.JCheckBoxMenuItem (implementa javax.accessibility.Accessible, javax.swing.SwingConstants)
        - class javax.swing.JMenu (implementa javax.accessibility.Accessible, javax.swing.MenuElement)
        - class javax.swing.JRadioButtonMenuItem (implementa javax.accessibility.Accessible)
    - class javax.swing.JToggleButton (implementa javax.accessibility.Accessible)
        - class javax.swing.JCheckBox (implementa javax.accessibility.Accessible)
        - class javax.swing.JRadioButton (implementa javax.accessibility.Accessible)
- class javax.swing.plaf.basic.BasicInternalFrameTitlePane
- class javax.swing.JColorChooser (implementa javax.accessibility.Accessible)
- class javax.swing.JComboBox (implementa javax.accessibility.Accessible, java.awt.event.ActionListener, java.awt.ItemSelectable, javax.swing.event.ListDataListener)
- class javax.swing.JFileChooser (implementa javax.accessibility.Accessible)
- class javax.swing.JInternalFrame (implementa javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
- class javax.swing.JInternalFrame.JDesktopIcon (implementa javax.accessibility.Accessible)
- class javax.swing.JLabel (implementa javax.accessibility.Accessible, javax.swing.SwingConstants)
    - class javax.swing.plaf.basic.BasicComboBoxRenderer (implementa javax.swing.ListCellRenderer, java.io.Serializable)
        - class javax.swing.plaf.basic.BasicComboBoxRenderer.UIResource (implementa javax.swing.plaf.UIResource)
    - class javax.swing.DefaultListCellRenderer (implementa javax.swing.ListCellRenderer, java.io.Serializable)
        - class javax.swing.DefaultListCellRenderer.UIResource (implementa javax.swing.plaf.UIResource)
        - class javax.swing.plaf.metal.MetalFileChooserUI.FileRenderer
        - class javax.swing.plaf.metal.MetalFileChooserUI.FilterComboBoxRenderer
    - class javax.swing.table.DefaultTableCellRenderer (implementa java.io.Serializable, javax.swing.table.TableCellRenderer)
        - class javax.swing.table.DefaultTableCellRenderer.UIResource (implementa javax.swing.plaf.UIResource)
    - class javax.swing.tree.DefaultTreeCellRenderer (implementa javax.swing.tree.TreeCellRenderer)

- class javax.swing.JLayeredPane (implementa javax.accessibility.Accessible)
  - class javax.swing.JDesktopPane (implementa javax.accessibility.Accessible)
- class javax.swing.JList (implementa javax.accessibility.Accessible, javax.swing.Scrollable)
- class javax.swing.JMenuBar (implementa javax.accessibility.Accessible, javax.swing.MenuElement)
  - class javax.swing.plaf.basic.BasicInternalFrameTitlePane.SystemMenuBar
- class javax.swing.JOptionPane (implementa javax.accessibility.Accessible)
- class javax.swing.JPanel (implementa javax.accessibility.Accessible)
  - class javax.swing.colorchooser.AbstractColorChooserPanel
- class javax.swing.JPopupMenu (implementa javax.accessibility.Accessible, javax.swing.MenuElement)
  - class javax.swing.plaf.basic.BasicComboPopup (implementa javax.swing.plaf.basic.ComboPopup)
    - class javax.swing.plaf.metal.MetalComboBoxUI.MetalComboPopup
- class javax.swing.JProgressBar (implementa javax.accessibility.Accessible, javax.swing.SwingConstants)
- class javax.swing.JRootPane (implementa javax.accessibility.Accessible)
- class javax.swing.JScrollBar (implementa javax.accessibility.Accessible, java.awt.Adjustable)
  - class javax.swing.JScrollPane.ScrollBar (implementa javax.swing.plaf.UIResource)
- class javax.swing.JScrollPane (implementa javax.accessibility.Accessible, javax.swing.ScrollPaneConstants)
- class javax.swing.JSeparator (implementa javax.accessibility.Accessible, javax.swing.SwingConstants)
  - class javax.swing.JPopupMenu.Separator
  - class javax.swing.JToolBar.Separator
- class javax.swing.JSlider (implementa javax.accessibility.Accessible, javax.swing.SwingConstants)
- class javax.swing.JSplitPane (implementa javax.accessibility.Accessible)
- class javax.swing.JTabbedPane (implementa javax.accessibility.Accessible, java.io.Serializable, javax.swing.SwingConstants)
- class javax.swing.JTable (implementa javax.accessibility.Accessible, javax.swing.event.CellEditorListener, javax.swing.event.ListSelectionListener, javax.swing.Scrollable, javax.swing.event.TableColumnModelListener, javax.swing.event.TableModelListener)
- class javax.swing.table.JTableHeader (implementa javax.accessibility.Accessible, javax.swing.event.TableColumnModelListener)
- class javax.swing.text.JTextComponent (implementa javax.accessibility.Accessible, javax.swing.Scrollable)

- class javax.swing.JEditorPane
  - class javax.swing.JTextPane
- class javax.swing.JTextArea
- class javax.swing.JTextField (implementa javax.swing.SwingConstants)
  - class javax.swing.tree.DefaultTreeCellEditor.DefaultTextField
  - class javax.swing.JPasswordField
- class javax.swing.JToolBar (implementa javax.accessibility.Accessible, javax.swing.SwingConstants)
- class javax.swing.JToolTip (implementa javax.accessibility.Accessible)
- class javax.swing.JTree (implementa javax.accessibility.Accessible, javax.swing.Scrollable)
- class javax.swing.JViewport (implementa javax.accessibility.Accessible)

- class java.awt.Panel
  - class java.applet.Applet
    - class javax.swing.JApplet (implementa javax.accessibility.Accessible, javax.swing.RootPaneContainer)
- class java.awt.ScrollPane
- class java.awt.Window
  - class javax.swing.plaf.basic.BasicToolBarUI.DragWindow
  - class java.awt.Dialog
    - class java.awt.FileDialog
    - class javax.swing.JDialog (implementa javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
  - class java.awt.Frame (implementa java.awt.MenuContainer)
    - class javax.swing.JFrame (implementa javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
  - class javax.swing.JWindow (implementa javax.accessibility.Accessible, javax.swing.RootPaneContainer)

- class java.awt.Label
- class java.awt.List (implementa java.awt.ItemSelectable)
- class java.awt.Scrollbar (implementa java.awt.Adjustable)
- class java.awt.TextComponent
  - class java.awt.TextArea
  - class java.awt.TextField

- class java.awt.event.ComponentAdapter (implementa java.awt.event.ComponentListener)
  - class javax.swing.plaf.basic.BasicSliderUI.ComponentHandler
  - class javax.swing.plaf.basic.BasicTreeUI.ComponentHandler (implementa java.awt.event.ActionListener)
  - class javax.swing.JViewport.ViewListener (implementa java.io.Serializable)
- class java.awt.ComponentOrientation (implementa java.io.Serializable)
- class javax.swing.plaf.ComponentUI
  - class javax.swing.plaf.ButtonUI
    - class javax.swing.plaf.basic.BasicButtonUI
      - class javax.swing.plaf.basic.BasicToggleButtonUI
        - class javax.swing.plaf.basic.BasicRadioButtonUI
          - class javax.swing.plaf.basic.BasicCheckBoxUI

- class javax.swing.plaf.metal.MetalRadioButtonUI
  - class javax.swing.plaf.metal.MetalCheckBoxUI
- class javax.swing.plaf.metal.MetalToggleButtonUI
- class javax.swing.plaf.metal.MetalButtonUI
- class javax.swing.plaf.MenuItemUI
  - class javax.swing.plaf.basic.BasicMenuItemUI
    - class javax.swing.plaf.basic.BasicCheckBoxMenuItemUI
    - class javax.swing.plaf.basic.BasicMenuUI
    - class javax.swing.plaf.basic.BasicRadioButtonMenuItemUI
  - class javax.swing.plaf.multi.MultiMenuItemUI
- class javax.swing.plaf.multi.MultiButtonUI

- class javax.swing.plaf.ColorChooserUI
  - class javax.swing.plaf.basic.BasicColorChooserUI
  - class javax.swing.plaf.multi.MultiColorChooserUI
- class javax.swing.plaf.ComboBoxUI
  - class javax.swing.plaf.basic.BasicComboBoxUI
    - class javax.swing.plaf.metal.MetalComboBoxUI
  - class javax.swing.plaf.multi.MultiComboBoxUI
- class javax.swing.plaf.DesktopIconUI
  - class javax.swing.plaf.basic.BasicDesktopIconUI
    - class javax.swing.plaf.metal.MetalDesktopIconUI
  - class javax.swing.plaf.multi.MultiDesktopIconUI
- class javax.swing.plaf.DesktopPaneUI
  - class javax.swing.plaf.basic.BasicDesktopPaneUI
  - class javax.swing.plaf.multi.MultiDesktopPaneUI
- class javax.swing.plaf.FileChooserUI
  - class javax.swing.plaf.basic.BasicFileChooserUI
    - class javax.swing.plaf.metal.MetalFileChooserUI
  - class javax.swing.plaf.multi.MultiFileChooserUI
- class javax.swing.plaf.InternalFrameUI
  - class javax.swing.plaf.basic.BasicInternalFrameUI
    - class javax.swing.plaf.metal.MetalInternalFrameUI
  - class javax.swing.plaf.multi.MultiInternalFrameUI
- class javax.swing.plaf.LabelUI
  - class javax.swing.plaf.basic.BasicLabelUI (implementa java.beans.PropertyChangeListener)
    - class javax.swing.plaf.metal.MetalLabelUI
  - class javax.swing.plaf.multi.MultiLabelUI
- class javax.swing.plaf.ListUI
  - class javax.swing.plaf.basic.BasicListUI
  - class javax.swing.plaf.multi.MultiListUI
- class javax.swing.plaf.MenuBarUI
  - class javax.swing.plaf.basic.BasicMenuBarUI
  - class javax.swing.plaf.multi.MultiMenuBarUI
- class javax.swing.plaf.OptionPaneUI
  - class javax.swing.plaf.basic.BasicOptionPaneUI
  - class javax.swing.plaf.multi.MultiOptionPaneUI
- class javax.swing.plaf.PanelUI
  - class javax.swing.plaf.basic.BasicPanelUI
  - class javax.swing.plaf.multi.MultiPanelUI
- class javax.swing.plaf.PopupMenuUI
  - class javax.swing.plaf.basic.BasicPopupMenuUI
  - class javax.swing.plaf.multi.MultiPopupMenuUI

