FUNDAÇÃO UNIVERSIDADE FEDERAL DE RONDÔNIA - CAMPUS DE JI-PARANÁ
DEPARTAMENTO DE FÍSICA DE JI-PARANÁ – DEFIJI

X SEMANA DA FISICA 2016/1

Prof. Antonio F Cardozo

UNIR

Programação Básica em Java

Minicurso 1

Programação Básica em Java

Público

•Professores e alunos do curso de Física da UNIR campus Ji-Paraná.

Pré-requisitos

•Estar ciente sobre o que é um programa de computador e o que é uma linguagem de programação de computadores?

Ferramentas necessárias:

1. Computador com Sistema Windows
2. Softwares: Java JDK (JAVA Development Kit)
3. Microsoft Bloco de Notas ou qualquer outro editor de texto

Programação Básica em Java

Conteúdo do Minicurso

1) Tecnologia java
2) Configurando o caminho para o Windows Programa Java
3) Compilar / executar programas Java
4) Estruturas Fundamentais de Programação em Java
5) Introdução à Linguagem Java
6) Fundamentos da Orientação a Objetos
    a) Interfaces em Java – A biblioteca AWT e Swing
    b) Applets

1 TECNOLOGIA JAVA

O que é Java ?

- Uma linguagem de programação
- Um ambiente de desenvolvimento
- Um ambiente de aplicação

Java é uma linguagem de programação desenvolvida pela SUN com o objetivo de manter o poder computacional de C++, agregando características de segurança, e portabilidade que permite criar programas multplantaforma

1 TECNOLOGIA JAVA

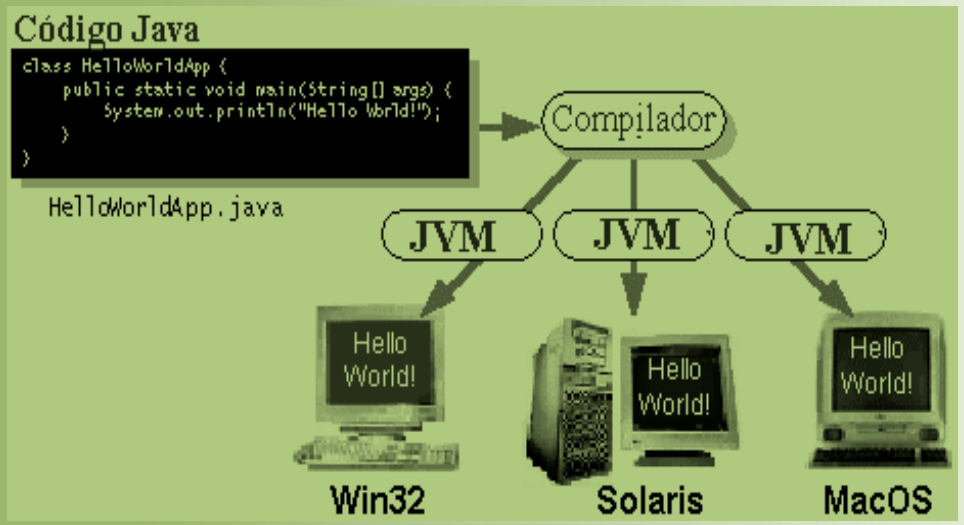
A plataforma Java possui dois componentes:

1. A máquina virtual Java (JVM);
2. A *Application Programming Interface (API).*

De acordo com a especificação da SUN, a máquina virtual do Java pode ser vista como:

Uma máquina imaginária que é implementada via software ou hardware. Um código a ser executado por essa máquina deve ser gravado em um arquivo com extensão *.class. e possuir um código compatível com as instruções Java.*

Para um programa Java ser executado, ele precisa passar pelo processo ilustrado na figura abaixo:



1 TECNOLOGIA JAVA

Máquina virtual

O código é compilado, gerando um conjunto de instruções chamado de *byte-code. Esse byte-code é* aplicado à **Máquina Virtual Java (JVM) que se encarrega de interpretar os comandos para o sistema** operacional onde o programa está rodando. Ou seja, a máquina virtual **traduz as instruções do código** Java para instruções válidas no sistema operacional em que está rodando.

Se essa portabilidade fosse requerida em C, o código deveria ser compilado várias vezes – uma para cada sistema operacional desejado. No caso do Java, o código é compilado apenas uma vez, gerando o *byte-code. Esse byte-code* poderá então ser interpretado por qualquer máquina virtual Java, rodando em Linux, Windows, Palm OS, Solaris ou qualquer outro sistema operacional que possua uma máquina virtual Java implementada.

IMPORTANTE: a JVM não permite que um programa Java acesse recursos de hardware diretamente, protegendo o computador de operações perigosas, como acesso à regiões protegidas da memória ou formatação física do disco rígido.

Um programa Java só é executado caso o seu byte-code passe pela verificação de segurança da JVM, que consiste em dizer que:

1. O programa foi escrito utilizando-se a sintaxe e semântica da linguagem Java
2. Não o existem violações de áreas restritas de memória no código
3. O código não gera *Stack Overflow*

1 TECNOLOGIA JAVA

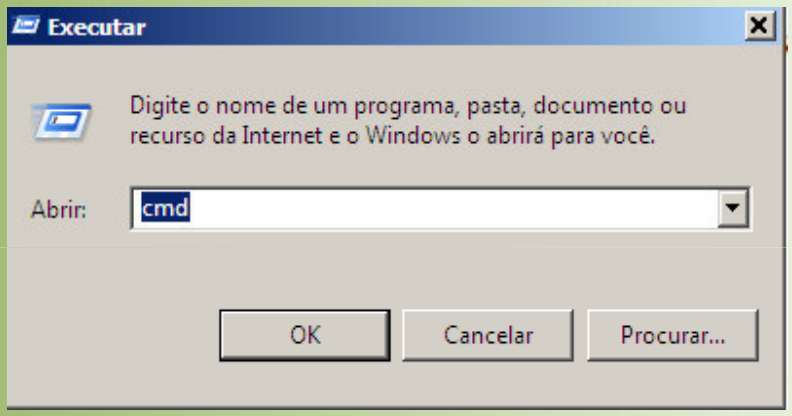
O ambiente de desenvolvimento

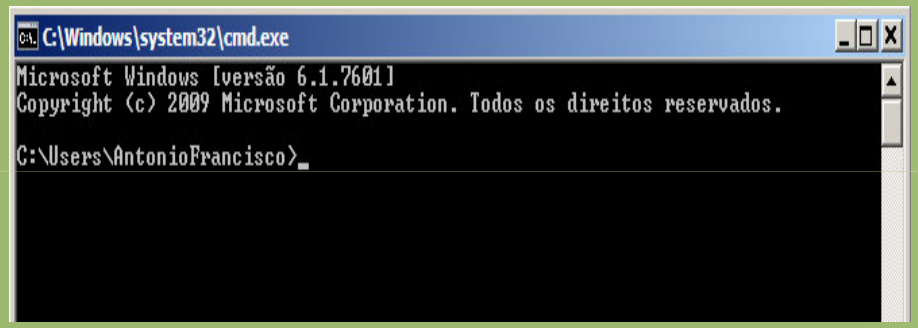
Para que se possa compreender o conteúdo é necessário que já tenha instalado o Java em sua máquina.

