



Java

Treinamento Java



Instrutores :

Marcelo Araujo Franco

Fátima Pires

1



Java

Desenvolvedores do Material

- ✓ Teoria e Prática Java
 - Edmilson Bellini Chiavegatto
(Analista de Sistemas)

- ✓ Java e Orientação a Objetos
 - Fátima Pires
(Analista de Sistemas)

2



Java

Histórico

- ✓ Projeto “Green” (1992)
 - 4 integrantes
 - Software para eletro-domésticos
 - Características do Software
 - ♦ Mínimo uso de memória
 - ♦ Mínimo preço
- ✓ Necessidade de um nome
- ✓ Por que surgiu o Java ?
 - C ++ simplificado
 - Suprir software para eletro-domésticos

3



Java

Histórico (cont.)

- ✓ Desenvolvido pela Sun Microsystems
- ✓ Estabelece Novo Paradigma de Programação
 - Totalmente Aberta
 - Independente de Plataforma e Sistema Operacional
- ✓ Usuários afincos do Java :
 - CSX (maior companhia ferroviária dos EUA) montou rede Java;
 - J.P. Morgan (banco de investimentos);
 - Hong Kong Telecom (operações de rede de TV interativa) ;
 - Fannie Mac (maior empresa americana de hipotecas);
 - American Express (serviços financeiros);
 - Banco Itaú usando em suas Intranets e outras.
- ✓ Hoje 400.000 programadores escrevem em Java
- ✓ 25% das companhias americanas com mais de 5000 funcionários pretendem usar Java este ano.

4



Java

Parte I

Orientação a Objetos e Java

5



Java

Objetivo

- ✓ Orientação a Objetos em Java
- ✓ Conceitos básicos de Orientação a Objetos
- ✓ Introdução à linguagem Java

6



Java

Roteiro

- ✓ Paradigma de Orientação a Objetos
- ✓ Objetos
- ✓ Encapsulamento
- ✓ Mensagens
- ✓ Métodos
- ✓ Abstração
- ✓ Classes

7



Java

Roteiro (cont.)

- ✓ Generalização
- ✓ Herança
- ✓ Agregação
- ✓ Polimorfismo

8



Java

Paradigma

“Paradigma é um conjunto de regras que estabelecem fronteiras e descreve como resolver os problemas dentro destas fronteiras.

Os paradigmas influenciam nossa percepção; ajudam-nos a organizar e a coordenar a maneira como olhamos para o mundo...”

Reengenharia - Reestruturando a Empresa
Daniel Morris e Joel Brandon

9



Java

Orientação a Objetos

O termo orientação a objetos significa organizar o mundo real como uma coleção de objetos que incorporam estrutura de dados e comportamento.

10



Java

Orientação a Objetos

Origens

- ✓ Linguagens de Programação - Simula, Smalltalk, Flavours, Objective C, C++,...
- ✓ Inteligencia Artificial - frames
- ✓ Banco de Dados - modelos semânticos de dados

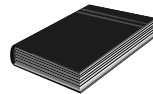
11



Java

Objetos

Coisas tangíveis



“A Profecia Celestina”

Incidente
(evento/ocorrência)



Olimpíada de Atlanta

Interação
(transação/contrato)



Minha consulta

12



Java

Objetos

Objetos são pacotes de software compostos de dados e procedimentos que atuam sobre estes dados.

Os procedimentos são também conhecidos como *métodos* e determinam o comportamento do objeto

Objeto = dado + método



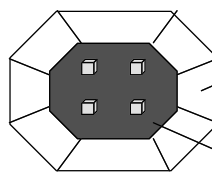
Objeto = estado + comportamento



Java

Objetos

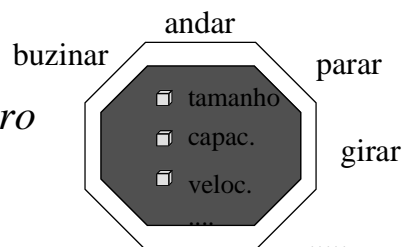
Um objeto



comportamento/
métodos/procedimentos

propriedades/dados/
variáveis

O objeto Carro

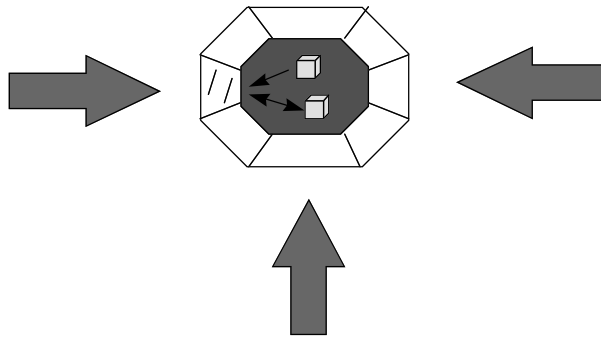




Java

Objetos

Todo o acesso aos dados ou propriedades do objeto é feito através da sua interface



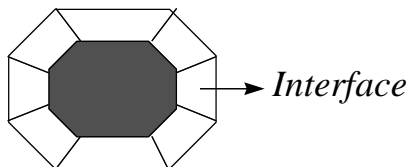
15



Java

Encapsulamento

Encapsulamento é definido como uma técnica para minimizar interdependências entre “módulos” através da definição de interfaces externas.



Mudanças na implementação de uma classe que preserve a *interface externa* não afeta outras definições de classes.

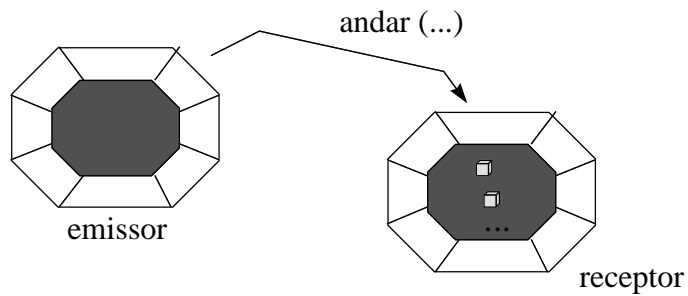
16



Java

Mensagens

Objetos interagem e comunicam-se através de *mensagens...*



Mensagem para um carro

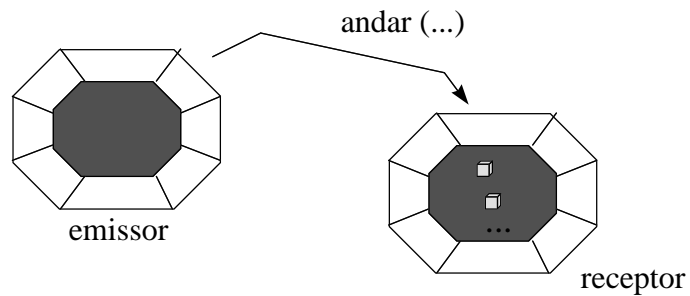
17



Java

Métodos

...as mensagens identificam os *métodos* a serem executados no objeto receptor



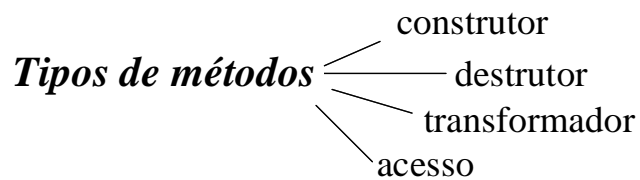
18



Java

Métodos

O que um determinado método pode fazer com os dados do objeto ?