- class javax.swing.plaf.ProgressBarUI
  - class javax.swing.plaf.basic.BasicProgressBarUI
    - class javax.swing.plaf.metal.MetalProgressBarUI
  - class javax.swing.plaf.multi.MultiProgressBarUI
- class javax.swing.plaf.ScrollBarUI
  - class javax.swing.plaf.basic.BasicScrollBarUI (implementa java.awt.LayoutManager, javax.swing.SwingConstants)
    - class javax.swing.plaf.metal.MetalScrollBarUI
  - class javax.swing.plaf.multi.MultiScrollBarUI
- class javax.swing.plaf.ScrollPaneUI
  - class javax.swing.plaf.basic.BasicScrollPaneUI (implementa javax.swing.ScrollPaneConstants)
    - class javax.swing.plaf.metal.MetalScrollPaneUI
  - class javax.swing.plaf.multi.MultiScrollPaneUI
- class javax.swing.plaf.SeparatorUI
  - class javax.swing.plaf.basic.BasicSeparatorUI
    - class javax.swing.plaf.basic.BasicPopupMenuSeparatorUI
    - class javax.swing.plaf.basic.BasicToolBarSeparatorUI
    - class javax.swing.plaf.metal.MetalSeparatorUI
      - class javax.swing.plaf.metal.MetalPopupMenuSeparatorUI
  - class javax.swing.plaf.multi.MultiSeparatorUI
- class javax.swing.plaf.SliderUI
  - class javax.swing.plaf.basic.BasicSliderUI
    - class javax.swing.plaf.metal.MetalSliderUI
  - class javax.swing.plaf.multi.MultiSliderUI
- class javax.swing.plaf.SplitPaneUI
  - class javax.swing.plaf.basic.BasicSplitPaneUI
    - class javax.swing.plaf.metal.MetalSplitPaneUI
  - class javax.swing.plaf.multi.MultiSplitPaneUI
- class javax.swing.plaf.TabbedPaneUI
  - class javax.swing.plaf.basic.BasicTabbedPaneUI (implementa javax.swing.SwingConstants)
    - class javax.swing.plaf.metal.MetalTabbedPaneUI
  - class javax.swing.plaf.multi.MultiTabbedPaneUI
- class javax.swing.plaf.TableHeaderUI
  - class javax.swing.plaf.basic.BasicTableHeaderUI
  - class javax.swing.plaf.multi.MultiTableHeaderUI
- class javax.swing.plaf.TableUI
  - class javax.swing.plaf.basic.BasicTableUI
  - class javax.swing.plaf.multi.MultiTableUI
- class javax.swing.plaf.TextUI
  - class javax.swing.plaf.basic.BasicTextUI (implementa javax.swing.text.ViewFactory)
    - class javax.swing.plaf.basic.BasicEditorPaneUI
      - class javax.swing.plaf.basic.BasicTextPaneUI
    - class javax.swing.plaf.basic.BasicTextAreaUI
    - class javax.swing.plaf.basic.BasicTextFieldUI
      - class javax.swing.plaf.basic.BasicPasswordFieldUI
      - class javax.swing.plaf.metal.MetalTextFieldUI
    - class javax.swing.text.DefaultTextUI
  - class javax.swing.plaf.multi.MultiTextUI
- class javax.swing.plaf.ToolBarUI
  - class javax.swing.plaf.basic.BasicToolBarUI (implementa javax.swing.SwingConstants)
    - class javax.swing.plaf.metal.MetalToolBarUI
  - class javax.swing.plaf.multi.MultiToolBarUI

- class javax.swing.plaf.ToolTipUI
  - class javax.swing.plaf.basic.BasicToolTipUI
    - class javax.swing.plaf.metal.MetalToolTipUI
  - class javax.swing.plaf.multi.MultiToolTipUI
- class javax.swing.plaf.TreeUI
  - class javax.swing.plaf.basic.BasicTreeUI
    - class javax.swing.plaf.metal.MetalTreeUI
  - class javax.swing.plaf.multi.MultiTreeUI
- class javax.swing.plaf.ViewportUI
  - class javax.swing.plaf.basic.BasicViewportUI
  - class javax.swing.plaf.multi.MultiViewportUI

- class java.awt.event.ContainerAdapter (implementa java.awt.event.ContainerListener)
- class java.net.ContentHandler
- class javax.swing.text.html.parser.ContentModel (implementa java.io.Serializable)
- class org.omg.CORBA.Context
- class org.omg.CORBA.ContextList
- class java.awt.image.ConvolveOp (implementa java.awt.image.BufferedImageOp, java.awt.image.RasterOp)
- class java.util.zip.CRC32 (implementa java.util.zip.Checksum)
- class java.security.cert.CRL
  - class java.security.cert.X509CRL (implementa java.security.cert.X509Extension)
- class javax.swing.text.html.CSS
- class javax.swing.text.html.CSS.Attribute
- class java.awt.geom.CubicCurve2D (implementa java.lang.Cloneable, java.awt.Shape)
  - class java.awt.geom.CubicCurve2D.Double
  - class java.awt.geom.CubicCurve2D.Float
- class java.awt.Cursor (implementa java.io.Serializable)
- class java.awt.image.DataBuffer
  - class java.awt.image.DataBufferByte
  - class java.awt.image.DataBufferInt
  - class java.awt.image.DataBufferShort
  - class java.awt.image.DataBufferUShort
- class java.awt.datatransfer.DataFlavor (implementa java.lang.Cloneable, java.io.Externalizable)
- class java.net.DatagramPacket
- class java.net.DatagramSocket
  - class java.net.MulticastSocket
- class java.net.DatagramSocketImpl (implementa java.net.SocketOptions)
- class java.util.Date (implementa java.lang.Cloneable, java.lang.Comparable, java.io.Serializable)
  - class java.sql.Date
  - class java.sql.Time
  - class java.sql.Timestamp
- class java.text.DateFormatSymbols (implementa java.lang.Cloneable, java.io.Serializable)
- class java.text.DecimalFormatSymbols (implementa java.lang.Cloneable, java.io.Serializable)
- class javax.swing.DefaultBoundedRangeModel (implementa javax.swing.BoundedRangeModel, java.io.Serializable)
- class javax.swing.DefaultButtonModel (implementa javax.swing.ButtonModel, java.io.Serializable)
  - class javax.swing.JToggleButton.ToggleButtonModel
- class javax.swing.DefaultCellEditor (implementa java.io.Serializable, javax.swing.table.TableCellEditor, javax.swing.tree.TreeCellEditor)
- class javax.swing.DefaultCellEditor.EditorDelegate (implementa java.awt.event.ActionListener, java.awt.event.ItemListener, java.io.Serializable)
- class javax.swing.colorchooser.DefaultColorSelectionModel (implementa javax.swing.colorchooser.ColorSelectionModel, java.io.Serializable)

- class javax.swing.DefaultDesktopManager (implementa javax.swing.DesktopManager, java.io.Serializable)
- class javax.swing.DefaultListSelectionModel (implementa java.lang.Cloneable, javax.swing.ListSelectionModel, java.io.Serializable)
- class javax.swing.tree.DefaultMutableTreeNode (implementa java.lang.Cloneable, javax.swing.tree.MutableTreeNode, java.io.Serializable)
    - class javax.swing.JTree.DynamicUtilTreeNode
- class javax.swing.DefaultSingleSelectionModel (implementa java.io.Serializable, javax.swing.SingleSelectionModel)
- class javax.swing.text.DefaultStyledDocument.ElementBuffer (implementa java.io.Serializable)
- class javax.swing.text.DefaultStyledDocument.ElementSpec
- class javax.swing.table.DefaultTableColumnModel (implementa javax.swing.event.ListSelectionListener, java.beans.PropertyChangeListener, java.io.Serializable, javax.swing.table.TableColumnModel)
- class javax.swing.tree.DefaultTreeCellEditor (implementa java.awt.event.ActionListener, javax.swing.tree.TreeCellEditor, javax.swing.event.TreeSelectionListener)
- class javax.swing.tree.DefaultTreeModel (implementa java.io.Serializable, javax.swing.tree.TreeModel)
- class javax.swing.tree.DefaultTreeSelectionModel (implementa java.lang.Cloneable, java.io.Serializable, javax.swing.tree.TreeSelectionModel)
    - class javax.swing.JTree.EmptySelectionModel
- class org.omg.CORBA.DefinitionKind (implementa org.omg.CORBA.portable.IDLEntity)
- class java.util.zip.Deflater
- class org.omg.CORBA.portable.Delegate
- class java.util.Dictionary
    - class java.util.Hashtable (implementa java.lang.Cloneable, java.util.Map, java.io.Serializable)
        - class java.util.Properties
            - class java.security.Provider
        - class javax.swing.UIDefaults
- class java.awt.geom.Dimension2D (implementa java.lang.Cloneable)
    - class java.awt.Dimension (implementa java.io.Serializable)
        - class javax.swing.plaf.DimensionUIResource (implementa javax.swing.plaf.UIResource)
- class java.awt.dnd.DnDConstants
- class javax.swing.event.DocumentEvent.EventType
- class org.omg.CORBA.DoubleHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.awt.dnd.DragGestureRecognizer
    - class java.awt.dnd.MouseDragGestureRecognizer (implementa java.awt.event.MouseListener, java.awt.event.MouseMotionListener)
- class java.awt.dnd.DragSource
- class java.awt.dnd.DragSourceContext (implementa java.awt.dnd.DragSourceListener)
- class java.sql.DriverManager
- class java.sql.DriverPropertyInfo
- class java.awt.dnd.DropTarget (implementa java.awt.dnd.DropTargetListener, java.io.Serializable)
- class java.awt.dnd.DropTarget.DropTargetAutoScroller (implementa java.awt.event.ActionListener)
- class java.awt.dnd.DropTargetContext
- class java.awt.dnd.DropTargetContext.TransferableProxy (implementa java.awt.datatransfer.Transferable)
- class java.security.spec.DSAParameterSpec (implementa java.security.spec.AlgorithmParameterSpec, java.security.interfaces.DSAParams)
- class java.security.spec.DSAPrivateKeySpec (implementa java.security.spec.KeySpec)
- class java.security.spec.DSAPublicKeySpec (implementa java.security.spec.KeySpec)

- class javax.swing.text.html.parser.DTD (implementa javax.swing.text.html.parser.DTDConstants)
- class javax.swing.text.EditorKit (implementa java.lang.Cloneable, java.io.Serializable)
    - class javax.swing.text.DefaultEditorKit
        - class javax.swing.text.StyledEditorKit
            - class javax.swing.text.html.HTMLEditorKit
            - class javax.swing.text.rtf.RTFEditorKit
- class javax.swing.text.html.parser.Element (implementa javax.swing.text.html.parser.DTDConstants, java.io.Serializable)
- class javax.swing.text.ElementIterator (implementa java.lang.Cloneable)
- class java.security.spec.EncodedKeySpec (implementa java.security.spec.KeySpec)
    - class java.security.spec.PKCS8EncodedKeySpec
    - class java.security.spec.X509EncodedKeySpec
- class javax.swing.text.html.parser.Entity (implementa javax.swing.text.html.parser.DTDConstants)
- class org.omg.CORBA.Environment
- class java.awt.Event (implementa java.io.Serializable)
- class javax.swing.event.EventListenerList (implementa java.io.Serializable)
- class java.util.EventObject (implementa java.io.Serializable)
    - class java.awt.AWTEvent
        - class java.awt.event.ActionEvent
        - class java.awt.event.AdjustmentEvent
        - class javax.swing.event.AncestorEvent
        - class java.awt.event.ComponentEvent
            - class java.awt.event.ContainerEvent
            - class java.awt.event.FocusEvent
            - class java.awt.event.InputEvent
                - class java.awt.event.KeyEvent
                    - class javax.swing.event.MenuKeyEvent
                - class java.awt.event.MouseEvent
                    - class javax.swing.event.MenuDragMouseEvent
            - class java.awt.event.PaintEvent
            - class java.awt.event.WindowEvent
        - class java.awt.event.InputMethodEvent
        - class javax.swing.event.InternalFrameEvent
        - class java.awt.event.InvocationEvent (implementa java.awt.ActiveEvent)
        - class java.awt.event.ItemEvent
        - class java.awt.event.TextEvent
    - class java.beans.beancontext.BeanContextEvent
        - class java.beans.beancontext.BeanContextMembershipEvent
        - class java.beans.beancontext.BeanContextServiceAvailableEvent
        - class java.beans.beancontext.BeanContextServiceRevokedEvent
    - class javax.swing.event.CaretEvent
    - class javax.swing.event.ChangeEvent
    - class java.awt.dnd.DragGestureEvent
    - class java.awt.dnd.DragSourceEvent
        - class java.awt.dnd.DragSourceDragEvent
        - class java.awt.dnd.DragSourceDropEvent
    - class java.awt.dnd.DropTargetEvent
        - class java.awt.dnd.DropTargetDragEvent
        - class java.awt.dnd.DropTargetDropEvent
    - class javax.swing.event.HyperlinkEvent
        - class javax.swing.text.html.HTMLFrameHyperlinkEvent
    - class javax.swing.event.ListDataEvent
    - class javax.swing.event.ListSelectionEvent
    - class javax.swing.event.MenuEvent