Para verificar se o Java foi corretamente instalado em sua máquina, faça o seguinte: clique iniciar/executar e digite o comando **cmd**

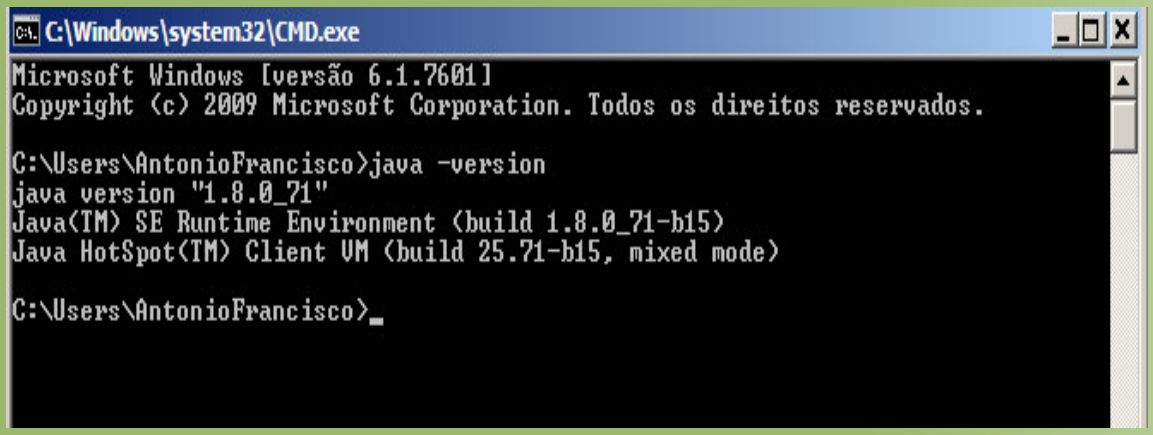
PRÁTICA1

executa o comondo Java –version





Digite o comando **java -version**



Caso não esteja funcionando, revise o processo de instalação do jdk.

jdk-8u25-windows-x64

2 Configurando o caminho para o Windows

Devemos criar uma pasta na unidade C: e salve o arquivo nessa pasta. JAVA Development Kit é um programa completo para desenvolvimentos em linguagem JAVA.

1-Click em Menu iniciar

2-Painel de controle

3-Click em Sistema

4-Irá aparecer a janela "Propriedades do Sistema"

5-Click na aba Avançado

6-Click no botão Variáveis de Ambiente

tem dois(2) tipos de Variáveis: a **"Variáveis de usuário"** e "Variáveis do sistema".

Devemos usar a **Variáveis de usuário.**

2 Configurando o caminho para o Windows Programa Java

A partir desse ponto devemos configurar as seguintes variáveis: CLASSPATH, JAVA_HOME e PATH.

7-Click no botão Novo
No **nome da variável escreva CLASSPATH** . **No valor da variável escreva .;JAVA_HOME** em seguida click OK.

8-Click no botão Novo
A próxima variável é **JAVA_HOME** Esta variável de ambiente aponta para o diretório onde o jdk foi instalado, no meu caso como não escolhi um lugar, ele instalou no local padrão.

Escreva **no nome da variável JAVA_HOME**. **o valor C:\Arquivos de programas\Java\jdk1.6.0_18** em seguida OK.
Note que o Java instalado na minha máquina é jdk1.6.0_18 ,uma dica pra não ter erro é abrir o Windows Explorer ir ao diretório onde o Java está intalado,e copiar da barra de endereço e colar no valor da variável.

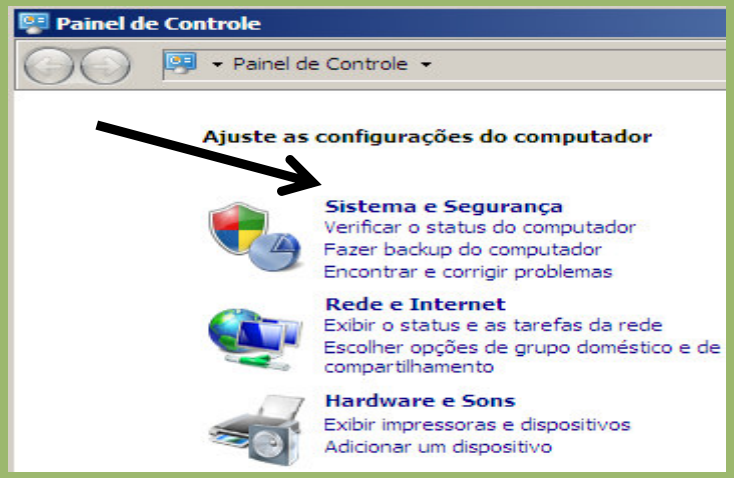
9-Por último Click no botão Novo

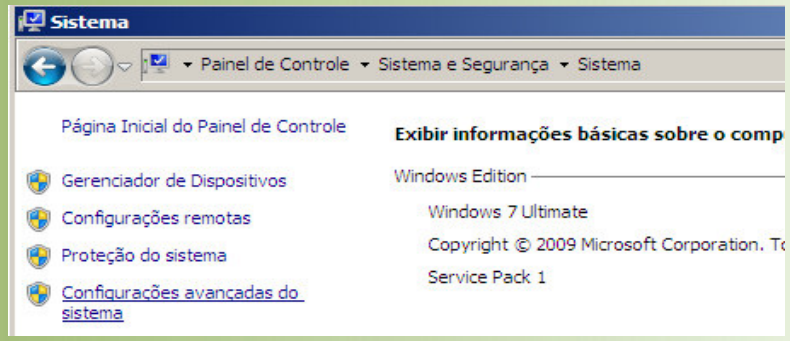
No **nome da Variável escreva PATH** Esta variável de ambiente é responsável por definir um caminho de pesquisa para arquivos executáveis. Neste momento esta variável de ambiente á a mais importante para nós, pois se esta variável não estiver configurada, o processo de compilação dos programas Java somente poderá ser executada dentro da pasta "bin" do diretório de instalação do JDK, pois dentro desta pasta que se encontra o compilador "javac.exe" e outras ferramentas importantes.