19



Java

Exercício

Caracterização de Objetos

- ✓ Dê 3 exemplos de Objetos
- ✓ Para cada um deles, sugira alguns métodos pertinentes

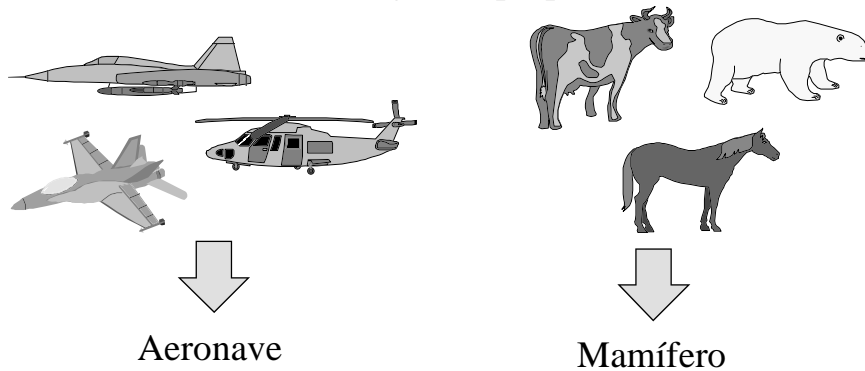
20



Java

Abstração

Focalizar o essencial, ignorar propriedades acidentais



A abstração deve ser sempre com algum objetivo, porque o objetivo determina o que é e o que não é importante.

21

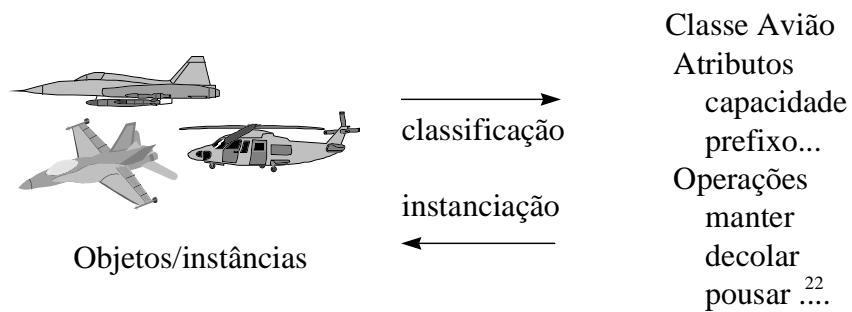


Java

Classes

Uma classe de objetos descreve um grupo de objetos

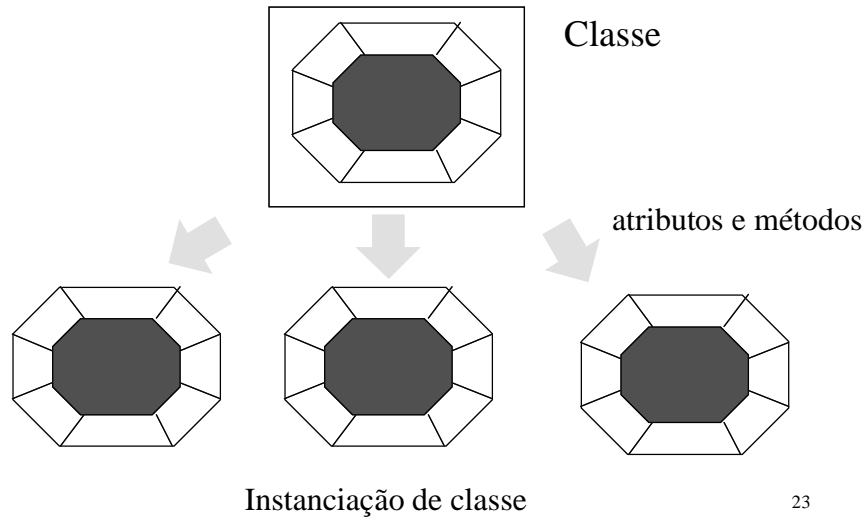
- com propriedades semelhantes
- comportamentos semelhantes
- relacionamentos comuns com outros objetos





Java

Classes

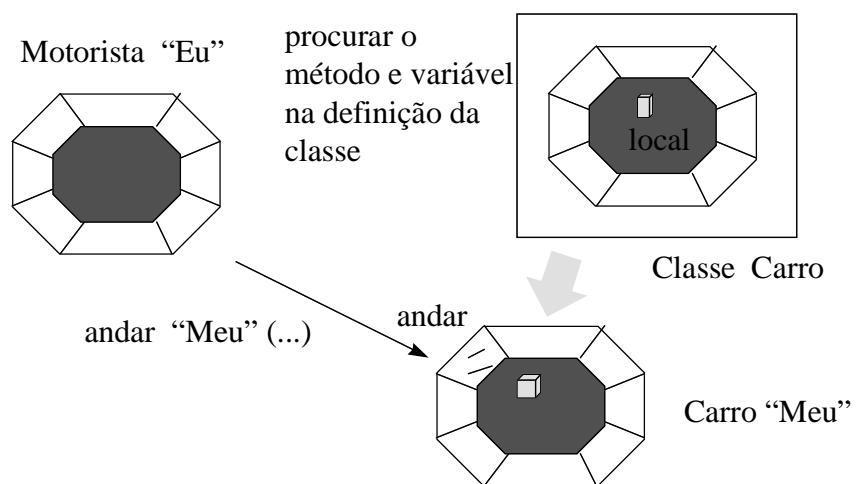


23



Java

Comunicação entre objetos



24



Java

Exercício de Abstração

O que os objetos em cada uma destas listas tem em comum ?

a) 1-microscópio 2-óculos 3-telescópio 4-binóculo

Exemplo: 1,2,3 e 4 - melhoram a visão de alguma forma

2,4 - usam os dois olhos

3,4 - para ver as coisas de longe

1 - para ver as coisas pequenas

2 - aumenta ou diminui dependendo do problema da visão

b) 1-barraca 2-caverna 3-barracão 4-garagem 5-celeiro

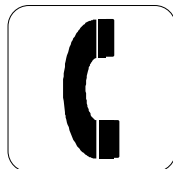
6-casa 7-edifício

25



Java

Vamos respirar.....