- class javax.swing.event.PopupMenuEvent
- class java.beans.PropertyChangeEvent
- class javax.swing.event.TableColumnModelEvent
- class javax.swing.event.TableModelEvent
- class javax.swing.event.TreeExpansionEvent
- class javax.swing.event.TreeModelEvent
- class javax.swing.event.TreeSelectionEvent
- class javax.swing.event.UndoableEditEvent

- class java.awt.EventQueue
- class org.omg.CORBA.ExceptionList
- class java.beans.FeatureDescriptor
  - class java.beans.BeanDescriptor
  - class java.beans.EventSetDescriptor
  - class java.beans.MethodDescriptor
  - class java.beans.ParameterDescriptor
  - class java.beans.PropertyDescriptor
    - class java.beans.IndexedPropertyDescriptor
- class java.text.FieldPosition
- class java.io.File (implementa java.lang.Comparable, java.io.Serializable)
- class java.io.FileDescriptor
- class javax.swing.filechooser.FileFilter
  - class javax.swing.plaf.basic.BasicFileChooserUI.AcceptAllFileFilter
- class javax.swing.filechooser.FileSystemView
- class javax.swing.filechooser.FileView
  - class javax.swing.plaf.basic.BasicFileChooserUI.BasicFileView
- class java.awt.image.FilteredImageSource (implementa java.awt.image.ImageProducer)
- class org.omg.CORBA.FixedHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.awt.geom.FlatteningPathIterator (implementa java.awt.geom.PathIterator)
- class org.omg.CORBA.FloatHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.awt.FlowLayout (implementa java.awt.LayoutManager, java.io.Serializable)
- class java.awt.event.FocusAdapter (implementa java.awt.event.FocusListener)
  - class javax.swing.plaf.basic.BasicSplitPaneUI.FocusHandler
  - class javax.swing.plaf.basic.BasicTabbedPaneUI.FocusHandler
- class javax.swing.FocusManager
  - class javax.swing.DefaultFocusManager
- class java.awt.Font (implementa java.io.Serializable)
  - class javax.swing.plaf.FontUIResource (implementa javax.swing.plaf.UIResource)
- class java.awt.FontMetrics (implementa java.io.Serializable)
- class java.awt.font.FontRenderContext
- class java.text.Format (implementa java.lang.Cloneable, java.io.Serializable)
  - class java.text.DateFormat
    - class java.text.SimpleDateFormat
  - class java.text.MessageFormat
  - class java.text.NumberFormat
    - class java.text.ChoiceFormat
    - class java.text.DecimalFormat
- class javax.swing.text.GapVector (implementa java.io.Serializable)
  - class javax.swing.text.GapContent (implementa javax.swing.text.AbstractDocument.Content, java.io.Serializable)
- class java.awt.geom.GeneralPath (implementa java.lang.Cloneable, java.awt.Shape)
- class java.awt.font.GlyphJustificationInfo
- class java.awt.font.GlyphMetrics
- class java.awt.font.GlyphVector (implementa java.lang.Cloneable)
- class java.awt.GradientPaint (implementa java.awt.Paint)
- class java.awt.font.GraphicAttribute
  - class java.awt.font.ImageGraphicAttribute

- class java.awt.font.ShapeGraphicAttribute
- class java.awt.Graphics
  - class javax.swing.DebugGraphics
  - class java.awt.Graphics2D
- class java.awt.GraphicsConfigTemplate (implementa java.io.Serializable)
- class java.awt.GraphicsConfiguration
- class java.awt.GraphicsDevice
- class java.awt.GraphicsEnvironment
- class java.awt.GridBagConstraints (implementa java.lang.Cloneable, java.io.Serializable)
- class java.awt.GridBagLayout (implementa java.awt.LayoutManager2, java.io.Serializable)
- class java.awt.GridLayout (implementa java.awt.LayoutManager, java.io.Serializable)
- class java.security.GuardedObject (implementa java.io.Serializable)
- class javax.swing.text.html.HTML
- class javax.swing.text.html.HTML.Attribute
- class javax.swing.text.html.HTML.Tag
  - class javax.swing.text.html.HTML.UnknownTag (implementa java.io.Serializable)
- class javax.swing.text.html.HTMLDocument.HTMLReader.TagAction
  - class javax.swing.text.html.HTMLDocument.HTMLReader.BlockAction
    - class javax.swing.text.html.HTMLDocument.HTMLReader.ParagraphAction
    - class javax.swing.text.html.HTMLDocument.HTMLReader.PreAction
  - class javax.swing.text.html.HTMLDocument.HTMLReader.CharacterAction
  - class javax.swing.text.html.HTMLDocument.HTMLReader.HiddenAction
  - class javax.swing.text.html.HTMLDocument.HTMLReader.IsindexAction
  - class javax.swing.text.html.HTMLDocument.HTMLReader.SpecialAction
    - class javax.swing.text.html.HTMLDocument.HTMLReader.FormAction
- class javax.swing.text.html.HTMLDocument.Iterator
- class javax.swing.text.html.HTMLEditorKit.HTMLFactory (implementa javax.swing.text.ViewFactory)
- class javax.swing.text.html.HTMLEditorKit.Parser
  - class javax.swing.text.html.parser.ParserDelegator
- class javax.swing.text.html.HTMLEditorKit.ParserCallback
  - class javax.swing.text.html.HTMLDocument.HTMLReader
- class javax.swing.event.HyperlinkEvent.EventType
- class java.awt.color.ICC_Profile
  - class java.awt.color.ICC_ProfileGray
  - class java.awt.color.ICC_ProfileRGB
- class javax.swing.plaf.IconUIResource (implementa javax.swing.Icon, java.io.Serializable, javax.swing.plaf.UIResource)
- class java.security.Identity (implementa java.security.Principal, java.io.Serializable)
  - class java.security.IdentityScope
  - class java.security.Signer
- class java.awt.Image
  - class java.awt.image.BufferedImage (implementa java.awt.image.WritableRenderedImage)
- class java.awt.image.ImageFilter (implementa java.lang.Cloneable, java.awt.image.ImageConsumer)
  - class java.awt.image.BufferedImageFilter (implementa java.lang.Cloneable)
  - class java.awt.image.CropImageFilter
  - class java.awt.image.ReplicateScaleFilter
    - class java.awt.image.AreaAveragingScaleFilter
  - class java.awt.image.RGBImageFilter
    - class javax.swing.GrayFilter
- class javax.swing.ImageIcon (implementa javax.swing.Icon, java.io.Serializable)
- class java.net.InetAddress (implementa java.io.Serializable)
- class java.util.zip.Inflater
- class java.awt.im.InputContext

- class java.awt.im.InputMethodHighlight
- class java.io.InputStream
  - class java.io.ByteArrayInputStream
  - class java.io.FileInputStream
  - class java.io.FilterInputStream
    - class java.io.BufferedInputStream
    - class java.util.zip.CheckedInputStream
    - class java.io.DataInputStream (implementa java.io.DataInput)
    - class java.security.DigestInputStream
    - class java.util.zip.InflaterInputStream
      - class java.util.zip.GZIPInputStream
      - class java.util.zip.ZipInputStream (implementa java.util.zip.ZipConstants)
        - class java.util.jar.JarInputStream
    - class java.io.LineNumberInputStream
    - class javax.swing.ProgressMonitorInputStream
    - class java.io.PushbackInputStream
  - class org.omg.CORBA.portable.InputStream
  - class java.io.ObjectInputStream (implementa java.io.ObjectInput, java.io.ObjectStreamConstants)
  - class java.io.PipedInputStream
  - class java.io.SequenceInputStream
  - class java.io.StringBufferInputStream
- class java.awt.Insets (implementa java.lang.Cloneable, java.io.Serializable)
  - class javax.swing.plaf.InsetsUIResource (implementa javax.swing.plaf.UIResource)
- class javax.swing.event.InternalFrameAdapter (implementa javax.swing.event.InternalFrameListener)
- class org.omg.CORBA.IntHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.beans.Introspector
- class org.omg.CosNaming.NamingContextPackage.InvalidNameHelper
- class org.omg.CosNaming.NamingContextPackage.InvalidNameHolder (implementa org.omg.CORBA.portable.Streamable)
- class org.omg.CosNaming.IstringHelper
- class javax.swing.JComponent.AccessibleJComponent.AccessibleContainerHandler (implementa java.awt.event.ContainerListener)
- class javax.swing.JRootPane.RootLayout (implementa java.awt.LayoutManager2, java.io.Serializable)
- class javax.swing.JTabbedPane.ModelListener (implementa javax.swing.event.ChangeListener, java.io.Serializable)
- class javax.swing.text.JTextComponent.KeyBinding
- class javax.swing.JTree.TreeModelHandler (implementa javax.swing.event.TreeModelListener)
- class javax.swing.JTree.TreeSelectionRedirector (implementa java.io.Serializable, javax.swing.event.TreeSelectionListener)
- class java.awt.image.Kernel (implementa java.lang.Cloneable)
- class java.awt.event.KeyAdapter (implementa java.awt.event.KeyListener)
  - class javax.swing.plaf.basic.BasicComboBoxUI.KeyHandler
  - class javax.swing.plaf.basic.BasicComboPopup.InvocationKeyHandler
  - class javax.swing.plaf.basic.BasicTreeUI.KeyHandler
- class java.security.KeyFactory
- class java.security.KeyFactorySpi
- class java.security.KeyPair (implementa java.io.Serializable)
- class java.security.KeyPairGeneratorSpi
  - class java.security.KeyPairGenerator
- class java.security.KeyStore
- class java.security.KeyStoreSpi