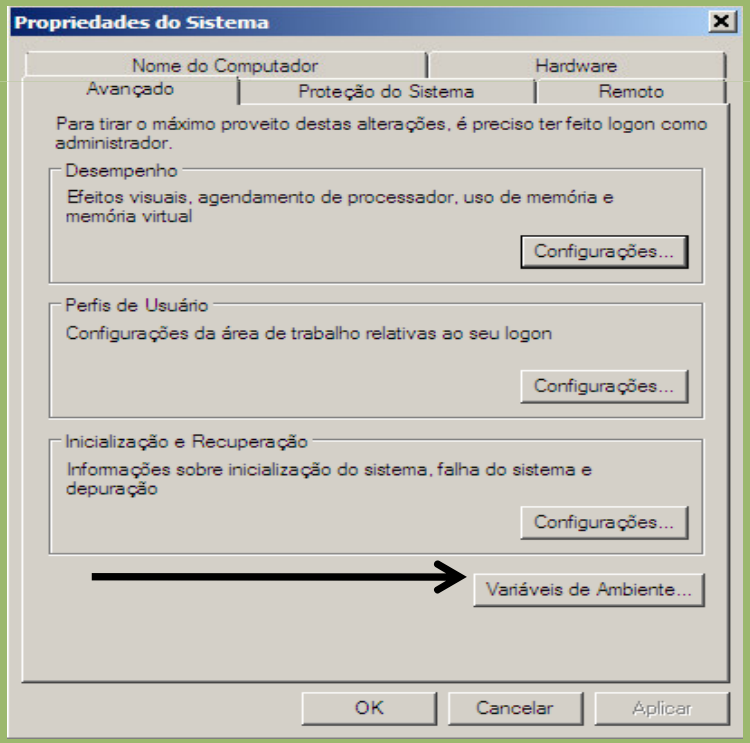
No **valor escreva C:\Arquivos de programas\Java\jdk1.6.0_18\bin** agora é só dar OK pra variável ser criada e dar OK na janela Variáveis de Ambiente e OK na Propriedades do Sitema.

Agora abra o Prompt (digite CMD java -version
Irá aparecer algo assim: java version "1.6.0_18" Java(TM) SE Runtime Environment (build 1.6.0_12-b04) Java HotSpot(TM) Client VM (build 11.2-b01, mixed mode, sharing) E pra ver se

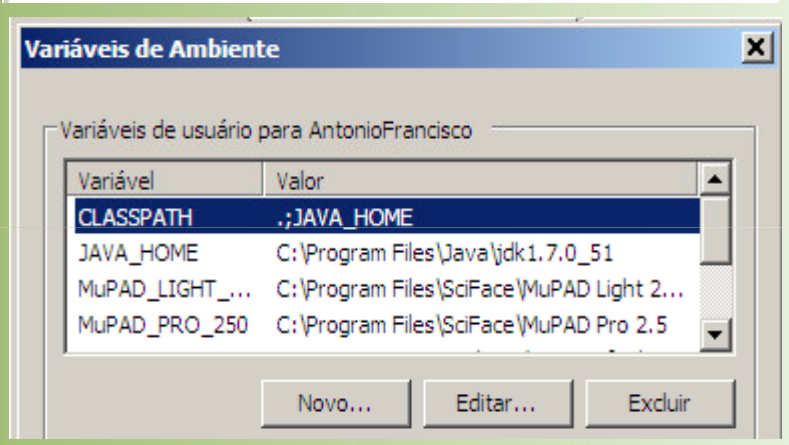
2. Configurando o caminho para o Windows Programa Java

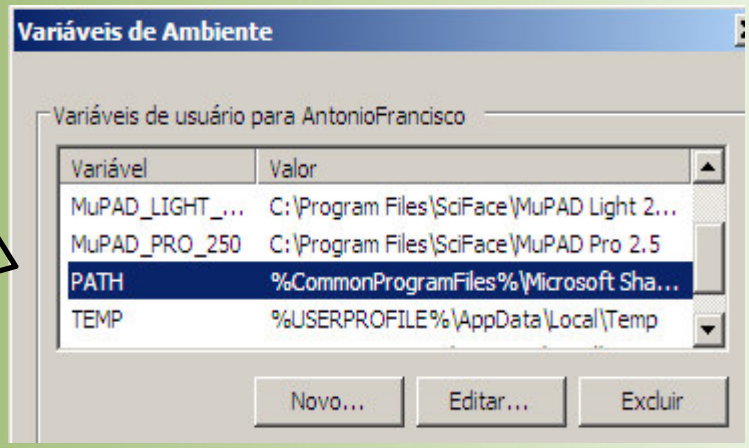






%CommonProgramFiles%\Microsoft Shared\Windows Live;C:\Program Files\Java\jdk1.7.0_51\bin





3 Compilar / executar programas Java

javac é o compilador primário da linguagem Java, incluído no Java Development Kit (JDK) da Oracle Corporation. Apesar de existirem outros compiladores, o criado pela Sun Microsystems é o mais usado.

Para compilar um programa digite	javac nome.java
Para executar um programa digite	Appletviewer nome.html

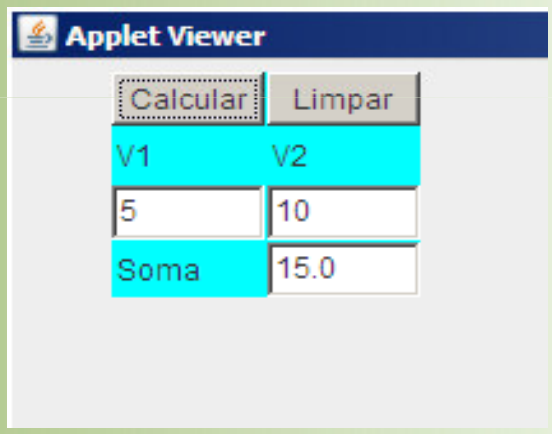
3 Compilar / executar programas Java

PRÁTICA1 ◘

1) Compilar e testar o código fonte de um applet que faz uma soma com dois valores, usando dois botões para calcular e limpar o valor da soma.

PASSOS

1) escrever o código e salvar com a extenção programa1.java
2) compilar o código com o comando javac programa1.java
3) criar um arquivo programa1.html na mesma pasta do applet
4) executar o arquivo programa1.html

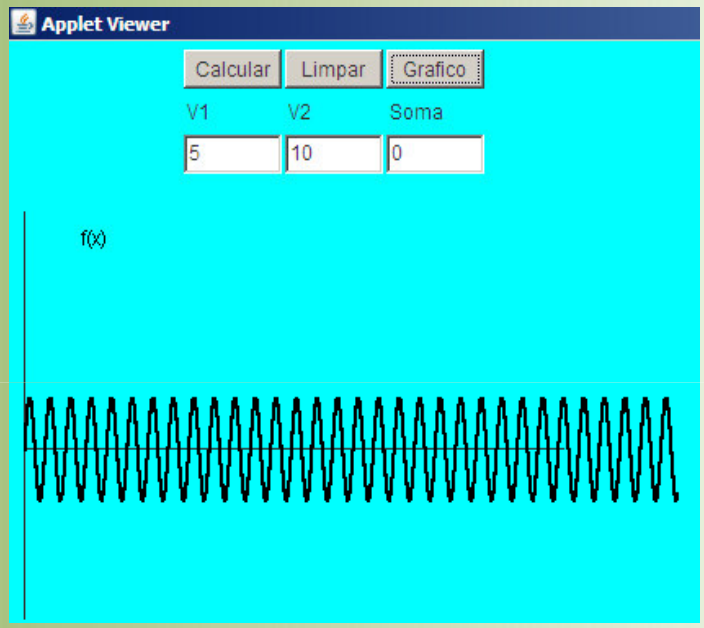


3 Compilar / executar programas Java

PRÁTICA1 ◘

2) Compilar e testar o código fonte de um applet que que faça:

1. Uma soma com dois valores
2. Desenhe as retas x e y de um plano cartesiano
3. Plote o gráfico de uma função f(x)
4. Limpe a área do gráfico



Os objetos awt são: botão1, botão2 e botão3
Caxa1, caixa2 e caixa3

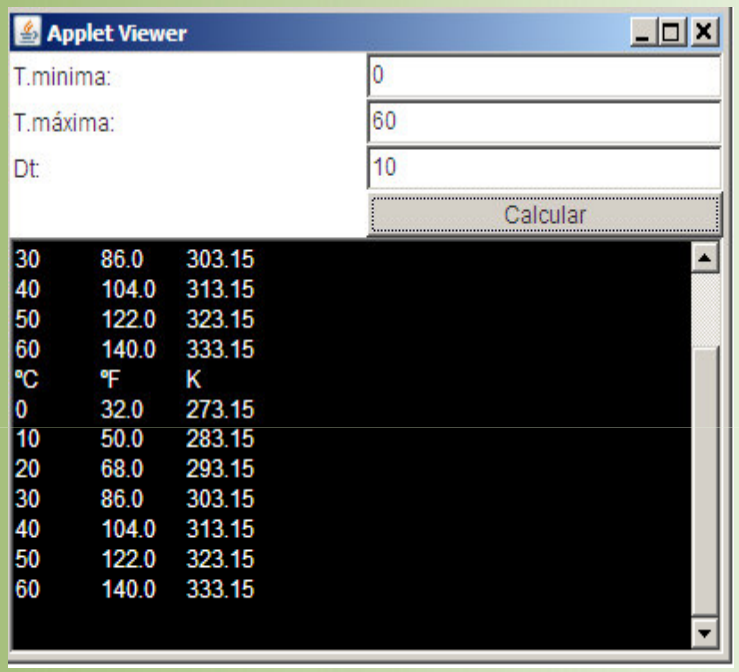
PASSOS

1) Escrever o código e salvar com a extenção programa2.java
2) Compilar o código com o comando javac programa2.java
3) Criar um arquivo programa2.html na mesma pasta do applet
4) Executar o applet

3 Compilar / executar programas Java

PRÁTICA1 ◘

3) Compilar e testar o código fonte de um applet para transformação de ºC para ºF e mostrar os resultados em uma TextArea



PASSOS

1) Escrever o código e salvar com a extensão programa3.java
2) Compilar o código com o comando javac programa3.java
3) Criar um arquivo programa3.html na mesma pasta para chamar o applet
4) Executar arquivo programa3.html

3 Compilar / executar programas Java

O comando Javac -help

```
C:\Minicurso2016\OsApplets>-help
O servidor DNS não está autorizado na zona.

C:\Minicurso2016\OsApplets> -help
O servidor DNS não está autorizado na zona.

C:\Minicurso2016\OsApplets>javac -help
Usage: javac <options> <source files>
where possible options include:
  -g                         Generate all debugging info
  -g:none                    Generate no debugging info
  -g:{lines,vars,source}     Generate only some debugging info
  -nowarn                    Generate no warnings
  -verbose                   Output messages about what the compiler is doing
  -deprecation               Output source locations where deprecated APIs are u
sed
  -classpath <path>          Specify where to find user class files and annotati
on processors
  -cp <path>                 Specify where to find user class files and annotati
on processors
  -sourcepath <path>         Specify where to find input source files
  -bootclasspath <path>      Override location of bootstrap class files
  -extdirs <dirs>            Override location of installed extensions
  -endorseddirs <dirs>       Override location of endorsed standards path
  -proc:{none,only}          Control whether annotation processing and/or compil
ation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors
to run; bypasses default discovery process
  -processorpath <path>      Specify where to find annotation processors
  -d <directory>             Specify where to place generated class files
  -s <directory>             Specify where to place generated source files
  -implicit:{none,class}     Specify whether or not to generate class files for
implicitly referenced files
  -encoding <encoding>       Specify character encoding used by source files
  -source <release>          Provide source compatibility with specified release

  -target <release>          Generate class files for specific VM version
  -version                   Version information
  -help                      Print a synopsis of standard options
  -Akey[=value]              Options to pass to annotation processors
  -X                         Print a synopsis of nonstandard options
  -J<flag>                   Pass <flag> directly to the runtime system
  -Werror                    Terminate compilation if warnings occur
  @<filename>                Read options and filenames from file


C:\Minicurso2016\OsApplets>_
```

3 Estruturas Fundamentais de Programação em Java

POO - Programação Orientada a Objetos. Este tipo de programação trata os elementos da linguagem de maneira semelhante aos objetos reais.

Packages - É semelhante ao conceito de biblioteca de funções, sendo que um package (pacote) é um conjunto de classes, que ficam num diretório com o mesmo nome do pacote. O package default é o java.lang.*; que é adicionado a todo arquivo java mesmo que o programador não o referencie. O .* diz ao java para incluir todas as classes do pacote.

Classes - É um conjunto de objetos com características comuns. Uma classe é como um modelo para a criação de objetos, que tem as mesmas características da classe à qual pertence.

Classes Abstratas

Em uma hierarquia é útil padronizar os serviços providos pelas classes. Por exemplo, suponha que você deseje implementar algumas classes que representem polígonos: Retângulo, Quadrado, Elipse e Triângulo. Estes polígonos terão dois métodos básicos: área ( ) e circunferência ( ). Agora, para ser fácil trabalhar com um array de polígonos, seria útil que todos os polígonos possuíssem uma mesma superclasse, Shape. Para isso, nós queremos que a classe Shape contenha todas as estruturas que nossos polígonos tenham em comum (os

Objetos - É um elemento de uma classe. Temos uma classe "gatos" que é formada pelos objetos "gato". Todos os objetos tem em comum o fato de serem gatos (mesma espécie), mas podem ter características diferentes entre si. Objetos tem variáveis e métodos como também classes.

Instância - Uma instância de uma classe é um novo objeto criado dessa classe, com o operador new. Instanciar uma classe é criar um novo objeto do mesmo tipo dessa classe. Uma classe somente poderá ser utilizada após ser instanciada.

3 Estruturas Fundamentais de Programação em Java

Métodos - Os métodos representam os estados e ações dos objetos e classes.

Variáveis - As variáveis e constantes representam as características dos objetos.