26



Java

Exemplo de Definição de Classe (atributos)

```
....

public class Morador...
{String nomeCompleto;
  String apartamento;
  String telefone;
  int   anoChegada;

.....
```

27



Java

Exemplo de Definição de Classe (métodos)

```
public class Morador...
{....
public morador(String no, String ap,
String te, int an)

{ nomeCompleto = no;
  apartamento  = ap;
  telefone     = te;
  anoChegada   = an;
}

public int permanencia()
  { return (1997 - anoChegada); }
}
```

28



Java

Exemplo de Instanciação de Classe

```
...
Morador a;
....
a = new morador("Fatima", "101", "257-2011",
    1992);
...
```

29



Java

Exemplo

Acionando Métodos com Mensagens (I)

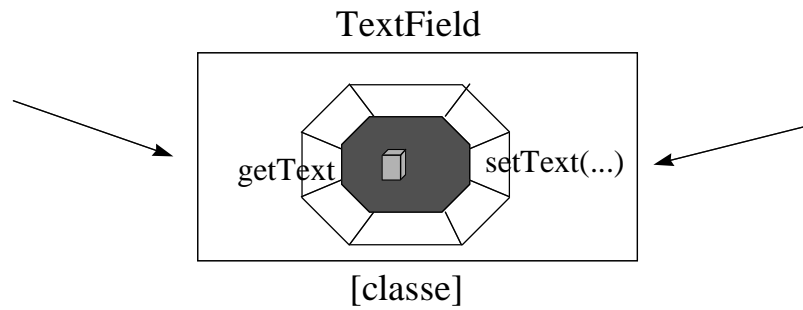
```
.....
Morador a;
int p;
....
a = new morador("Fatima", "101", "257-2011",
    1992);
....
p = a.permanencia(); // acionando o método
                    // permanencia para o
                    // objeto definido em a
    |
    |— indica o envio de mensagem para o
        objeto a
....
```

30



Java

Outro Exemplo de Envio de Mensagens



31



Java

Exemplo de Envio de Mensagens

tf Exemplo 1

[objeto da classe TextField]

```
TextField tf;  
String s;  
..  
tf = new TextField(10);  
tf.setText("Exemplo 1");  
..  
s = tf.getText();
```

32



Java

Relacionamentos entre Classes

- ✓ Generalização
- ✓ Herança
- ✓ Agregação
- ✓ Polimorfismo (overriding, overloading, late binding)

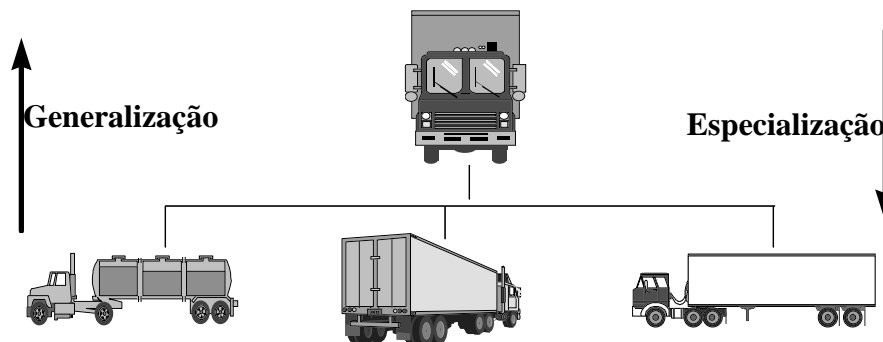
33



Java

Generalização/Especialização

Generalização é o relacionamento entre uma classe e uma ou mais **versões refinadas** dessa classe



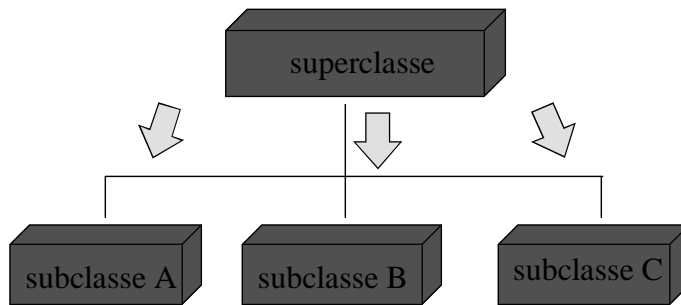
Generalização é a abstração que permite **compartilhar** semelhanças entre classes, preservando suas diferenças

34



Java

Hierarquia de Classes



Classes derivadas

35

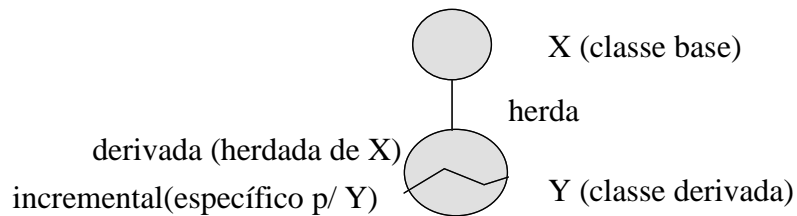


Java

Herança

Uma classe derivada herda as propriedades e métodos da classe pai, mas pode:

- adicionar novos métodos
- estender os atributos
- redefinir a implementação de métodos existentes

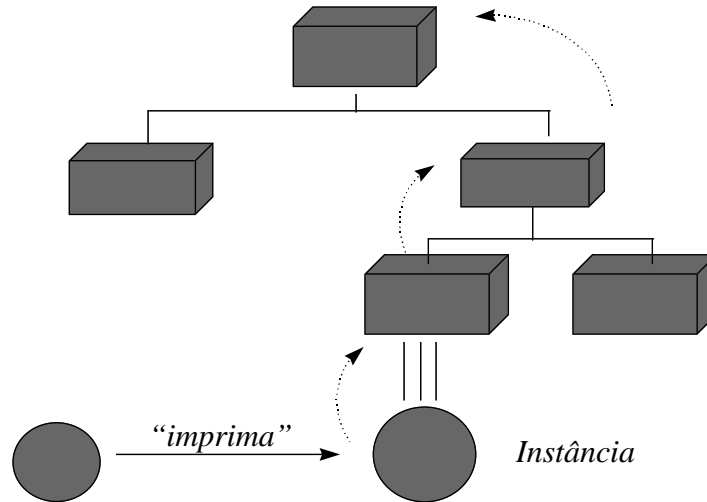


36



Java

Localizando Métodos e Atributos na Hierarquia



37



Java

Exemplo de Herança

```
import morador;  
  
public class morador_inq extends morador  
{int aluguel;  
  
public morador_inq(String no, String ap,  
                    String tel, int an, int va)  
{super(no, ap, tel, an);  
  aluguel = va;  
}  
}
```

38



Java

Exemplo de Herança

Acessando atributos de subclasses

```
public class aplher extends Object
{static morador m;
  static morador_inq mi;

public static void main(String[] args)
{m = new morador("Fatima", "100", "239", 1990);
  mi= new morador_inq("Rey", "101", "234", 1991,
                    200);

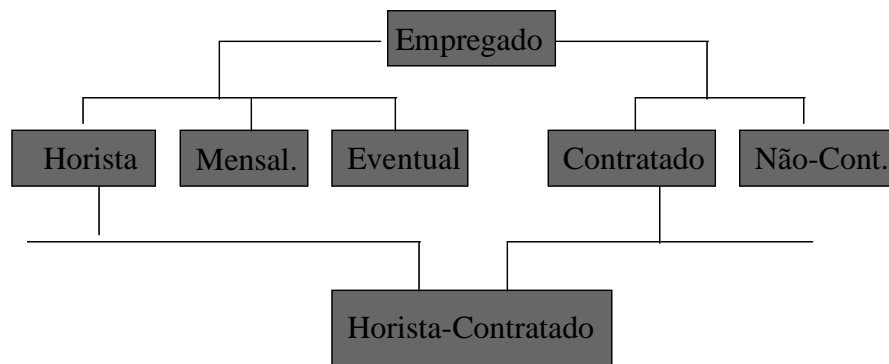
  System.out.println(m.nomeCompleto);
  System.out.println(mi.nomeCompleto);
  System.out.println(mi.aluguel);
}}
```