- class javax.swing.KeyStroke (implementa java.io.Serializable)
- class javax.swing.text.LayeredHighlighter (implementa javax.swing.text.Highlighter)
    - class javax.swing.text.DefaultHighlighter
        - class javax.swing.plaf.basic.BasicTextUI.BasicHighlighter (implementa javax.swing.plaf.UIResource)
- class javax.swing.text.LayeredHighlighter.LayerPainter (implementa javax.swing.text.Highlighter.HighlightPainter)
    - class javax.swing.text.DefaultHighlighter.DefaultHighlightPainter
- class java.rmi.dgc.Lease (implementa java.io.Serializable)
- class java.awt.geom.Line2D (implementa java.lang.Cloneable, java.awt.Shape)
    - class java.awt.geom.Line2D.Double
    - class java.awt.geom.Line2D.Float
- class java.awt.font.LineBreakMeasurer
- class java.awt.font.LineMetrics
- class java.util.Locale (implementa java.lang.Cloneable, java.io.Serializable)
- class java.rmi.registry.LocateRegistry
- class org.omg.CORBA.LongHolder (implementa org.omg.CORBA.portable.Streamable)
- class javax.swing.LookAndFeel
    - class javax.swing.plaf.basic.BasicLookAndFeel (implementa java.io.Serializable)
        - class javax.swing.plaf.metal.MetalLookAndFeel
    - class javax.swing.plaf.multi.MultiLookAndFeel
- class java.awt.image.LookupOp (implementa java.awt.image.BufferedImageOp, java.awt.image.RasterOp)
- class java.awt.image.LookupTable
    - class java.awt.image.ByteLookupTable
    - class java.awt.image.ShortLookupTable
- class java.util.jar.Manifest (implementa java.lang.Cloneable)
- class java.rmi.MarshalledObject (implementa java.io.Serializable)
- class java.lang.Math
- class java.awt.MediaTracker (implementa java.io.Serializable)
- class java.awt.image.MemoryImageSource (implementa java.awt.image.ImageProducer)
- class java.awt.MenuComponent (implementa java.io.Serializable)
    - class java.awt.MenuBar (implementa java.awt.MenuContainer)
    - class java.awt.MenuItem
        - class java.awt.CheckboxMenuItem (implementa java.awt.ItemSelectable)
        - class java.awt.Menu (implementa java.awt.MenuContainer)
            - class java.awt.PopupMenu
- class javax.swing.MenuSelectionManager
- class java.awt.MenuShortcut (implementa java.io.Serializable)
- class java.security.MessageDigestSpi
    - class java.security.MessageDigest
- class javax.swing.plaf.metal.MetalBorders
- class javax.swing.plaf.metal.MetalCheckBoxIcon (implementa javax.swing.Icon, java.io.Serializable, javax.swing.plaf.UIResource)
- class javax.swing.plaf.metal.MetalComboBoxIcon (implementa javax.swing.Icon, java.io.Serializable)
- class javax.swing.plaf.metal.MetalIconFactory (implementa java.io.Serializable)
- class javax.swing.plaf.metal.MetalIconFactory.FileIcon16 (implementa javax.swing.Icon, java.io.Serializable)
    - class javax.swing.plaf.metal.MetalIconFactory.TreeLeafIcon
- class javax.swing.plaf.metal.MetalIconFactory.FolderIcon16 (implementa javax.swing.Icon, java.io.Serializable)
    - class javax.swing.plaf.metal.MetalIconFactory.TreeFolderIcon
- class javax.swing.plaf.metal.MetalIconFactory.TreeControlIcon (implementa javax.swing.Icon, java.io.Serializable)
- class javax.swing.plaf.metal.MetalTheme

        - class javax.swing.plaf.metal.DefaultMetalTheme
- class javax.swing.plaf.metal.MetalToolBarUI.MetalContainerListener (implementa java.awt.event.ContainerListener)
- class javax.swing.plaf.metal.MetalToolBarUI.MetalRolloverListener (implementa java.beans.PropertyChangeListener)
- class java.lang.reflect.Modifier
- class java.awt.event.MouseAdapter (implementa java.awt.event.MouseListener)
    - class javax.swing.plaf.basic.BasicComboPopup.InvocationMouseHandler
    - class javax.swing.plaf.basic.BasicComboPopup.ListMouseHandler
    - class javax.swing.plaf.basic.BasicFileChooserUI.DoubleClickListener
    - class javax.swing.plaf.basic.BasicScrollBarUI.ArrowButtonListener
    - class javax.swing.plaf.basic.BasicScrollBarUI.TrackListener (implementa java.awt.event.MouseMotionListener)
    - class javax.swing.plaf.basic.BasicSplitPaneDivider.MouseHandler (implementa java.awt.event.MouseMotionListener)
    - class javax.swing.plaf.basic.BasicTabbedPaneUI.MouseHandler
    - class javax.swing.plaf.basic.BasicTreeUI.MouseHandler
    - class javax.swing.text.html.FormView.MouseEventListener
    - class javax.swing.text.html.HTMLEditorKit.LinkController (implementa java.io.Serializable)
    - class javax.swing.plaf.metal.MetalFileChooserUI.SingleClickListener
    - class javax.swing.ToolTipManager (implementa java.awt.event.MouseMotionListener)
- class javax.swing.event.MouseInputAdapter (implementa javax.swing.event.MouseInputListener)
    - class javax.swing.plaf.basic.BasicDesktopIconUI.MouseInputHandler
    - class javax.swing.plaf.basic.BasicInternalFrameUI.BorderListener (implementa javax.swing.SwingConstants)
    - class javax.swing.plaf.basic.BasicSliderUI.TrackListener
- class java.awt.event.MouseMotionAdapter (implementa java.awt.event.MouseMotionListener)
    - class javax.swing.plaf.basic.BasicComboPopup.InvocationMouseMotionHandler
    - class javax.swing.plaf.basic.BasicComboPopup.ListMouseMotionHandler
- class org.omg.CosNaming.NameComponent (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CosNaming.NameComponentHelper
- class org.omg.CosNaming.NameComponentHolder (implementa org.omg.CORBA.portable.Streamable)
- class org.omg.CORBA.NamedValue
- class org.omg.CosNaming.NameHelper
- class org.omg.CosNaming.NameHolder (implementa org.omg.CORBA.portable.Streamable)
- class org.omg.CORBA.NameValuePair (implementa org.omg.CORBA.portable.IDLEntity)
- class java.rmi.Naming
- class org.omg.CosNaming.NamingContextHelper
- class org.omg.CosNaming.NamingContextHolder (implementa org.omg.CORBA.portable.Streamable)
- class org.omg.CosNaming.NamingContextPackage.NotEmptyHelper
- class org.omg.CosNaming.NamingContextPackage.NotEmptyHolder (implementa org.omg.CORBA.portable.Streamable)
- class org.omg.CosNaming.NamingContextPackage.NotFoundHelper
- class org.omg.CosNaming.NamingContextPackage.NotFoundHolder (implementa org.omg.CORBA.portable.Streamable)
- class org.omg.CosNaming.NamingContextPackage.NotFoundReason (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CosNaming.NamingContextPackage.NotFoundReasonHelper
- class org.omg.CosNaming.NamingContextPackage.NotFoundReasonHolder (implementa org.omg.CORBA.portable.Streamable)

- class java.lang.Number (implementa java.io.Serializable)
  - class java.math.BigDecimal (implementa java.lang.Comparable)
  - class java.math.BigInteger (implementa java.lang.Comparable)
  - class java.lang.Byte (implementa java.lang.Comparable)
  - class java.lang.Double (implementa java.lang.Comparable)
  - class java.lang.Float (implementa java.lang.Comparable)
  - class java.lang.Integer (implementa java.lang.Comparable)
  - class java.lang.Long (implementa java.lang.Comparable)
  - class java.lang.Short (implementa java.lang.Comparable)
- class org.omg.CORBA.NVList
- class org.omg.CORBA.ObjectHolder (implementa org.omg.CORBA.portable.Streamable)
- class org.omg.CORBA.portable.ObjectImpl (implementa org.omg.CORBA.Object)
  - class org.omg.CosNaming._BindingIteratorStub (implementa org.omg.CosNaming.BindingIterator)
  - class org.omg.CosNaming._NamingContextStub (implementa org.omg.CosNaming.NamingContext)
  - class org.omg.CORBA.DynamicImplementation
    - class org.omg.CosNaming._BindingIteratorImplBase (implementa org.omg.CosNaming.BindingIterator)
    - class org.omg.CosNaming._NamingContextImplBase (implementa org.omg.CosNaming.NamingContext)
- class java.io.ObjectInputStream.GetField
- class java.io.ObjectOutputStream.PutField
- class java.io.ObjectStreamClass (implementa java.io.Serializable)
- class java.io.ObjectStreamField (implementa java.lang.Comparable)
- class java.rmi.server.ObjID (implementa java.io.Serializable)
- class java.util.Observable
- class java.rmi.server.Operation
- class javax.swing.text.html.Option
- class org.omg.CORBA.ORB
- class java.io.OutputStream
  - class java.io.ByteArrayOutputStream
  - class java.io.FileOutputStream
  - class java.io.FilterOutputStream
    - class java.io.BufferedOutputStream
    - class java.util.zip.CheckedOutputStream
    - class java.io.DataOutputStream (implementa java.io.DataOutput)
    - class java.util.zip.DeflaterOutputStream
      - class java.util.zip.GZIPOutputStream
      - class java.util.zip.ZipOutputStream (implementa java.util.zip.ZipConstants)
        - class java.util.jar.JarOutputStream
    - class java.security.DigestOutputStream
    - class java.io.PrintStream
      - class java.rmi.server.LogStream
  - class java.io.ObjectOutputStream (implementa java.io.ObjectOutput, java.io.ObjectStreamConstants)
  - class org.omg.CORBA.portable.OutputStream
  - class java.io.PipedOutputStream
- class javax.swing.OverlayLayout (implementa java.awt.LayoutManager2, java.io.Serializable)
- class java.lang.Package
- class java.awt.print.PageFormat (implementa java.lang.Cloneable)
- class java.awt.print.Paper (implementa java.lang.Cloneable)
- class java.awt.image.renderable.ParameterBlock (implementa java.lang.Cloneable, java.io.Serializable)
- class java.text.ParsePosition