Packages - Classes - Objetos - Métodos e variáveis.

Interface - É a declaração de um conjunto de constantes e métodos sem qq implementação, usando a palavra implements. É o conjunto de requisições que um objeto pode atender.

Superclasse - Todas as classes são criadas tendo outra como base. A classe que gerou a outra é chamada de superclasse. E fazemos referência à supercasse de uma classe usando a palavra extends. A classe gerada é chamada de subclasse. Toda classe tem uma subclasse. Quando não declaramos explicitamente a superclasse a super default é a Object.

Construtores - São usados para inicializar objetos. É o método que tem o mesmo nome da sua classe. Não pode ter um tipo de retorno e é chamado pelo operador new.

Polimorfismo - É a capacidade de um método executar a ação adequada dependendo do tipo de objeto.

Threads - São fluxos de execução paralelos, executando cada um tarefas diferentes. Em máquinas com dois ou mais processadores a execução pode ser simultânea e em máquinas com um único a execução será simulada, dividindo o tempo de processamento.

Herança - Quando a subclasse herda as características da superclasse.

Encapsulamento - É o empacotamento de variáveis e métodos, ocultando a implementação do usuário. Representa reutilização, segurança e facilidade de manutenção.

COMO CRIAR CLASSES EM JAVA

Como iniciar uma classe: Para iniciar uma classe, é necessário informar essa ação ao compilador, que interpreta como um acesso à esse objeto. Uma classe é um tipo de objeto. A sintaxe para acessar um objeto é a seguinte: public tipo_de_objeto nome_do_objeto No caso ilustrado ao lado, o tipo de objeto seria uma classe, e por isso temos class, cujo nome é modelo1. Como essa classe deriva de outra classe, no caso a classe Applet, orientamos o compilador dizendo que essa classe extende as propriedades da uma determinada classe mãe: extends nome_da_classe_mãe Para uma classe aceitar um método, muitas vezes é preciso que você implemente um tipo de ação que permite a um programa responder a um ou vários eventos. Para implementar a classe utilizase: implements tipo_de_evento

O que é uma classe?

Na verdade, a classe não é exatamente um objeto, mas sim um modelo ou especificação que define um tipo de objeto. Certo? Mas o que seria um objeto? De uma forma muito simplificada, o objeto seria uma ferramenta de comunicação entre o usuário e o seu programa, que apresenta um agrupamento de dados e procedimentos definidos na classe.

Primeiro exemplo: Vamos supor que você está escrevendo um programa de corrida de carro. Será necessário criar um objeto carro, que terá as características definidas pela classe modelo. Então, de acordo com as instruções do usuário, esse carro poderá acelerar
ou retardar seu movimento. Mas esse "comportamento" do carro só poderá ser anipulado, se ele estiver corretamente definido pela classe modelo (programação para reconhecer quando deve acelerar, quando deve se virar para a direita, etc.).

Um outro exemplo: Vamos supor que você tem uma ficha cadastral online, em que se pede: nome, endereço, bairro, cidade, estado, cep, telefone, email e opções de produtos a serem comprados. Cada identificação de campo a ser preenchido é um label. Cada ampo é uma caixa de texto. E as opções de compra, são do tipo checkbox. O label, a caixa de texto e o checkbox utilizados são os objetos dessa ficha cadastral. Para que no label apareça o texto "Nome", por exemplo, é necessário que isso seja definido em uma classe. E para que o texto digitado na caixa de texto seja identificado e armazenado em uma determinada variável, é necessário que os procedimentos sejam descritos no corpo da classe. E assim por diante. Temos que preparar uma classe. Nosso programa em Java é uma classe.

Convém observar que os objetos devem ser declarados logo após a inicialização da classe, para que durante a descrição de suas características, eles possam ser reconhecidos como pertencentes à classe em que se está trabalhando. Características do objeto (cor, legenda, etc) sintaxe simples na interface Respostas (seleção do checkbox, armazenar dados do textfield, etc) > evento

Classe derivada de outra classe?

Quando se diz que uma classe deriva de outra, quer se dizer que ela herda as ropriedades e os métodos dessa classe (classemãe). Uma classe é identificada como filha de uma classe mãe através da palavrachave extends.

```
Public class modelo1 extends Applet
{
...
}
```

(quer dizer que a classe modelo1, que você criou com suas próprias variáveis e métodos, herdará todas as variáveis e métodos da classe mãe Applet)

Quando uma classe não é explicitamente derivada de outra classe, ela é implicitamente derivada da classe Object que é a classemãe original de todas as outras classes.

Como definir classes

Para definir uma classe use a palavrachave *class e o nome da classe.*
Exemplo:

```
class Minhaclasse
{
...
}
```

Se esta classe é uma subclasse de outra classe, use *extends para indicar* superclasse.
Exemplo:

```
class Minhaclasse extends SuperClasse
{
...
}
```

Como definir variáveis de instância

As variáveis de instância aparentemente são declaradas e definidas quase exatamente da mesma forma que as variáveis locais, a principal diferença é que a alocação delas é na definição da classe.
Exemplo:

```
class Bike extends Veículo
{
String tipo;
int correia;
int pedal;
}
```

Polimorfismo ou sobrecarga

O polimorfismo é um dos princípios básicos da orientação a objetos, fazendo referência ao poder que os objetos de classes distintas têm de invocar um mesmo método e obter comportamentos distintos. O polimorfismo está diretamente relacionado a métodos. Os métodos em Java podem ser sobrecarregados, ou seja, podemos criar métodos com o mesmo nome, mas com diferentes assinaturas (parâmetros) e diferentes definições. Quando se chama um método em um objeto, o Java casa o nome do método, o número de argumentos e o tipo dos argumentos e escolhe qual a definição do método a executar.

Para criar um método sobrecarregado, é necessário criar diferentes definições de métodos na sua classe, todos com o mesmo nome, mas com diferentes parâmetros (número de argumentos ou tipos). No exemplo a seguir, veremos a definição da classe MyRect, a qual define um retângulo plano. A classe MyRect tem quatro variáveis para instanciar, as quais definem o canto superior esquerdo e o canto inferior direito do retângulo: x1, y1, x2 e y2. public static int x1 = 0,y1 = 0, x2 = 0, y2 = 0;]

Quando uma nova instância da classe MyRect for criada, todos as suas variáveis são inicializadas com 0. Definindo um método MyRect buildRect( ): este método recebe quatro inteiros e faz um "resize" do retângulo de acordo com as novas coordenadas e retorna o objeto retângulo resultante (note que os argumentos possuem o mesmo nome das variáveis instanciáveis, portanto deve se usar o *this para referenciá-las):*

```
MyRect buildRect(int x1, int y1, int x2, int y2)
{
this.x1 = x1; this.y1 = y1;
this.x2 = x2; this.y2 = y2;
return this;
}
```