39



Java

Herança Múltipla



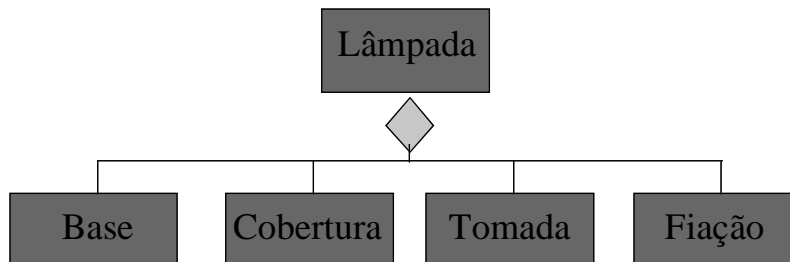
40



Java

Agregação

Um objeto agregado é “feito” de componentes



Agregação Fixa

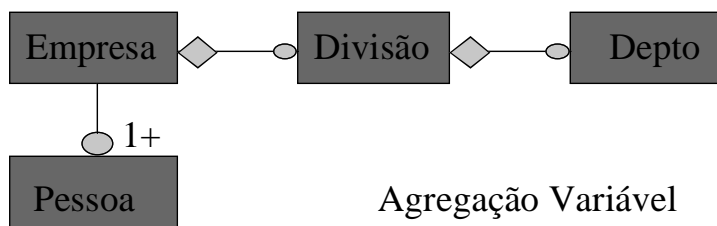
41



Java

Agregação

Um objeto agregado é “feito” de componentes



Agregação Variável

42



Java

Java - Agregação

Exemplo de objeto composto:

```
public class material extends Object
{String rotulo;
 Boolean emCaixa;
 int anoEstocagem;
 double valor;
 Morador proprietario;

public material (....)
....
```

43



Java

Java - Agregação

Exemplo de objeto composto (cont.):

```
public class material extends Object
{....
public material (String ro, double va,
                boolean em, Morador pro, int an)
{rotulo = ro;          valor = va;
 emCaixa = em;        proprietario = pro;
 anoEstocagem = an;
}
public int permanencia()
{ return (1997 - anoEstocagem); }
}
```

44



Java

Exercício - Instanciação de Objeto

Instanciar 3 materiais (mt1, mt2, mt3) da classe Material cada um deles pertencendo a cada um dos seguintes moradores instanciados (a, j, al)

45



Java

Exercício - Acesso aos Atributos dos Objetos

Dê a expressão que indica o acesso ao atributo *nomeCompleto* do proprietário do material identificado por *mt*.

46



Java

Instanciação de Objeto

```
Material mt1, mt2, mt3;  
...  
mt1 = new material("mala",110.00,false,a,1992);  
mt2 = new material("fraseira",50.00,false,j,1990);  
mt3 = new material("furadeira",150.00,true,al,1991);  
....
```

47



Java

Acesso aos Atributos dos Objetos

```
Material mt;  
...  
mt.proprietario.nomeCompleto;  
....
```

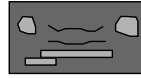
48



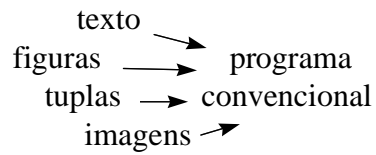
Java

Overloading/Overriding

Ambiente Convencional



tela multimídia



```

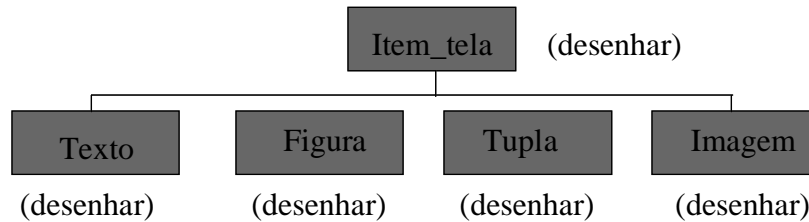
begin case of type(x)
  texto:  desenhar_texto(x)
  imagem: desenhar_imagem(x)
  tupla:  desenhar_tupla(x)
  ....
end
end
  
```



Java

Overloading/Overriding

Ambiente OO



- . redefinição da operação de desenhar (“overriding”)
- . desenhar - mesmo nome para 3 programas (“overloading”)

For x in X do desenhar(x); */ “late binding”



Java

Java - Polimorfismo

Exemplo:

```
public class morador extends Object
{..

public morador (...) [método construtor]
{....}

public int permanencia()
{ return (1997 - anoChegada); }
//há quanto tempo o
// morador reside no
// condomínio
.. }
```

51



Java

Java - Polimorfismo

Exemplo:

```
public class material extends Object
{..
public material (...)
{....proprietario = ....}

public int permanencia()
{ return (1997 - anoEstocagem); } //há quanto
//tempo o material
//está estocado

public double taxaMensal()
{ return valor/(proprietario.permanencia() -
permanencia() ) * 0.01;
}
..}
```

52



Java

Programando em Java

- ✓ Entendimento do Problema
- ✓ Identificação das classes necessárias (atributos e métodos) - classes Java, classes próprias não persistentes, classes de Banco de Dados
- ✓ Definição das aplicações (lógica e interface) - Java “puro”, applets, Java Script
- ✓ Construção (ambiente, Web, Biblioteca Java, sites de consulta, suporte)

53



Java

Parte 2

Teoria e Prática Java

54



Java

Bibliografia

- ✓ Aprendendo Java
 - Programação na Internet
 - Autor Américo Damasceno Jr.
 - Editora Érica Ltda
- ✓ The Java Language Environment
 - A White Paper
 - Jasmes Gosling
 - Henry McGilton
 - SUn Microsystems Computer Company
- ✓ Páginas de Internet
- ✓ Reportagens de Revistas sobre atualidades envolvendo Java

55



Java

Características Java

- ✓ Linguagem Orientada a Objetos (Reuso)
- ✓ Semelhante ao C ++
- ✓ C ++ --
- ✓ Gera Bytecodes
 - Interpretada
 - Alta Performance
- ✓ Segurança
 - Endereçamento Restrito
 - Objetos Assinados
- ✓ Aplicação Carregada Localmente

56



Java

Características Java (cont.)

- ✓ Aplicações Personalizadas
- ✓ Independência de Arquitetura
 - Neutra
 - Distribuída
 - ◆ Funciona em Diferentes Máquinas
- ✓ Não há Herança Múltipla
- ✓ Não há Overloading de Operadores (Sobrecarga)
- ✓ Não há Aritmética de Ponteiros
- ✓ Inclui Tratamento de Exceções
- ✓ Garbage Collector

57



Java

Conceitos || Java Script

- ✓ Primeira Versão do Java
- ✓ Aplicação Interna ao HTML
- ✓ Interpretada
- ✓ Não havia o Conceito de ByteCodes
- ✓ Ex.

```
<script language = "Java Script"
    Function -----
    { .....
    }
</script>
```

58



Java

Conceitos || Java Script (cont.)