- class javax.swing.text.html.parser.Parser (implementa javax.swing.text.html.parser.DTDConstants)
  - class javax.swing.text.html.parser.DocumentParser
- class java.net.PasswordAuthentication
- class java.security.Permission (implementa java.security.Guard, java.io.Serializable)
  - class java.security.AllPermission
  - class java.security.BasicPermission (implementa java.io.Serializable)
    - class java.awt.AWTPermission
    - class java.net.NetPermission
    - class java.util.PropertyPermission
    - class java.lang.reflect.ReflectPermission
    - class java.lang.RuntimePermission
    - class java.security.SecurityPermission
    - class java.io.SerializablePermission
  - class java.io.FilePermission (implementa java.io.Serializable)
  - class java.net.SocketPermission (implementa java.io.Serializable)
  - class java.security.UnresolvedPermission (implementa java.io.Serializable)
- class java.security.PermissionCollection (implementa java.io.Serializable)
  - class java.security.Permissions (implementa java.io.Serializable)
- class java.awt.image.PixelGrabber (implementa java.awt.image.ImageConsumer)
- class java.awt.geom.Point2D (implementa java.lang.Cloneable)
  - class java.awt.Point (implementa java.io.Serializable)
  - class java.awt.geom.Point2D.Double
  - class java.awt.geom.Point2D.Float
- class java.security.Policy
- class java.awt.Polygon (implementa java.io.Serializable, java.awt.Shape)
- class javax.swing.text.Position.Bias
- class org.omg.CORBA.Principal
- class org.omg.CORBA.PrincipalHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.awt.print.PrinterJob
- class java.awt.PrintJob
- class java.lang.Process
- class javax.swing.ProgressMonitor
- class java.beans.PropertyChangeSupport (implementa java.io.Serializable)
  - class javax.swing.event.SwingPropertyChangeSupport
- class java.beans.PropertyEditorManager
- class java.beans.PropertyEditorSupport (implementa java.beans.PropertyEditor)
- class java.security.ProtectionDomain
- class java.awt.geom.QuadCurve2D (implementa java.lang.Cloneable, java.awt.Shape)
  - class java.awt.geom.QuadCurve2D.Double
  - class java.awt.geom.QuadCurve2D.Float
- class java.util.Random (implementa java.io.Serializable)
  - class java.security.SecureRandom
- class java.io.RandomAccessFile (implementa java.io.DataInput, java.io.DataOutput)
- class java.awt.image.Raster
  - class java.awt.image.WritableRaster
- class java.io.Reader
  - class java.io.BufferedReader
    - class java.io.LineNumberReader
  - class java.io.CharArrayReader
  - class java.io.FilterReader
    - class java.io.PushbackReader
  - class java.io.InputStreamReader
    - class java.io.FileReader
  - class java.io.PipedReader
  - class java.io.StringReader

- class java.awt.geom.RectangularShape (implementa java.lang.Cloneable, java.awt.Shape)
  - class java.awt.geom.Arc2D
    - class java.awt.geom.Arc2D.Double
    - class java.awt.geom.Arc2D.Float
  - class java.awt.geom.Ellipse2D
    - class java.awt.geom.Ellipse2D.Double
    - class java.awt.geom.Ellipse2D.Float
  - class java.awt.geom.Rectangle2D
    - class java.awt.Rectangle (implementa java.io.Serializable, java.awt.Shape)
      - class javax.swing.text.DefaultCaret (implementa javax.swing.text.Caret, java.awt.event.FocusListener, java.awt.event.MouseListener, java.awt.event.MouseMotionListener)
        - class javax.swing.plaf.basic.BasicTextUI.BasicCaret (implementa javax.swing.plaf.UIResource)
    - class java.awt.geom.Rectangle2D.Double
    - class java.awt.geom.Rectangle2D.Float
  - class java.awt.geom.RoundRectangle2D
    - class java.awt.geom.RoundRectangle2D.Double
    - class java.awt.geom.RoundRectangle2D.Float
- class java.lang.ref.Reference
  - class java.lang.ref.PhantomReference
  - class java.lang.ref.SoftReference
  - class java.lang.ref.WeakReference
- class java.lang.ref.ReferenceQueue
- class java.rmi.server.RemoteObject (implementa java.rmi.Remote, java.io.Serializable)
  - class java.rmi.server.RemoteServer
    - class java.rmi.activation.Activatable
    - class java.rmi.server.UnicastRemoteObject
      - class java.rmi.activation.ActivationGroup (implementa java.rmi.activation.ActivationInstantiator)
  - class java.rmi.server.RemoteStub
- class java.awt.image.renderable.RenderableImageOp (implementa java.awt.image.renderable.RenderableImage)
- class java.awt.image.renderable.RenderableImageProducer (implementa java.awt.image.ImageProducer, java.lang.Runnable)
- class java.awt.image.renderable.RenderContext (implementa java.lang.Cloneable)
- class java.awt.RenderingHints (implementa java.lang.Cloneable, java.util.Map)
- class java.awt.RenderingHints.Key
- class javax.swing.RepaintManager
- class org.omg.CORBA.Request
- class java.awt.image.RescaleOp (implementa java.awt.image.BufferedImageOp, java.awt.image.RasterOp)
- class java.util.ResourceBundle
  - class java.util.ListResourceBundle
    - class javax.accessibility.AccessibleResourceBundle
  - class java.util.PropertyResourceBundle
- class java.rmi.server.RMIClassLoader
- class java.rmi.server.RMISocketFactory (implementa java.rmi.server.RMIClientSocketFactory, java.rmi.server.RMIServerSocketFactory)
- class java.security.spec.RSAPrivateKeySpec (implementa java.security.spec.KeySpec)
  - class java.security.spec.RSAPrivateCrtKeySpec
- class java.security.spec.RSAPublicKeySpec (implementa java.security.spec.KeySpec)
- class java.lang.Runtime
- class java.awt.image.SampleModel
  - class java.awt.image.ComponentSampleModel

        - class java.awt.image.BandedSampleModel
        - class java.awt.image.PixelInterleavedSampleModel
    - class java.awt.image.MultiPixelPackedSampleModel
    - class java.awt.image.SinglePixelPackedSampleModel
- class javax.swing.ScrollPaneLayout (implementa java.awt.LayoutManager, javax.swing.ScrollPaneConstants, java.io.Serializable)
    - class javax.swing.ScrollPaneLayout.UIResource (implementa javax.swing.plaf.UIResource)
- class java.security.SecureRandomSpi (implementa java.io.Serializable)
- class java.security.Security
- class java.lang.SecurityManager
    - class java.rmi.RMISecurityManager
- class javax.swing.text.Segment
- class org.omg.CORBA.portable.ServantObject
- class org.omg.CORBA.ServerRequest
- class java.net.ServerSocket
- class org.omg.CORBA.ServiceDetail (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CORBA.ServiceDetailHelper
- class org.omg.CORBA.ServiceInformation (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CORBA.ServiceInformationHelper
- class org.omg.CORBA.ServiceInformationHolder (implementa org.omg.CORBA.portable.Streamable)
- class org.omg.CORBA.SetOverrideType (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CORBA.ShortHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.security.SignatureSpi
    - class java.security.Signature
- class java.security.SignedObject (implementa java.io.Serializable)
- class javax.swing.text.SimpleAttributeSet (implementa java.lang.Cloneable, javax.swing.text.MutableAttributeSet, java.io.Serializable)
- class java.beans.SimpleBeanInfo (implementa java.beans.BeanInfo)
- class javax.swing.SizeRequirements (implementa java.io.Serializable)
- class java.net.Socket
- class java.net.SocketImpl (implementa java.net.SocketOptions)
- class java.io.StreamTokenizer
- class java.lang.String (implementa java.lang.Comparable, java.io.Serializable)
- class java.lang.StringBuffer (implementa java.io.Serializable)
- class java.text.StringCharacterIterator (implementa java.text.CharacterIterator)
- class javax.swing.text.StringContent (implementa javax.swing.text.AbstractDocument.Content, java.io.Serializable)
- class org.omg.CORBA.StringHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.awt.datatransfer.StringSelection (implementa java.awt.datatransfer.ClipboardOwner, java.awt.datatransfer.Transferable)
- class java.util.StringTokenizer (implementa java.util.Enumeration)
- class org.omg.CORBA.StructMember (implementa org.omg.CORBA.portable.IDLEntity)
- class javax.swing.text.StyleConstants
    - class javax.swing.text.StyleConstants.CharacterConstants (implementa javax.swing.text.AttributeSet.CharacterAttribute)
    - class javax.swing.text.StyleConstants.ColorConstants (implementa javax.swing.text.AttributeSet.CharacterAttribute, javax.swing.text.AttributeSet.ColorAttribute)
    - class javax.swing.text.StyleConstants.FontConstants (implementa javax.swing.text.AttributeSet.CharacterAttribute, javax.swing.text.AttributeSet.FontAttribute)
    - class javax.swing.text.StyleConstants.ParagraphConstants (implementa javax.swing.text.AttributeSet.ParagraphAttribute)

- class javax.swing.text.StyleContext (implementa javax.swing.text.AbstractDocument.AttributeContext, java.io.Serializable)
  - class javax.swing.text.html.StyleSheet
- class javax.swing.text.StyleContext.NamedStyle (implementa java.io.Serializable, javax.swing.text.Style)
- class javax.swing.text.StyleContext.SmallAttributeSet (implementa javax.swing.text.AttributeSet)
- class javax.swing.text.html.StyleSheet.BoxPainter (implementa java.io.Serializable)
- class javax.swing.text.html.StyleSheet.ListPainter (implementa java.io.Serializable)
- class javax.swing.SwingUtilities (implementa javax.swing.SwingConstants)
- class java.lang.System
- class java.awt.datatransfer.SystemFlavorMap (implementa java.awt.datatransfer.FlavorMap)
- class javax.swing.table.TableColumn (implementa java.io.Serializable)
- class javax.swing.text.TabSet (implementa java.io.Serializable)
- class javax.swing.text.TabStop (implementa java.io.Serializable)
- class javax.swing.text.html.parser.TagElement
- class org.omg.CORBA.TCKind
- class java.awt.font.TextHitInfo
- class java.awt.font.TextLayout (implementa java.lang.Cloneable)
- class java.awt.font.TextLayout.CaretPolicy
- class java.awt.font.TextLine.TextLineMetrics
- class java.awt.TexturePaint (implementa java.awt.Paint)
- class java.lang.Thread (implementa java.lang.Runnable)
- class java.lang.ThreadGroup
- class java.lang.ThreadLocal
  - class java.lang.InheritableThreadLocal
- class java.lang.Throwable (implementa java.io.Serializable)
  - class java.lang.Error
    - class java.awt.AWTError
    - class java.lang.LinkageError
      - class java.lang.ClassCircularityError
      - class java.lang.ClassFormatError
        - class java.lang.UnsupportedClassVersionError
      - class java.lang.ExceptionInInitializerError
      - class java.lang.IncompatibleClassChangeError
        - class java.lang.AbstractMethodError
        - class java.lang.IllegalAccessError
        - class java.lang.InstantiationError
        - class java.lang.NoSuchFieldError
        - class java.lang.NoSuchMethodError
      - class java.lang.NoClassDefFoundError
      - class java.lang.UnsatisfiedLinkError
      - class java.lang.VerifyError
    - class java.lang.ThreadDeath
    - class java.lang.VirtualMachineError
      - class java.lang.InternalError
      - class java.lang.OutOfMemoryError
      - class java.lang.StackOverflowError
      - class java.lang.UnknownError
  - class java.lang.Exception
    - class java.security.acl.AclNotFoundException
    - class java.rmi.activation.ActivationException
      - class java.rmi.activation.UnknownGroupException
      - class java.rmi.activation.UnknownObjectException
    - class java.rmi.AlreadyBoundException
    - class org.omg.CORBA.portable.ApplicationException