Querendose definir as dimensões do retângulo de outra forma, por exemplo, podese usar o objeto Point ao invés de coordenadas individuais. Faremos a sobrecarga do método MyRect buildRect ( ), passando agora como parâmetro dois objetos Point:

```
MyRect buildRect(Point topLeft, Point bottomRight)
{
        x1 = topLeft.x; y1 = topLeft.y;
        x2 = bottomRight.x; y2 = bottomRight.y;
        return this;
}
```

Entretanto, que rendose definir um retângulo usando somente o canto superior esquerdo e uma largura e altura do retângulo pode se ainda definir mais um método MyRect buildRect ( ):

```
MyRect buildRect(Point topLeft, int w, int h)
{
x1 = topLeft.x; y1 = topLeft.y;
x2 = (x1 + w); y2 = (y1 + h);
return this;
}
```

O Programa exemplo de polimorfismo:

```
import java.applet.Applet;
import java.applet.*;
import java.awt.*;
import java.awt.Point;
public class MyRect extends Applet
{
TextArea ta;
public static int x1 = 0,y1 = 0, x2 = 0, y2 = 0;
public void init()
{
ta = new TextArea(16,62); add(ta);
MyRect rect = new MyRect( );
ta.append(" A função buildRect() pode ser chamada, dentro de um
programa,\n");
ta.append(" usando 3 formas diferentes nos argumentos da função.
\n");
ta.append(" A cada tipo de chamada responde de forma diferente \n");
ta.append(" pois foi construida para ser sobrecarregada \n");
ta.append(" \n Chamada a buildRect com coordenadas 25,25, 50,50:");
rect.buildRect(25, 25, 50, 50);
ta.append("\n MyRect: <" + x1 + ", " + y1 +", " + x2 + ", " + y2 +
">");
ta.append("\n ")
;
ta.append("\n Chamada a buildRect com os pontos (10,10),
(20,20):");
rect.buildRect(new Point(10,10), new Point(20,20));
ta.append("\n MyRect: <" + x1 + ", " + y1 + ", " + x2 + ", " + y2 +
">");
ta.append("\n ")
;
ta.append("\n Chamada a buildRect com o point (10,10), largura (50),
altura (50):");
rect.buildRect(new Point(10,10), 50, 50);
ta.append("\n MyRect: <" + x1 + ", " + y1 +", " + x2 + ", " + y2 + ">");
ta.append("\n ")
;
}
MyRect buildRect(int x1, int y1, int x2, int y2)
{
this.x1 = x1; this.y1 = y1;
this.x2 = x2; this.y2 = y2;
return this;
}
MyRect buildRect(Point topLeft, Point bottomRight)
{

x1 = topLeft.x; y1 = topLeft.y;
x2 = bottomRight.x; y2 = bottomRight.y;
return this;
}
MyRect buildRect(Point topLeft, int w, int h)
{
x1 = topLeft.x; y1 = topLeft.y;
x2 = (x1 + w); y2 = (y1 + h);
return this;
}
}
```

Com

4 .Introdução à Linguagem Java

A instrução import da linguagem Java tem como objetivo disponibilizar em uma classe, de um determinado pacote, o acesso a demais classes que estejam em pacotes diferentes. Há duas formas de realizar a importação de uma classe usando a instrução import, a forma explícita e a forma implícita. A partir do lançamento do Java 5, passou a ser disponibilizada também a importação de membros estaticos através da instrução import.

A palavra **import** é uma das muitas palavras reservadas da linguagem Java e não poderá ser usada como nome de variável. Para importar uma classe devesse usar a instrução import logo após a instrução package, caso exista, e antes da declaração da classe. A instrução será seguida pelo caminho do pacote, delimitado por pontos, e terminará com o nome de uma classe ou um caractere do tipo asterisco, encerrando a instrução com um ponto e vírgula, como mostra a **Tabela 1**.

Instrução import	Definição
`import java.net.*;`	Importa todas as classes do pacote java.net.
`import java.net.URL;`	Importa apenas a classe URL do pacote java.net.
`import static java.awt.Color.*;`	Importa todos os membros estáticos da classe Color do pacote java.awt (diposnivel a partir do Java 5).
`import static java.awt.color.ColorSpace.CS_GRAY;`	Importa o membro estático CS_GRAY da classe Color do pacote java.awt (diposnivel a partir do Java 5).

4 Introdução à Linguagem Java

Comentários em Java

Os comentários de uma linha começam com // (duas barras) O comentário de várias linhas é iniciado com **/* (barra-asterisco) e finalizado com*/**

Palavras-chave reservadas

As palavras-chave reservadas Java são usadas para identificar os tipos, modificadores e mecanismos de controle de fluxo. Essas palavras, juntamente com os operadores e separadores, formam a definição da linguagem Java. Elas não podem ser usadas como nome de variável, método ou classe. abstract

abstract	boolean	break	byte	byvalue
case	cast	catch	char	class
const	continue	default	do	double
else	extends	false	final	finally
float	for	future	generic	goto
if	implements	import	inner	instanceof
int	interface	long	native	new
null	operator	outer	package	private
protected	public	rest	return	short
static	super	switch	synchronized	this
throw	throws	transient	true	try
var	void	volatile	while	

4 Introdução à Linguagem Java

Caracteres

Caracter	Significado
\n	Nova Linha
\t	Tab
\b	Backspace
\r	Retorno do Carro
\f	"Formfeed" (avança página na impressora)
\\	Barra invertida
\'	Apóstrofe
\"	Aspas
\ddd	Octal
\xdd	Hexadecimal

Operadores Aritméticos

Operador	Significado	Exemplo
+	soma	3 + 4
-	subtração	5 - 7
*	multiplicação	5 * 5
/	divisão	14 / 7
%	módulo	20 % 7

Operadores de comparação

Operador	Significado	Exemplo
==	Igual	x == 3
!=	Diferente (Não igual)	x != 3
<	Menor que	x < 3
>	Maior que	x > 3
<=	Menor ou igual	x <= 3
>=	Maior ou igual	x >= 3

Operadores lógicos

Operador	Significado
&&	Operação lógica E (AND)
||	Operação lógica OU (OR)
!	Negação lógica
&	Comparação bit-a-bit E (AND)
|	Comparação bit-a-bit OU (OR)
^	Comparação bit-a-bit OU-Exclusivo (XOR)
-	Complemento bit-a-bit
x &= y	atribuição AND (x = x & y)
x |= y	atribuição OR (x = x | y)
x ^= y	atribuição XOR (x = x ^ y)

4 Introdução à Linguagem Java

Declaração de uma String em Applet

```
String teta=new String("\u03b8");
String dis=new String("\u2260");
String pet1 =new String("\u00B2");
String pet2 =new String("\u00B9");
String pet3 =new String("\u2260");
```

Uso da String em Applet

```
g.drawString(teta, 780, 380);
```

Método construtor de um número formatado

```
NumberFormat nf1 = NumberFormat.getNumberInstance();
NumberFormat nf2 = NumberFormat.getNumberInstance();
```

Declaração dentro de uma classe

```
nf1.setMaximumFractionDigits(2);
nf2.setMaximumFractionDigits(0);
```

Uso do número

```
ta.append(nf1.format(12.654321)
```

Declarando um Array:

```
double M[];
M = new double[100]
M[1] = 20;
M[2] = 30;
M[3] = X;
```

Quando criamos um objeto array usando o operador *new,* todos os índices são inicializados para você (0 para arrays numéricos, falso para boolean, '\0' para caracteres, e NULL para objetos). Você também pode criar e inicializar um array ao mesmo tempo.

```
String[] chiles = { "jalapeno",  "anaheim",  "serrano" ,  "jumbou",  "thai"};
```

Cada um dos elementos internos deve ser do mesmo tipo e deve ser também do mesmo tipo que a variável que armazena o array. O exemplo acima cria um array de Strings chamado *chiles* que contém 5 elementos.