✓ Ex.

```
<SCRIPT LANGUAGE="JavaScript">
function scrollit_r2l(seed)
{
    var m1 = " Bem Vindo à Página minha Página - Excelente Escolha !!!";
    var msg = m1;
    var out = " ";
    var c = 1;
    if (seed > 50)
    {
        seed--;
        var cmd="scrollit_r2l(" + seed + ")";
        timerTwo=window.setTimeout(cmd,50);
    }
    else if (seed <= 50 && seed > 0)
    {
        for (c=0 ; c < seed ; c++)
            out+=" ";
        out+=msg;
        seed--;
        var cmd="scrollit_r2l(" + seed + ")";
```

59



Java

Conceitos || Java Script (cont.)

```
seed--;
var cmd="scrollit_r2l(" + seed + ")";
window.status=out;
timerTwo=window.setTimeout(cmd,50);
}
else if (seed <= 0) {
    if (-seed < msg.length) {
        out+=msg.substring(-seed,msg.length);
        seed--;
        var cmd="scrollit_r2l(" + seed + ")";
        window.status=out;
        timerTwo=window.setTimeout(cmd,50);
    }
    else { window.status=" ";
        timerTwo=window.setTimeout("scrollit_r2l(50)",75);
    }
}
}
</SCRIPT>
```

60



Java

Conceitos || Applet

- ✓ Aplicação Executada quando se Chama Página WWW
- ✓ É Carregada na Máquina do Cliente
- ✓ Restringe-se a uma Determinada Área (Janela)
- ✓ Ex.

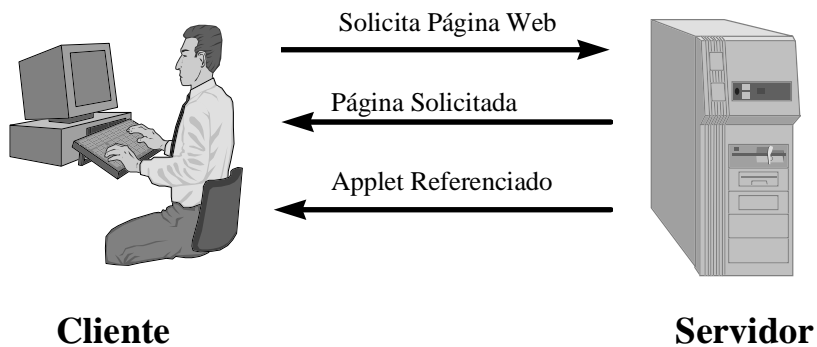
```
<applet code = "ap1.class"  
        codebase = "http://www.unicamp.br/~edmilson"  
        lign=left ou <right,top,middle,bottom>  
        width=300  
        height=100  
<param name=tamanho value = 30>  
<param name=fontevalue value "Times Roman">  
</applet>
```

61



Java

Funcionamento Applet Java



62



Java

Conceitos || Método

✓ Determinada Ação que o Objeto pode ter ao ser Ativado

✓ Ex.

Funcionario f 1;

f 1.mtTempoAposentadoria();

63



Java

Conceitos || Método Construtor

✓ Método Ativado ao ser Criado o Objeto

✓ Útil para Inicializar Propriedades na Criação

✓ Tem o mesmo Nome da Classe

✓ Caso não Exista será Ativado um Método Construtor Default

64



Java

Conceitos || Método Construtor (cont.)

✓ Ex.

```
import java .applet .*;
public class apexemplo extends Applet
{
    int idade;
    String endereco;
    public apexemplo (int parm1, String parm2)
    {
        this.idade    = parm1;
        this.endereco = parm2;
    }
}
```

✓ Como é a chamada deste Método ?

65



Java

Ambiente de Desenvolvimento

- ✓ Configuração Mínima
 - 486 DX/33
 - Windows 95, Windows NT 4.0, Solaris, MacIntosh, Unix
 - 16 Mb de Ram
- ✓ Gerador de Bytecodes
 - Javac <programa.java>. Ex: javac prog1.java
Será gerado um arquivo de extensão (class). Ex : prog1.class
- ✓ Execução
 - Java Puro
 - ♦ java <nome do arquivo de extensão class>. Ex. java prog1
 - Java Applet
 - ♦ appletviewer <nome do arquivo de extensão html> .
 - ♦ Abrir no Browser o arquivo html que faz chamada a uma applet java
- ✓ Depuração
 - jdb <nome.class>

66



Java

Programa Java Applet

✓ Ex.

```
import java.applet.*
import java.awt.*
public class hello extends Applet
{
    Panel p1;
    Label l1;
    public hello ()
    {
        p1 = new Panel ();
        add (p1);
        l1 = new Label ();
        p1.add (l1);
        l1.setText ("Olá Internet !");
    }
}
```

67



Java

Programa Java Puro

✓ Ex.

```
public class OlaInternet
{
    public static void main (String[] args)
    {
        System.out.println("Olá Internet !!!");
    }
}
```

- ✓ Digitar o exemplo de programa Java Applet , compilá-lo e rodá-lo
- ✓ Digitar o exemplo de programa Java Puro, compilá-lo e rodá-lo

68



Java

Comandos || Fundamental/Sequencial

- ✓ Comandos fundamentais
 - Todo comando deve terminar por um ponto e vírgula;
 - `y=0;` // atribuição
 - `Executa();` // invocação

- ✓ Sequência
 - `comando1;`
 - `comando2;`

69



Java

Comandos || Condição

✓ Condição

- **if** (expressão-booleana) // se expressão verdadeira
 comando; // a expressão booleana deve vir entre parênteses
- else** // opcional
 comando; // se expressão falsa

```
public class figuras extends Object
{
    public void main(String[] args)
    {
        int[] x = new int[Integer.parseInt(args[0])];
        if (x.length < 5)
            System.out.println("tamanho array = "+x.length);
        else
        {
            System.out.println("Erro !!!!");
            x[2]=10;
            x[3]=15;
        }
    }
}
```

- ✓ **Digite este exemplo e o faça funcionar**

70



Java

Comandos || Condição

✓ Evitando if em Cascata

```
int x;  
switch (x)  
{  
    case 0 : System.out.println("sexo é masculino");  
        break;  
    case 1 : System.out.println("sexo é feminino");  
        break;  
    default : System.out.println("sexo é indefinido");  
        break;  
}
```

- ✓ Ps. Pode-se fazer switch com os tipos byte, char, short, int, long.