- class java.awt.AWTException
- class javax.swing.text.BadLocationException
- class java.lang.ClassNotFoundException
- class java.lang.CloneNotSupportedException
    - class java.rmi.server.ServerCloneException
- class java.util.zip.DataFormatException
- class javax.swing.tree.ExpandVetoException
- class java.security.GeneralSecurityException
    - class java.security.cert.CertificateException
        - class java.security.cert.CertificateEncodingException
        - class java.security.cert.CertificateExpiredException
        - class java.security.cert.CertificateNotYetValidException
        - class java.security.cert.CertificateParsingException
    - class java.security.cert.CRLException
    - class java.security.DigestException
    - class java.security.InvalidAlgorithmParameterException
    - class java.security.spec.InvalidKeySpecException
    - class java.security.spec.InvalidParameterSpecException
    - class java.security.KeyException
        - class java.security.InvalidKeyException
        - class java.security.KeyManagementException
    - class java.security.KeyStoreException
    - class java.security.NoSuchAlgorithmException
    - class java.security.NoSuchProviderException
    - class java.security.SignatureException
    - class java.security.UnrecoverableKeyException
- class java.lang.IllegalAccessException
- class java.lang.InstantiationException
- class java.lang.InterruptedException
- class java.beans.IntrospectionException
- class java.lang.reflect.InvocationTargetException
- class java.io.IOException
    - class javax.swing.text.ChangedCharSetException
    - class java.io.CharConversionException
    - class java.io.EOFException
    - class java.io.FileNotFoundException
    - class java.io.InterruptedIOException
    - class java.net.MalformedURLException
    - class java.io.ObjectStreamException
        - class java.io.InvalidClassException
        - class java.io.InvalidObjectException
        - class java.io.NotActiveException
        - class java.io.NotSerializableException
        - class java.io.OptionalDataException
        - class java.io.StreamCorruptedException
        - class java.io.WriteAbortedException
    - class java.net.ProtocolException
    - class java.rmi.RemoteException
        - class java.rmi.AccessException
        - class java.rmi.activation.ActivateFailedException
        - class java.rmi.ConnectException
        - class java.rmi.ConnectIOException
        - class java.rmi.server.ExportException
            - class java.rmi.server.SocketSecurityException
        - class java.rmi.MarshalException
        - class java.rmi.NoSuchObjectException

- class java.rmi.ServerError
- class java.rmi.ServerException
- class java.rmi.ServerRuntimeException
- class java.rmi.server.SkeletonMismatchException
- class java.rmi.server.SkeletonNotFoundException
- class java.rmi.StubNotFoundException
- class java.rmi.UnexpectedException
- class java.rmi.UnknownHostException
- class java.rmi.UnmarshalException

- class java.net.SocketException
  - class java.net.BindException
  - class java.net.ConnectException
  - class java.net.NoRouteToHostException
- class java.io.SyncFailedException
- class java.net.UnknownHostException
- class java.net.UnknownServiceException
- class java.io.UnsupportedEncodingException
- class java.io.UTFDataFormatException
- class java.util.zip.ZipException
  - class java.util.jar.JarException

- class java.security.acl.LastOwnerException
- class java.awt.geom.NoninvertibleTransformException
- class java.lang.NoSuchFieldException
- class java.lang.NoSuchMethodException
- class java.rmi.NotBoundException
- class java.security.acl.NotOwnerException
- class java.text.ParseException
- class java.awt.print.PrinterException
  - class java.awt.print.PrinterAbortException
  - class java.awt.print.PrinterIOException
- class java.security.PrivilegedActionException
- class java.beans.PropertyVetoException
- class org.omg.CORBA.portable.RemarshalException
- class java.lang.RuntimeException
  - class java.lang.ArithmeticException
  - class java.lang.ArrayStoreException
  - class javax.swing.undo.CannotRedoException
  - class javax.swing.undo.CannotUndoException
  - class java.lang.ClassCastException
  - class java.awt.color.CMMException
  - class java.util.ConcurrentModificationException
  - class java.util.EmptyStackException
  - class java.lang.IllegalArgumentException
    - class java.lang.IllegalThreadStateException
    - class java.security.InvalidParameterException
    - class java.lang.NumberFormatException
  - class java.lang.IllegalMonitorStateException
  - class java.awt.geom.IllegalPathStateException
  - class java.lang.IllegalStateException
    - class java.awt.IllegalComponentStateException
    - class java.awt.dnd.InvalidDnDOperationException
  - class java.awt.image.ImagingOpException
  - class java.lang.IndexOutOfBoundsException
    - class java.lang.ArrayIndexOutOfBoundsException
    - class java.lang.StringIndexOutOfBoundsException
  - class java.util.MissingResourceException

- class java.lang.NegativeArraySizeException
- class java.util.NoSuchElementException
- class java.lang.NullPointerException
- class java.awt.color.ProfileDataException
- class java.security.ProviderException
- class java.awt.image.RasterFormatException
- class java.lang.SecurityException
  - class java.security.AccessControlException
  - class java.rmi.RMISecurityException
- class org.omg.CORBA.SystemException
  - class org.omg.CORBA.BAD_CONTEXT
  - class org.omg.CORBA.BAD_INV_ORDER
  - class org.omg.CORBA.BAD_OPERATION
  - class org.omg.CORBA.BAD_PARAM
  - class org.omg.CORBA.BAD_TYPECODE
  - class org.omg.CORBA.COMM_FAILURE
  - class org.omg.CORBA.DATA_CONVERSION
  - class org.omg.CORBA.FREE_MEM
  - class org.omg.CORBA.IMP_LIMIT
  - class org.omg.CORBA.INITIALIZE
  - class org.omg.CORBA.INTERNAL
  - class org.omg.CORBA.INTF_REPOS
  - class org.omg.CORBA.INV_FLAG
  - class org.omg.CORBA.INV_IDENT
  - class org.omg.CORBA.INV_OBJREF
  - class org.omg.CORBA.INV_POLICY
  - class org.omg.CORBA.INVALID_TRANSACTION
  - class org.omg.CORBA.MARSHAL
  - class org.omg.CORBA.NO_IMPLEMENT
  - class org.omg.CORBA.NO_MEMORY
  - class org.omg.CORBA.NO_PERMISSION
  - class org.omg.CORBA.NO_RESOURCES
  - class org.omg.CORBA.NO_RESPONSE
  - class org.omg.CORBA.OBJ_ADAPTER
  - class org.omg.CORBA.OBJECT_NOT_EXIST
  - class org.omg.CORBA.PERSIST_STORE
  - class org.omg.CORBA.TRANSACTION_REQUIRED
  - class org.omg.CORBA.TRANSACTION_ROLLEDBACK
  - class org.omg.CORBA.TRANSIENT
  - class org.omg.CORBA.UNKNOWN
- class java.lang.UnsupportedOperationException
- class java.rmi.server.ServerNotActiveException
- class java.sql.SQLException
  - class java.sql.BatchUpdateException
  - class java.sql.SQLWarning
    - class java.sql.DataTruncation
- class java.util.TooManyListenersException
- class java.awt.datatransfer.UnsupportedFlavorException
- class javax.swing.UnsupportedLookAndFeelException
- class org.omg.CORBA.UserException (implementa org.omg.CORBA.portable.IDLEntity)
  - class org.omg.CosNaming.NamingContextPackage.AlreadyBound (implementa org.omg.CORBA.portable.IDLEntity)
  - class org.omg.CORBA.TypeCodePackage.BadKind
  - class org.omg.CORBA.Bounds
  - class org.omg.CORBA.TypeCodePackage.Bounds

- class org.omg.CosNaming.NamingContextPackage.CannotProceed (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CORBA.ORBPackage.InconsistentTypeCode
- class org.omg.CORBA.DynAnyPackage.Invalid
- class org.omg.CORBA.ORBPackage.InvalidName
- class org.omg.CosNaming.NamingContextPackage.InvalidName (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CORBA.DynAnyPackage.InvalidSeq
- class org.omg.CORBA.DynAnyPackage.InvalidValue
- class org.omg.CosNaming.NamingContextPackage.NotEmpty (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CosNaming.NamingContextPackage.NotFound (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CORBA.PolicyError
- class org.omg.CORBA.DynAnyPackage.TypeMismatch
- class org.omg.CORBA.UnknownUserException
- class org.omg.CORBA.WrongTransaction

- class javax.swing.Timer (implementa java.io.Serializable)
- class java.util.TimeZone (implementa java.lang.Cloneable, java.io.Serializable)
  - class java.util.SimpleTimeZone
- class java.awt.Toolkit
- class javax.swing.ToolTipManager.insideTimerAction (implementa java.awt.event.ActionListener)
- class javax.swing.ToolTipManager.outsideTimerAction (implementa java.awt.event.ActionListener)
- class javax.swing.ToolTipManager.stillInsideTimerAction (implementa java.awt.event.ActionListener)
- class java.awt.font.TransformAttribute (implementa java.io.Serializable)
- class javax.swing.tree.TreePath (implementa java.io.Serializable)
- class org.omg.CORBA.TypeCode (implementa org.omg.CORBA.portable.IDLEntity)
- class org.omg.CORBA.TypeCodeHolder (implementa org.omg.CORBA.portable.Streamable)
- class java.sql.Types
- class java.rmi.server.UID (implementa java.io.Serializable)
- class javax.swing.UIManager (implementa java.io.Serializable)
- class javax.swing.UIManager.LookAndFeelInfo
- class javax.swing.undo.UndoableEditSupport
- class org.omg.CORBA.UnionMember (implementa org.omg.CORBA.portable.IDLEntity)
- class java.net.URL (implementa java.io.Serializable)
- class java.net.URLConnection
  - class java.net.HttpURLConnection
  - class java.net.JarURLConnection
- class java.net.URLDecoder
- class java.net.URLEncoder
- class java.net.URLStreamHandler
- class javax.swing.text.Utilities
- class org.omg.CORBA.ValueMember (implementa org.omg.CORBA.portable.IDLEntity)
- class java.beans.VetoableChangeSupport (implementa java.io.Serializable)
- class javax.swing.text.View (implementa javax.swing.SwingConstants)
  - class javax.swing.text.ComponentView
    - class javax.swing.text.html.FormView (implementa java.awt.event.ActionListener)
    - class javax.swing.text.html.ObjectView
  - class javax.swing.text.CompositeView
    - class javax.swing.text.BoxView
      - class javax.swing.text.html.BlockView
        - class javax.swing.text.html.ListView

- class javax.swing.text.ParagraphView (implementa javax.swing.text.TabExpander)
  - class javax.swing.text.html.ParagraphView
- class javax.swing.text.TableView
- class javax.swing.text.TableView.TableCell (implementa javax.swing.text.TableView.GridCell)
- class javax.swing.text.TableView.TableRow
- class javax.swing.text.WrappedPlainView (implementa javax.swing.text.TabExpander)

- class javax.swing.text.IconView
- class javax.swing.text.LabelView
  - class javax.swing.text.html.InlineView
- class javax.swing.text.PlainView (implementa javax.swing.text.TabExpander)
  - class javax.swing.text.FieldView
    - class javax.swing.text.PasswordView