Acessando os Elementos do Array

Uma vez que você têm um array com valores iniciais, você pode testar e mudar os valores em cada índice de cada array. Os arrays em Java sempre iniciam-se na posição 0 como no C++. Por exemplo:

```
String[]  arr= new String[10];
arr[10]="out";
```

Isto provoca um erro de compilação pois o índice 10 não existe, pois isto está fora das bordas do array.

```
arr[9] = "inside";
```

Declaração e Inicialização de Valores

As variáveis do tipo byte, short, int, long, float, double, char e boolean podem ser declaradas de acordo com uma das formas exibidas abaixo. int a, b, c; Declarando as variáveis a, b e c.

int d = 3, e, f=5;	Declarando d, e, f e inicializando d com 3 e f com 5.
double pi = 3.14159;	Declarando e inicializando pi com o valor 3.14159;
char x = „x";	Declarando e inicializando x com o caractere „x";

Arrays Multidimensionais

Java não suporta arrays multidimensionais. No entanto, você pode declarar e criar um array de arrays e acessá-los como você faria no estilo-C.

```
 double M2[][];
M2 = new double[10][10];
              M2[0][0] = 1;
              M2[0][1] = 2;
```

Condicionais

O condicional contém a palavra chave *if,* seguido por um teste booleano. Um opcional *else* como palavra chave pode ser executado na caso do teste ser falso, Exemplo:

```
if  ( x < y)
      {
       System.out.println(" x e menor do que y");
      }
else
      {
       System.out.println(" y e maior);
      }
```

O switch

Um comum mecanismo para substituição de *ifs* que pode ser usado para um grupo de testes e ações junto a um simples agrupamento, chama-se *switch.*

```
        switch (teste)
{
    case  valorum;
        resultum;
        break;
    case valordois;
        resultdois;
        break;
    case valortres:
        resulttres;
        break;
    default: defaultresult;
}
```

O valor é comparado com cada um dos casos relacionados. Se a combinação não for encontrada, o bloco *default* executado. O *default* é opcional, então caso este não esteja associado ao comando, o bloco do *swicth* sem executar nada.

4 Introdução à Linguagem Java

. *Looping* For

O loop em Java tem esta sintaxe:

```
for(inicialização; teste; incremento)
{
     bloco de comandos;
}
```

```
for ( i=0; i<100; i++)
    {
       strArray[i]="";
    }
```

Loop While

O *while* é usado para repetir um comando, ou um conjunto de comando enquanto a condição é verdadeira.

```
While (condição){
        bloco de comandos;
}
```

A *condição* é uma expressão booleana. Exemplo:

```
int count=0;
while( count < array1.length  && array1[count]!=0)
{
        array2[count]=(float) array1[count++];
}
```

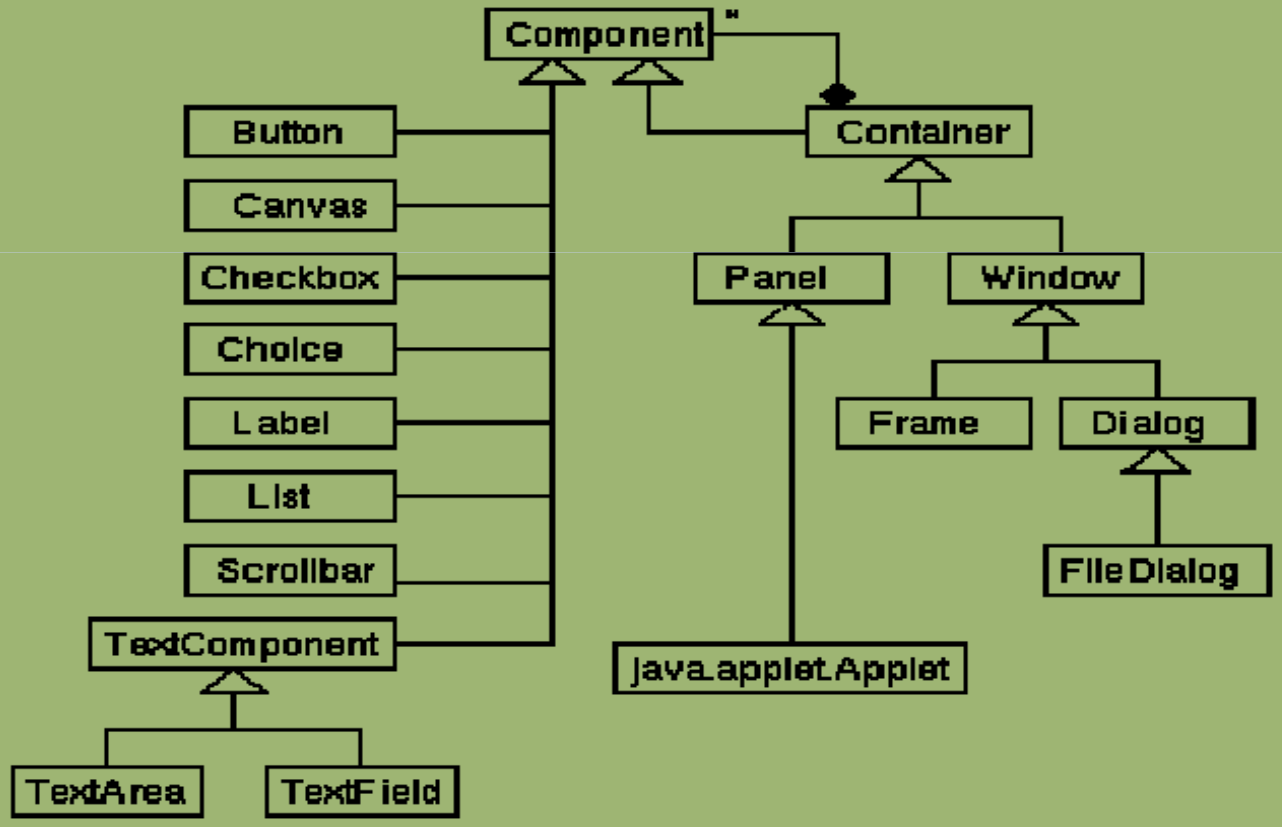
break

O termo break é usado interromper laços (for, while, do-while).

```
if (a == 5) break;
```