Java

Comandos || Iteração

✓ Enquanto ..

```
int raio = 0;  
while (++raio < 10)  
    figuras.desenha_circulo(0,0,raio);
```

✓ faça... Enquanto (Executa ao menos uma vez)

```
int i=10;  
do  
    figuras.desenha_circulo(0,0,i);  
while (++i < 10);
```

✓ for (Super While)

```
for (int i=0;i < 10; ++i)  
    window.desenha_reta(10,20,50,i);
```



Java

Léxico

- ✓ Comentários
 - `int x = 0; // comentário de linha`
 - `/*`
função : desenhar uma reta conforme parâmetros
parâmetros : x1,y1 --> coordenadas iniciais
x2,y2 --> coordenadas finais
 - `*/` **comentário de mais de uma linha**
 - `class Figura {`
 - `/**` este tipo é **utilizado** para se **gerar documentação Java**
 - `@see` Área
 - `@version` 1.0
 - `@author` Edmilson Bellini Chiavegatto
 - `*/` }
 - **javadoc** <nome.java>
 - ◆ converte em documentação HTML

73



Java

Packages Principais

- ✓ `java.lang`
 - Pacote em que não é necessário se dar um import
 - Principais classes : Boolean, Character, Double, Float, Integer, Long, Math, Object, String, System, Thread
- ✓ `java.io`
 - Pacote que permite manipulação de Streams lendo ou gravando em arquivos e outros
 - Principais classes : DataInputStream, FileInputStream, FileOutputStream, PrintStream
- ✓ `java.util`
 - Pacote que provê uma miscelânea de classe úteis incluindo estrutura de dados, time, date, geração de números randômicos, etc..
- ✓ `java.net`
 - Pacote que provê suporte a redes, incluindo-se URL'S, TCP Sockets, UDP Sockets, Endereços IP, etc..

74



Java

Packages Principais (cont.)

- ✓ java.awt
 - Pacote que provê um conjunto de manipulações de interfaces para o usuário tais como windows, caixas de diálogos, botões, cores, checkboxes, listas, menus, scrollbars, textfields, etc...

- ✓ java.applet
 - Pacote que habilita a criação de applets através da classe applet. Também prove recursos de áudio.
 - Principais métodos da classe Applet : destroy, getParameter, init, play, resize, showStatus, start, stop

75



Java

Classe Array

- ✓ Representa um conjunto de elementos de um determinado tipo
- ✓ Sua criação é parecida com a criação de outros objetos
- ✓ `int i[] = new int[4];` // criação de um array de 4 posições
- ✓ `i[0] = 55;`
- ✓ `int i[] = { 10,25,40,64};` // outra maneira de se criar um array de 4 posições
- ✓ `int y = i[3];` // **atribuição de que elemento para y ???**
- ✓ `int y = i[4];` // **atribuição de que elemento para y ???**

76



Java

Classe Array (cont.)

✓ `int k[] [] = new int[3][2];`

k:

14	12
20	01
10	34

✓ *Como obter o elemento 34 do array ?*

✓ Propriedade length

- `i.length` resultado : 4
- `k.length` resultado : 3
- `k[2].length` resultado : 2

77



Java

Classe String

✓ Classe que Armazena caracteres , podendo ser manipulada através de métodos

```
char letra;  
String nome = "Joao";  
String nome2 = "Mario";  
int idade = 34;
```

✓ Relação de alguns métodos úteis :

- **charAt(int i)** - retorna caracter que estiver no dado índice i

```
letra = nome.charAt(2);
```

```
System.out.println("A letra na posicao 2 da palavra " + nome +  
"e' " + letra);
```

- **concat(String s)** - concatena com outro String chamado s

```
System.out.println(nome + " + Carlos = " + nome.concat("Carlos"));
```

```
nome = nome.concat(" Carlos");
```

78



Java

Classe String (cont.)

- **equals(Object o)** - retorna true se a String for igual a o
if (nome.equals(nome2))
 System.out.println(nome + " e igual a " + nome2);
else
 System.out.println(nome + " e diferente de " + nome2);
- **valueOf(elemento)** - retorna um objeto String com o valor equivalente ao do dado elemento. É uma maneira para se usar outros métodos em cima da tal palavra.
 System.out.println("A idade de " + nome + " e' " +
 String.valueOf(idade) + " anos");
- **indexOf(char c)** - retorna índice da primeira ocorrência do caractere c. Retornará -1 se não achar.
 System.out.println("A primeira letra o em " + nome + " esta na
 posicao "+ String.valueOf(nome.indexOf("o")));

79



Java

Classe String (cont.)

- **length()** - retorna tamanho da String. Neste caso é um método, ao contrário da classe Array.
 System.out.println(nome + " tem" + String.valueOf(nome.length()) +
 " caracteres");
- **replace(char v, char n)** - retorna String com substituição de caracter velho por um novo
 System.out.println("A Troca da letra i por c de " + nome2 + " e' " +
 nome2.replace('i','c'));
 nome2 = nome2.replace('i','c');
- **substring(int sI, int sF)** - retorna String com os caracteres entre sI(inclusive) e sF(exclusive)
 System.out.println("Substring(0,2) de " + nome + " e' " +
 nome.substring(0,2));
 nome = nome.substring(0,2);

80



Java

Classe String (cont.)

- **toLowerCase()** - retorna String convertida para minúscula
System.out.println(nome + " em minusculo e' " + nome.toLowerCase());
nome = nome.toLowerCase();
- **toUpperCase()** - retorna String convertida para maiúscula
System.out.println(nome + " em maiusculo e' " + nome.toUpperCase());
nome = nome.toUpperCase();
- **trim()** - retorna String sem espaços
nome = nome.concat("ao Carlos ");
System.out.println('*' + nome + "*" sem brancos desnecessarios e' "*" +
nome.trim() + '*');
nome = nome.trim();

✓ **Crie uma classe, Digite estes comandos e Teste-os**

81



Java

Programa Java || Definição de Classe

✓ Ex.

```
import java.lang.*;    // não é necessário dar este import
public class empregado extends Object
{
String nomefunc;
String rgeral;
String matricula;
int anomatricula;
/* método construtor */
public empregado(String nfunc, String rg, String matri, int anomat)
{
    this.nomefunc = nfunc;
    this.rgeral = rg;
    this.matricula = matri;
    this.anomatricula = anomat;
}
```

82



Java

Programa Java || Definição Classe (cont.)

```
✓ ...
/* método que retorna valor inteiro */
public int mtTempoServico ()
{
    return (1997 - this.anomatrícula);
}

/* Se método não retornar valor, usa-se a palavra void
Ex. public void <nome método> ()
*/

}
```

83



Java

Programa Java || Usando a Classe Definida

```
✓ Ex.
import java.applet.*;
import java.awt.*;
import empregado;

public class appfunc extends Applet
{
    Panel p1;
    Label l1;
    empregado carlos;
}
```

84



Java

Programa Java || Usando a Classe Definida (cont.)

✓ ...

```
public appfunc ()
{
    carlos = new empregado("Carlos Silva", "15.499.789-
        X", "17.4859", 1980);
    setLayout(new BorderLayout());
    p1 = new Panel ();
    add("Center", p1);
    l1 = new Label(carlos.nomefunc + " Matr.:" +
        carlos.matricula + " RG.:" + carlos.rgeral);
    p1.add(l1);
}
}
```

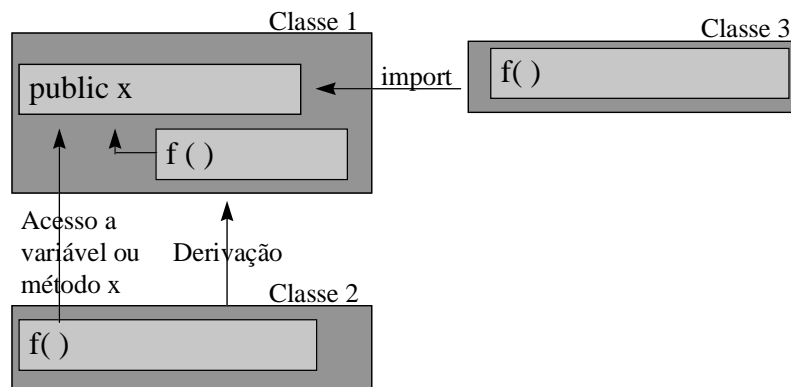
✓ *Adicione ao Exemplo a chamada do Método que Devolve o Tempo de Serviço de Carlos* ⁸⁵