- class javax.swing.ViewportLayout (implementa java.awt.LayoutManager, java.io.Serializable)
- class java.rmi.dgc.VMID (implementa java.io.Serializable)
- class java.lang.Void
- class java.awt.event.WindowAdapter (implementa java.awt.event.WindowListener)
  - class javax.swing.plaf.basic.BasicToolBarUI.FrameListener
  - class javax.swing.JMenu.WinListener (implementa java.io.Serializable)
- class java.io.Writer
  - class java.io.BufferedWriter
  - class java.io.CharArrayWriter
  - class java.io.FilterWriter
  - class java.io.OutputStreamWriter
    - class java.io.FileWriter
  - class java.io.PipedWriter
  - class java.io.PrintWriter
  - class java.io.StringWriter
- class java.security.cert.X509CRLEntry (implementa java.security.cert.X509Extension)
- class java.util.zip.ZipEntry (implementa java.lang.Cloneable, java.util.zip.ZipConstants)
  - class java.util.jar.JarEntry
- class java.util.zip.ZipFile (implementa java.util.zip.ZipConstants)
  - class java.util.jar.JarFile

Anexo II

Hierarquia de Interfaces

- interface javax.swing.text.AbstractDocument.AttributeContext
- interface javax.swing.text.AbstractDocument.Content
- interface javax.accessibility.Accessible
- interface javax.accessibility.AccessibleAction
- interface javax.accessibility.AccessibleComponent
- interface javax.accessibility.AccessibleSelection
- interface javax.accessibility.AccessibleText
    - interface javax.accessibility.AccessibleHypertext
- interface javax.accessibility.AccessibleValue
- interface java.awt.ActiveEvent
- interface java.awt.Adjustable
- interface java.security.spec.AlgorithmParameterSpec
- interface java.applet.AppletContext
- interface java.beans.AppletInitializer
- interface java.applet.AppletStub
- interface org.omg.CORBA.ARG_IN
- interface org.omg.CORBA.ARG_INOUT
- interface org.omg.CORBA.ARG_OUT
- interface java.sql.Array
- interface javax.swing.text.AttributeSet
    - interface javax.swing.text.MutableAttributeSet
        - interface javax.swing.text.Style
- interface javax.swing.text.AttributeSet.CharacterAttribute
- interface javax.swing.text.AttributeSet.ColorAttribute
- interface javax.swing.text.AttributeSet.FontAttribute
- interface javax.swing.text.AttributeSet.ParagraphAttribute
- interface java.applet.AudioClip
- interface java.awt.dnd.Autoscroll
- interface org.omg.CORBA.BAD_POLICY
- interface org.omg.CORBA.BAD_POLICY_TYPE
- interface org.omg.CORBA.BAD_POLICY_VALUE
- interface java.beans.beancontext.BeanContextChild
    - interface java.beans.beancontext.BeanContext (também implementa java.util.Collection, java.beans.DesignMode, java.beans.Visibility)
        - interface java.beans.beancontext.BeanContextServices (também implementa java.beans.beancontext.BeanContextServicesListener)
- interface java.beans.beancontext.BeanContextChildComponentProxy
- interface java.beans.beancontext.BeanContextContainerProxy
- interface java.beans.beancontext.BeanContextProxy
- interface java.beans.beancontext.BeanContextServiceProvider
- interface java.beans.BeanInfo
    - interface java.beans.beancontext.BeanContextServiceProviderBeanInfo
- interface java.sql.Blob
- interface javax.swing.border.Border
- interface javax.swing.BoundedRangeModel
- interface java.awt.image.BufferedImageOp

- interface javax.swing.text.Caret
- interface javax.swing.CellEditor
  - interface javax.swing.table.TableCellEditor
  - interface javax.swing.tree.TreeCellEditor
- interface java.security.Certificate
- interface java.util.zip.Checksum
- interface java.awt.datatransfer.ClipboardOwner
- interface java.sql.Clob
- interface java.lang.Cloneable
  - interface java.security.acl.AclEntry
  - interface java.text.CharacterIterator
    - interface java.text.AttributedCharacterIterator
- interface java.util.Collection
  - interface java.beans.beancontext.BeanContext (também implementa java.beans.beancontext.BeanContextChild, java.beans.DesignMode, java.beans.Visibility)
    - interface java.beans.beancontext.BeanContextServices (também implementa java.beans.beancontext.BeanContextServicesListener)
  - interface java.util.List
  - interface java.util.Set
    - interface java.util.SortedSet
- interface javax.swing.colorchooser.ColorSelectionModel
- interface javax.swing.ComboBoxEditor
- interface javax.swing.plaf.basic.ComboPopup
- interface java.lang.Comparable
- interface java.util.Comparator
- interface java.awt.Composite
- interface java.awt.CompositeContext
- interface java.sql.Connection
- interface java.net.ContentHandlerFactory
- interface org.omg.CORBA.CTX_RESTRICT_SCOPE
- interface java.beans.Customizer
- interface java.sql.DatabaseMetaData
- interface java.io.DataInput
  - interface java.io.ObjectInput
- interface java.io.DataOutput
  - interface java.io.ObjectOutput
- interface java.beans.DesignMode
  - interface java.beans.beancontext.BeanContext (também implementa java.beans.beancontext.BeanContextChild, java.util.Collection, java.beans.Visibility)
    - interface java.beans.beancontext.BeanContextServices (também implementa java.beans.beancontext.BeanContextServicesListener)
- interface javax.swing.DesktopManager
- interface javax.swing.text.Document
  - interface javax.swing.text.StyledDocument
- interface javax.swing.event.DocumentEvent
- interface javax.swing.event.DocumentEvent.ElementChange
- interface java.sql.Driver
- interface java.security.interfaces.DSAKey
  - interface java.security.interfaces.DSAPrivateKey (também implementa java.security.PrivateKey)
  - interface java.security.interfaces.DSAPublicKey (também implementa java.security.PublicKey)
- interface java.security.interfaces.DSAKeyPairGenerator
- interface java.security.interfaces.DSAParams
- interface javax.swing.text.html.parser.DTDConstants
- interface javax.swing.text.Element
- interface java.util.Enumeration

- interface java.util.EventListener
  - interface java.awt.event.ActionListener
    - interface javax.swing.Action
  - interface java.awt.event.AdjustmentListener
  - interface javax.swing.event.AncestorListener
  - interface java.awt.event.AWTEventListener
  - interface java.beans.beancontext.BeanContextMembershipListener
  - interface java.beans.beancontext.BeanContextServiceRevokedListener
    - interface java.beans.beancontext.BeanContextServicesListener
      - interface java.beans.beancontext.BeanContextServices (também implementa java.beans.beancontext.BeanContext)
  - interface javax.swing.event.CaretListener
  - interface javax.swing.event.CellEditorListener
  - interface javax.swing.event.ChangeListener
  - interface java.awt.event.ComponentListener
  - interface java.awt.event.ContainerListener
  - interface javax.swing.event.DocumentListener
  - interface java.awt.dnd.DragGestureListener
  - interface java.awt.dnd.DragSourceListener
  - interface java.awt.dnd.DropTargetListener
  - interface java.awt.event.FocusListener
  - interface javax.swing.event.HyperlinkListener
  - interface java.awt.event.InputMethodListener
  - interface javax.swing.event.InternalFrameListener
  - interface java.awt.event.ItemListener
  - interface java.awt.event.KeyListener
  - interface javax.swing.event.ListDataListener
  - interface javax.swing.event.ListSelectionListener
  - interface javax.swing.event.MenuDragMouseListener
  - interface javax.swing.event.MenuKeyListener
  - interface javax.swing.event.MenuListener
  - interface java.awt.event.MouseListener
    - interface javax.swing.event.MouseInputListener (também implementa java.awt.event.MouseMotionListener)
  - interface java.awt.event.MouseMotionListener
    - interface javax.swing.event.MouseInputListener (também implementa java.awt.event.MouseListener)
  - interface javax.swing.event.PopupMenuListener
  - interface java.beans.PropertyChangeListener
  - interface javax.swing.event.TableColumnModelListener
  - interface javax.swing.event.TableModelListener
  - interface java.awt.event.TextListener
  - interface javax.swing.event.TreeExpansionListener
  - interface javax.swing.event.TreeModelListener
  - interface javax.swing.event.TreeSelectionListener
  - interface javax.swing.event.TreeWillExpandListener
  - interface javax.swing.event.UndoableEditListener
  - interface java.beans.VetoableChangeListener
  - interface java.awt.event.WindowListener
- interface java.io.FileFilter
- interface java.io.FilenameFilter
- interface java.net.FileNameMap
- interface java.awt.datatransfer.FlavorMap
- interface java.security.Guard
- interface javax.swing.text.Highlighter
- interface javax.swing.text.Highlighter.Highlight

- interface javax.swing.text.Highlighter.HighlightPainter
- interface javax.swing.Icon
- interface java.awt.image.ImageConsumer
- interface java.awt.image.ImageObserver
- interface java.awt.image.ImageProducer
- interface java.awt.im.InputMethodRequests
- interface org.omg.CORBA.portable.InvokeHandler
- interface java.awt.ItemSelectable
    - interface javax.swing.ButtonModel
- interface java.util.Iterator
    - interface java.util.ListIterator
- interface javax.swing.JComboBox.KeySelectionManager
- interface javax.swing.text.Keymap
- interface java.security.spec.KeySpec
- interface java.awt.LayoutManager
    - interface java.awt.LayoutManager2
- interface javax.swing.ListCellRenderer
- interface javax.swing.ListModel
    - interface javax.swing.ComboBoxModel
        - interface javax.swing.MutableComboBoxModel
- interface javax.swing.ListSelectionModel
- interface java.rmi.server.LoaderHandler
- interface java.util.Map
    - interface java.util.SortedMap
- interface java.util.Map.Entry
- interface java.lang.reflect.Member
- interface java.awt.MenuContainer
- interface javax.swing.MenuElement
- interface java.awt.font.MultipleMaster
- interface org.omg.CORBA.Object
    - interface org.omg.CosNaming.BindingIterator (também implementa org.omg.CORBA.portable.IDLEntity)
    - interface org.omg.CORBA.Current
    - interface org.omg.CORBA.DomainManager
    - interface org.omg.CORBA.DynAny
        - interface org.omg.CORBA.DynArray (também implementa org.omg.CORBA.Object)
        - interface org.omg.CORBA.DynEnum (também implementa org.omg.CORBA.Object)
        - interface org.omg.CORBA.DynFixed (também implementa org.omg.CORBA.Object)
        - interface org.omg.CORBA.DynSequence (também implementa org.omg.CORBA.Object)
        - interface org.omg.CORBA.DynStruct (também implementa org.omg.CORBA.Object)
        - interface org.omg.CORBA.DynUnion (também implementa org.omg.CORBA.Object)
        - interface org.omg.CORBA.DynValue (também implementa org.omg.CORBA.Object)
    - interface org.omg.CORBA.DynArray (também implementa org.omg.CORBA.DynAny)
    - interface org.omg.CORBA.DynEnum (também implementa org.omg.CORBA.DynAny)
    - interface org.omg.CORBA.DynFixed (também implementa org.omg.CORBA.DynAny)
    - interface org.omg.CORBA.DynSequence (também implementa org.omg.CORBA.DynAny)
    - interface org.omg.CORBA.DynStruct (também implementa org.omg.CORBA.DynAny)
    - interface org.omg.CORBA.DynUnion (também implementa org.omg.CORBA.DynAny)
    - interface org.omg.CORBA.DynValue (também implementa org.omg.CORBA.DynAny)
    - interface org.omg.CORBA.IDLType (também implementa org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.IRObject)
    - interface org.omg.CORBA.IRObject (também implementa org.omg.CORBA.portable.IDLEntity)
        - interface org.omg.CORBA.IDLType (também implementa org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)