Java

Controle de Acesso || public

✓ Acesso permitido para qualquer método de qualquer classe



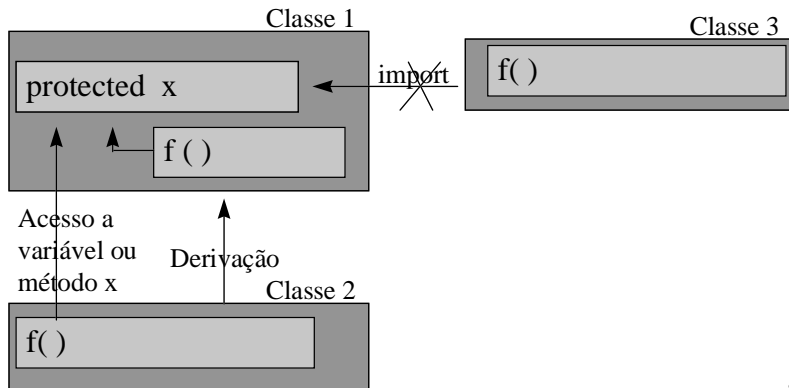
Ps) Pode-se determinar para quais métodos um atributo ou método é visível ⁸⁶



Java

Controle de Acesso || protected

- ✓ Acesso para métodos compatíveis (da mesma classe ou de derivadas)



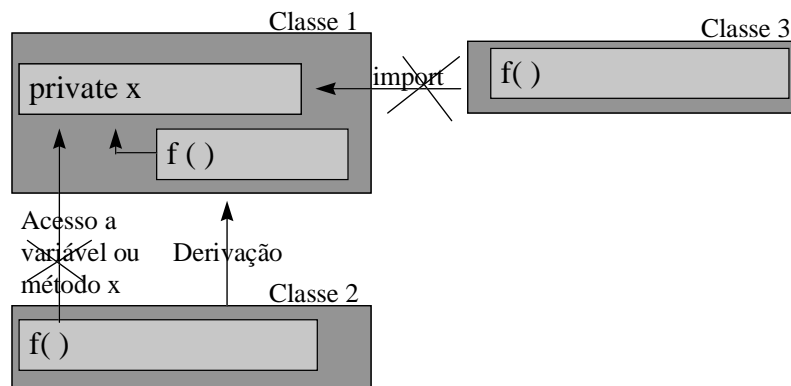
87



Java

Controle de Acesso || private

- ✓ Acesso restrito a métodos da própria classe



88



Java

Obstruindo Mudanças || final

- ✓ **final** : impede modificações
- ✓ Atributo final: Contante
 - **final** int TamVetor = 50; // constante
- ✓ Método Final : Não pode ser redefinido por classes derivadas
 - class Conta {
 final void Retirada (float Quanto)
 { if (Saldo >= Quanto) ... }
};
- ✓ Classe final : Não pode ser herdada
 - **final** class Frame { . . . };
class MessageBox extends Frame // Erro !!!!

89



Java

Tipos de Dados Primitivos || booleano

- ✓ Estes tipos não são objetos
- ✓ Tipo **booleano** pode assumir os valores true e false

```
boolean TemSogra = true;  
if not (TemSogra)  
    System.out.println(" Pessoa Feliz !!");  
else  
    System.out.println(" Pessoa mais Feliz ainda !!");
```

90



Java

Tipos de Dados Primitivos || char

- ✓ Tipo **char** (**caracter**)
 - ✓ Ocupa 2 bytes
 - ✓ Padrão Unicode
 - ✓ public class ExCaracter extends Object
- ```
{
 public static void main (String[] args) {
 char exemploLetra = 'é';
 System.out.println(exemploLetra);
 }
}
```

91



# Java

## Definição da Classe messagebox

```
import java.awt.*;
public class messagebox extends Frame {
 Panel p1,p2;
 Label l1;
 Button b1;

 public messagebox(String mens) {
 setLayout(new BorderLayout());

 this.setTitle("MENSAGEM");
 this.reshape(100,100,300,150);
 }
}
```

92



# Java

## Definição da Classe messagebox (cont.)

```
p1 = new Panel();
add("North",p1);
l1 = new Label(mens);
p1.add(l1);
```

```
p2 = new Panel();
add("South",p2);
b1 = new Button("OK");
p2.add(b1);
```

```
public boolean action (Event evt, Object arg) {
 if ("OK".equals(arg)) OU if (evt.target == b1)
 hide(); hide();
 return true; return true;
 } }
```

93



# Java

## Exercício

### ATENÇÃO : LEIA ATENTAMENTE ESTE EXERCÍCIO

- ✓ Digite a classe messagebox e a “compile”
- ✓ Defina uma classe que se utilize da classe messagebox
- ✓ Esta nova classe deverá importar as classes applet e awt
- ✓ A nova classe deverá conter um panel ao norte da applet e nele deverá existir um botão
- ✓ A função deste botão será a de exibir o messagebox com uma mensagem a livre escolha do programador.
- ✓ O comando para se ativar o messagebox é :

m1.show ( ) , sendo que m1 é do tipo messagebox

94



# Java

## Tipos de Referência

- ✓ São representados por objetos e arrays
- ✓ Guardam o endereço do objeto ou Array e não o seu valor

```
✓ public class Veículo extends Object {
 public static void main (String [] args) {
 String carro = new String ("carro");
 String carroça = new String ("carroça");
 System.out.println(carro);
 System.out.println(carroça);
 carro = carroça;
 System.out.println(carro);
 System.out.println(carroça);
 }
}
```

95



# Java

## Tipos de Referência

```
✓ public class Soma extends Object {
 int i , j;
 public static void main(String[] args) {
 j = i = 1;
 System.out.println (i+" + "+j);
 i = 2;
 System.out.println(i+" + "+j);
 }
}
```

- ✓ **Qual a diferença entre os dois tipos de referências ???** <sub>96</sub>



# Java

## Igualdade e Cópia entre Objetos

- ✓ O operador == verifica se objetos tem a mesma referência
- ✓ Existem em várias classes o método **equals**
- ✓ Este método realiza teste de igualdade de conteúdos.

```
✓ public class TestaIgualdade extends Object {
 public static void main (String [] args) {
 if (args[0].equals ("branco"))
 System.out.println("Cor e' branca !!!")
 else
 System.out.println("Cor não é branca !!!")
 }
}
```

- ✓ **Digite este exemplo, “Compile-o” e Teste-o**

97



# Java

## Igualdade e Cópia entre Objetos

- ✓ Todas as classes possuem um método clone();
- ✓ Retorna uma cópia do objeto em referências diferentes;

Ex.

```
Figura circulo1 = new Figura();
Figura circulo2 = circulo1.clone();
```

98



# Java

## Tratamento de Exceções

- ✓ Erros mais comuns :
  - Problema no acesso a arquivo;
  - Entrada Inválida;
  - Divisão de inteiro por zero;
  - Divisão de não-inteiro por zero;
  - Dado inadequado para conversão;
  - Falta de Memória.
- ✓ O Java cria objeto “**e**” da classe **Exception**
- ✓ É guardado neste objeto a mensagem de erro

99



# Java

## Tratamento de Exceções

```
✓ public boolean action (Event evt, Object arg) {
 if (“soma”.equals(arg)) {
 try {
 x = Integer.parseInt(t1.getText());
 y = Integer.parseInt(t2.getText());
 z = x + y;
 l1.setText(“RESULTADO: “ + z + “ “);
 }
 catch (Exception e) {
 m1 = new messagebox(“Erro: “ + e.toString());
 m1.show();
 }
 }
}
```

100



# Java

## Tratando Eventos do Windows

- ✓ Verificar documentação da classe Event
- ✓ Eventos são capturados através do método `handleEvent`

```
✓ public boolean handleEvent(Event evt)
{
 if (evt.id == (Event.WINDOW_DESTROY))
 System.exit(0);
 return true;
}
```

101



# Java

## Eventos do Mouse e Teclado

- ✓ Estes eventos **são tratados dentro de métodos pré-definidos** tais como :  
mouseUp, mouseDown, mouseDrag, mouseMove, mouseEnter, mouseExit,  
KeyDown, GotFocus, LostFocus, ...

```
✓ Ex.
import java.applet.*;
import java.awt.*;
public class Eventos extends Applet {
 public boolean mouseUp(Event e, int x, int y)
 { showStatus("mouseUp posição: " + x + " e " + y + "");
 return true;
 }
 public boolean mouseDown(Event e, int x, int y)
 { showStatus("mouse Down posição: " + x + " e " + y + "");
 return true;
 }
}
```

102



# Java

## Eventos do Mouse e Teclado (cont.)

```
public boolean mouseDrag(Event e, int x, int y)
{ showStatus("mouse Drag posição: " + x + " e " + y + "");
 return true;
}
public boolean mouseMove(Event e, int x, int y)
{ showStatus("mouse Move posição: " + x + " e " + y + "");
 return true;
}
public boolean mouseExit(Event e, int x, int y)
{ showStatus("mouseExit posição: " + x + " e " + y + "");
 return true;
}
```

103



# Java

## Eventos do Mouse e Teclado (cont.)

```
public boolean Keydown(Event e, int x)
{ showStatus("Keydown código: " + x);
 return true;
}
}
```

✓ **Digitar estes eventos e ver como se comportam**

104



# Java

## Tratando Cores

- ✓ Cores **pré-definidas** :
  - black, blue, cyan, darkGray, gray, green, lightGray, magenta, orange, pink, red, white, yellow.
- ✓ Pode-se criar cores utilizando-se de cores básicas
- ✓ O Padrão utilizado é o **RGB**(RED, GREEN, BLUE)
  - Cor branca = 255,255,255 ou FFFFFFFF
  - Cor preta = 0,0,0 ou 000000
- ✓ Ex.
  - Panel p1, p2;
  - ...
  - p1.setBackground(Color.magenta);
  - p1.setForeground(Color.black);
  - Color c1 = new Color(230,240,250);
  - p1.setBackground(c1);

105



# Java

## Exercício

- ✓ Dado que :
  - Integer.parseInt(<string>) ---> retorna um valor inteiro do parâmetro passado
  - Ex: i = Integer.parseInt("464") ---> i = 464
- ✓ Escreva os comandos em Java para :
  - Obter o conteúdo de um TextField t2;
  - somar o valor 10 ;
  - Colocar o resultado no próprio TextField t2;
- ✓ Crie variáveis auxiliares se necessário

106



# Java

## Exercício

- ✓ Defina uma classe chamada **paint**
- ✓ Esta nova classe deverá importar as classes applet e awt
- ✓ A nova classe deverá ter uma cor de frente e de fundo
- ✓ Quando se pressionar o botão do mouse e começar a operação de arrasto, deverá ir sendo desenhada uma linha.
- ✓ O comando para se ativar modo gráfico é :  
Graphics g;  
g = getGraphics();
- ✓ O método para se desenhar uma linha é :  
g.drawLine(x1,y1,x2,y2);
- ✓ Use o comando **getGraphics()** e **os métodos de setar cores dentro do método init()**, o qual funciona como um método construtor

107



# Java

## Exercício || Saideira

- ✓ Existe uma classe previamente digitada chamada **apaluno**
- ✓ Digitar o código referente ao cálculo do Salário Família, bem como a consistência do campo Idade do Funcionário, que serão ativadas ao se clicar o botão **envia**
- ✓ **Salário Família = número de dependentes \* padrao\_familia;**
- ✓ **Se a idade do funcionário for maior que 65, enviar mensagem através do messagebox alertando-o para sua aposentadoria;**
- ✓ Para se ler o conteúdo de um TextField usamos o método `getText()`.  
Exemplo : `t1.getText();`
- ✓ Para se escrever em um TextField usamos o método `setText (< String a ser escrita >)`. Exemplo : `t1.setText("texto");`
- ✓ Lembre-se, o método para se transformar uma String em inteiro(int) é o `parseInt`. Exemplo : `Integer.parseInt(<string>)`.

108



# Java

## Tendências

- ✓ Java em plataforma não tradicional
  - Javaships (silício)
    - ◆ Ex. barbeador elétrico, telefones celulares, ...
- ✓ Onde serão usadas a plataforma Java ?
  - Atuais plataformas (Sistemas Operacionais)
  - Eletro-Eletrônicos
  - Pacotes de Software (Ex. Corel Office for Java)
  - Intranets
- ✓ Sistema Operacional Java
  - Tamanho Compacto
- ✓ Java Security
  - Criptografia
  - Assinatura
  - Autenticação
- ✓ Java Média
  - Vídeo Conferência

109



# Java

## Tendências (cont.)

- ✓ Java Commerce
  - Carteira Eletrônica
  - Dinheiro Eletrônico
  - Cartão de Crédito
- ✓ JDBC
  - Driver instalado no servidor
  - Comunica-se com qualquer banco de dados
  - Independe de plataforma
- ✓ Inexistência de Driver
  - Embutido no S.O. Java

110



# Java

## Sites Interessantes

- ✓ [www.bulletprooj.com/jagg](http://www.bulletprooj.com/jagg)
  - JDBC
- ✓ [java.sun.com:80/products/jdk/1.0.2/api](http://java.sun.com:80/products/jdk/1.0.2/api)
  - Download de releases do Java para várias plataformas com documentação de bibliotecas de classes
- ✓ [www.di.ufpe.br/~java](http://www.di.ufpe.br/~java)
  - Dicas, Introdução em Tópicos, FAQ
- ✓ [www.j-g.com/java](http://www.j-g.com/java)
  - Exemplos Java
- ✓ [java.sun.com/applets/applet.html](http://java.sun.com/applets/applet.html)
  - Exemplos Java
- ✓ [www-a.gamelan.com / pages/Gamelan.programming.ui.html](http://www-a.gamelan.com / pages/Gamelan.programming.ui.html)
  - Libraries, graficos, Base de Dados, Exemplos