    - interface org.omg.CosNaming.NamingContext (também implementa org.omg.CORBA.portable.IDLEntity)
    - interface org.omg.CORBA.Policy
- interface java.io.ObjectInputValidation
- interface java.io.ObjectStreamConstants
- interface java.util.Observer
- interface java.awt.font.OpenType
- interface java.security.acl.Owner
    - interface java.security.acl.Acl
- interface java.awt.print.Pageable
- interface java.awt.PaintContext
- interface java.awt.geom.PathIterator
- interface java.security.acl.Permission
- interface javax.swing.text.Position
- interface java.security.Principal
    - interface java.security.acl.Group
- interface java.awt.print.Printable
- interface java.awt.print.PrinterGraphics
- interface java.awt.PrintGraphics
- interface org.omg.CORBA.PRIVATE_MEMBER
- interface java.security.PrivilegedAction
- interface java.security.PrivilegedExceptionAction
- interface java.beans.PropertyEditor
- interface org.omg.CORBA.PUBLIC_MEMBER
- interface java.awt.image.RasterOp
- interface java.sql.Ref
- interface java.rmi.registry.RegistryHandler
- interface java.rmi.Remote
    - interface java.rmi.activation.ActivationInstantiator
    - interface java.rmi.activation.ActivationMonitor
    - interface java.rmi.activation.ActivationSystem
    - interface java.rmi.activation.Activator
    - interface java.rmi.dgc.DGC
    - interface java.rmi.registry.Registry
- interface java.rmi.server.RemoteCall
- interface java.awt.image.renderable.RenderableImage
- interface java.awt.image.RenderedImage
    - interface java.awt.image.WritableRenderedImage
- interface java.awt.image.renderable.RenderedImageFactory
    - interface java.awt.image.renderable.ContextualRenderedImageFactory
- interface javax.swing.Renderer
- interface org.omg.CORBA.portable.ResponseHandler
- interface java.sql.ResultSet
- interface java.sql.ResultSetMetaData
- interface java.rmi.server.RMIClientSocketFactory
- interface java.rmi.server.RMIFailureHandler
- interface java.rmi.server.RMIServerSocketFactory
- interface javax.swing.RootPaneContainer
- interface javax.swing.tree.RowMapper
- interface java.lang.Runnable
- interface javax.swing.Scrollable
- interface javax.swing.ScrollPaneConstants
- interface java.io.Serializable
    - interface java.io.Externalizable
        - interface java.rmi.server.RemoteRef
            - interface java.rmi.server.ServerRef

- interface org.omg.CORBA.portable.IDLEntity
  - interface org.omg.CosNaming.BindingIterator (também implementa org.omg.CORBA.Object)
  - interface org.omg.CORBA.IDLType (também implementa org.omg.CORBA.IRObject, org.omg.CORBA.Object)
  - interface org.omg.CORBA.IRObject (também implementa org.omg.CORBA.Object)
    - interface org.omg.CORBA.IDLType (também implementa org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
  - interface org.omg.CosNaming.NamingContext (também implementa org.omg.CORBA.Object)
- interface java.security.Key
  - interface java.security.PrivateKey
    - interface java.security.interfaces.DSAPrivateKey (também implementa java.security.interfaces.DSAKey)
    - interface java.security.interfaces.RSAPrivateKey
      - interface java.security.interfaces.RSAPrivateCrtKey
  - interface java.security.PublicKey
    - interface java.security.interfaces.DSAPublicKey (também implementa java.security.interfaces.DSAKey)
    - interface java.security.interfaces.RSAPublicKey
- interface java.awt.Shape
- interface javax.swing.SingleSelectionModel
- interface java.rmi.server.Skeleton
- interface java.net.SocketImplFactory
- interface java.net.SocketOptions
- interface java.sql.SQLData
- interface java.sql.SQLInput
- interface java.sql.SQLOutput
- interface javax.swing.undo.StateEditable
- interface java.sql.Statement
  - interface java.sql.PreparedStatement
    - interface java.sql.CallableStatement
- interface org.omg.CORBA.portable.Streamable
- interface java.awt.Stroke
- interface java.sql.Struct
- interface javax.swing.SwingConstants
- interface javax.swing.text.TabableView
- interface javax.swing.text.TabExpander
- interface javax.swing.table.TableCellRenderer
- interface javax.swing.table.TableColumnModel
- interface javax.swing.table.TableModel
- interface java.awt.image.TileObserver
- interface java.awt.datatransfer.Transferable
- interface java.awt.Transparency
  - interface java.awt.Paint
- interface javax.swing.tree.TreeCellRenderer
- interface javax.swing.tree.TreeModel
- interface javax.swing.tree.TreeNode
  - interface javax.swing.tree.MutableTreeNode
- interface javax.swing.tree.TreeSelectionModel
- interface javax.swing.UIDefaults.ActiveValue
- interface javax.swing.UIDefaults.LazyValue
- interface javax.swing.plaf.UIResource
- interface javax.swing.undo.UndoableEdit
- interface java.rmi.server.Unreferenced
- interface org.omg.CORBA.UNSUPPORTED_POLICY

- interface org.omg.CORBA.UNSUPPORTED_POLICY_VALUE
- interface java.net.URLStreamHandlerFactory
- interface javax.swing.text.ViewFactory
- interface java.beans.Visibility
  - interface java.beans.beancontext.BeanContext (também implementa java.beans.beancontext.BeanContextChild, java.util.Collection, java.beans.DesignMode)
    - interface java.beans.beancontext.BeanContextServices (também implementa java.beans.beancontext.BeanContextServicesListener)
- interface org.omg.CORBA.VM_ABSTRACT
- interface org.omg.CORBA.VM_CUSTOM
- interface org.omg.CORBA.VM_NONE
- interface org.omg.CORBA.VM_TRUNCATABLE
- interface javax.swing.WindowConstants
- interface java.security.cert.X509Extension

Anexo III

Exercícios

I. GUIs

1. Você deve fazer um programa com uma GUI que implemente uma janela na qual encontramos: (1) 1 rótulo (“Entre com seu texto: ”) associado a 1 caixa de texto na qual espera-se a entrada de um texto qualquer; (2) um botão (“Processar”) que, quando acionado, calcula a freqüência dos caracteres presentes no texto e os prepara para serem exibidos juntamente com suas freqüências; (3) 2 rótulos (“Caractere: X (999)” e “Freqüência: 999”) que, respectivamente, exibirão um caractere presente no texto (com seu código ASCII) e sua freqüência; e (4) 2 botões (“Anterior” e “Próximo”) que, respectivamente, retrocederão e avançarão um caractere. Pede-se implementar uma classe com a finalidade descrita acima e nela:
   - Implementar uma função que preenche com zero todas as posições de um vetor global a nível de classe de 256 inteiros;
   - Implementar uma função que recebe como parâmetro um caractere e incrementa a posição correspondente a seu código ASCII de um vetor de inteiros global em nível de classe;
   - Implementar um construtor que monte adequadamente a interface de sua aplicação; e
   - Implementar uma função tratadora de eventos que trate adequadamente os eventos de sua aplicação.

   Para efeito de isolar as palavras do texto, deve-se considerar que palavras são quaisquer seqüência de caracteres alfabéticos.
2. Realize as transformações necessárias para fazer do programa do exercício anterior uma Applet.
3. O objetivo deste exercício é implementar um jogo muito conhecido entre as crianças; trata-se do jogo “Papel, Tesoura ou Pedra”. O jogo é muito simples; 2 jogadores jogam; os 2, frente a frente, escondem suas mãos atrás das costas e dizem “Papel, Tesoura ou Pedra”; após isso, mostram suas mãos que podem estar em 1 de 3 possíveis configurações: (a) mão aberta (significa papel; (b) mão fechada com os dedos indicador e médio estendidos (significa tesoura); e (c) mão fechada (significa pedra); tesoura ganha de papel, papel ganha de pedra, pedra ganha de tesoura, as demais combinações representam empate. Pede-se:
   - Classe JogoPTP:
     - Declare a classe JogoPTP e as estruturas internas para representá-la;

- Implemente um método público de instância chamado Jogada que recebe como parâmetro um número inteiro que representa o tipo da jogada (0 representa papel, 1 representa tesoura e 2 representa pedra);

  Este método deverá sortear um número aleatório entre 0 e 2 inclusive, que será a jogada do programa e que será o retorno deste método, compará-lo com o parâmetro recebido e ajustar flags internos que indicam quem ganhou, se houve empate ou se a jogada representada pelo parâmetro foi inválida;

- Implemente um método público de instância chamado JogadaInvalida, que retorna true, caso a chamada mais recente do método Jogada tenha representado uma jogada inválida, e false, caso contrário;

- Implemente um método público de instância chamado DeuEmpate, que retorna true, caso a chamada mais recente do método Jogada tenha resultado em empate, e false, caso contrário;

- Implemente um método público de instância chamado JogGanhou, que retorna true, caso a chamada mais recente do método Jogada tenha resultado na vitória do jogador, e false, caso contrário;

- Implemente um método público de instância chamado ProgGanhou, que retorna true, caso a chamada mais recente do método Jogada tenha resultado na vitória do programa, e false, caso contrário;

- Classe TelaPTP:

  - Declare a classe TelaPTP e as estruturas internas para representá-la;

  - Implemente um construtor público que monta uma tela gráfica na qual constam: (a) um Label (“0 = Papel / 1 = Tesoura / 2 = Pedra); (b) um Label (“Sua Jogada: ”); (c) um TextField no qual o jogador indicará sua jogada; (d) um Label (“Minha Jogada: X”); (e) um Button (“Efetuar Jogada”); e (f) um Label que servirá como área de exibição de mensagens;

  - Implemente os métodos necessários ao tratamento do evento de click no Button (“Efetuar Jogada”);

  - Implemente os métodos necessários ao atendimento da solicitação de fechamento desta janela;

- Classe PTP:

  - Declare uma classe executável chamada PTP que executa as ações necessárias à execução do jogo “Papel, Tesoura ou Pedra”.

4. Realize as transformações necessárias para fazer do programa do exercício anterior uma Applet.

Anexo IV

Bibliografia

SUN Microsystems, "The Java™ Tutorial", (http://java.sun.com/docs/books/tutorial/).

SUN Microsystems, "The Java™ Language Specification", (http://java.sun.com/docs/books/jls/index.html).

SUN Microsystems, "The Java™ API Specification", (http://java.sun.com/products/jdk/1.2/docs/api/index.html).

SUN Microsystems, "The Java™ VM Specification", (http://java.sun.com/docs/books/vmspec/index.html).

Naughton, Patrick, "Dominando o Java", Makron Books, 1997.

Morrison, M.; "Java Unleashed", December, J.; et alii; SAMS Publishing.

Naughton, P.; "Dominando o Java"; Makron Books.

Linden, P.; "Just Java"; Makron Books.

Fraizer, C.; e Bond, J.; "API Java: Manual de Referência"; Makron Books.

Thomas, M.D.; Patel, P.r.; et alii; "Programando em Java para a Internet"; Makron Books.

Ritchey, T.; "Programando Java e JavaScript"; Quark Editora.