Interfaces de usuario com AWT y Swing

Jerarquía de Componentes AWT



Componentes GUI AWT: Clases Window fundamentais

Clase AWT	Descrição
Component	Uma classe abstrata para objetos que podem ser exibidos no console e interagir com o utilizador. A raiz de todas as outras classes AWT
Container	Uma subclasse abstrata da classe Componente. Um componente que pode conter outros componentes AWT
Panel	Estende a classe Container. Um quadro ou uma janela sem o barra de título, a barra de menu nem a fronteira. Superclasse do Applet classe
Window	Estende a classe Container. Um objeto Window é um de nível superior janela sem fronteiras e nenhuma barra de menu. (Padrão BorderLayout)
Frame	Entends a classe Window. Uma janela com um título, barra de menus, fronteira, e redimensionar cantos.

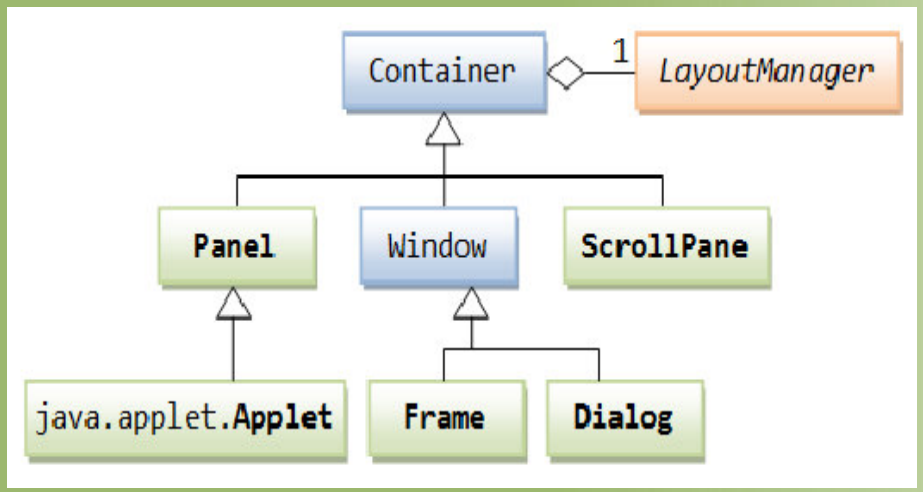
5. Fundamentos da Orientação a Objetos

Componentes GUI AWT: Abstract Window Toolkit foi a interface gráfica original da linguagem.

Componentes que permitem ao usuario interagir com a aplicação GUI. Subclases da clase Component

Label	Label
Button	Button
TextField	Text Field
TextArea	Text Area
Checkbox	Checkbox
Choice	Choice
List	List
Scrollbar	Scrollbar
ScrollPane	Scroll Pane
Panel	Panel
Canvas	Canvas
MenuBar	Menu Bar
PopupMenu	Popup Menu

Hierarquia das AWT Container Classes



Componentes GUI Swing

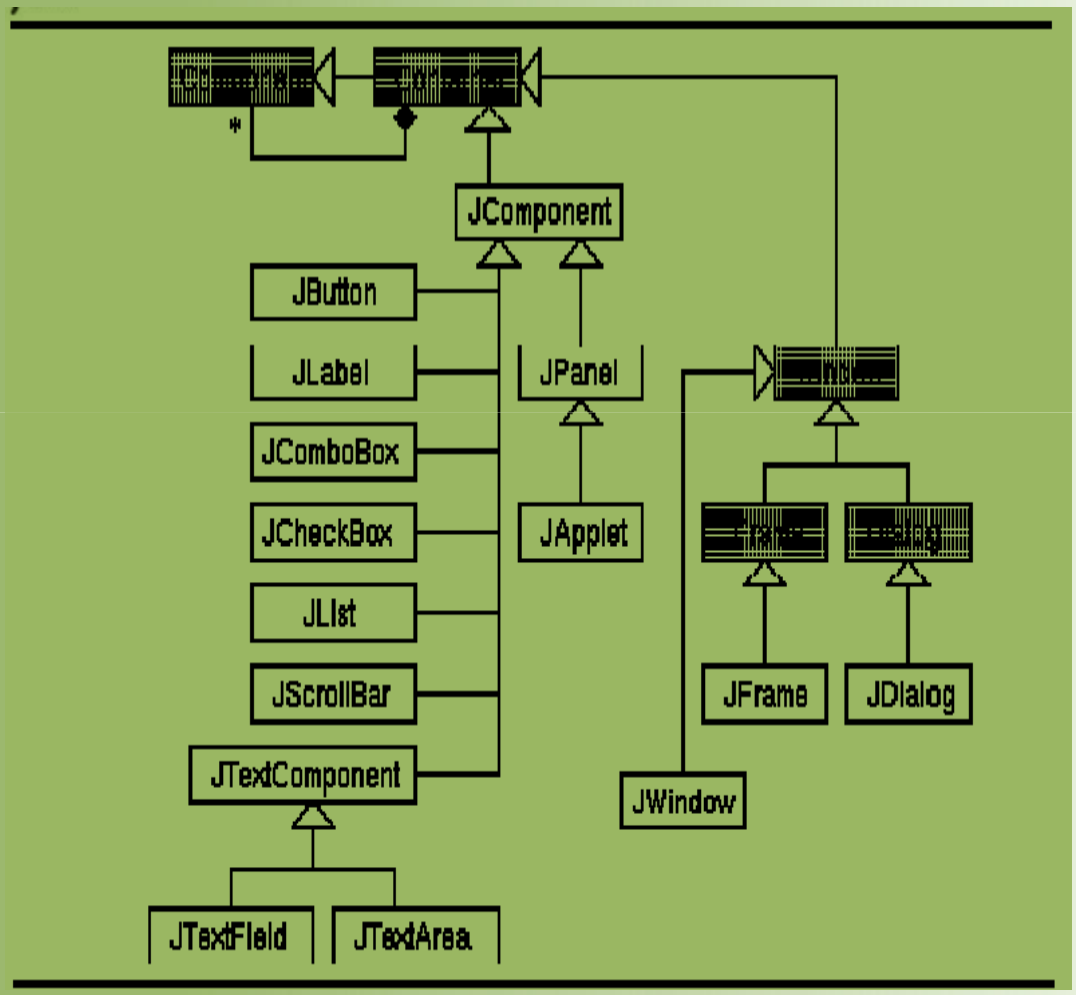
Os Componentes GUI Swing são usados no javax.swing e consiste várias centenas e várias classes subpackages

• escrito inteiramente em Java, e, portanto, têm a mesma aparência se em execução em diferentes plataformas

•Nomes semelhantes aos componentes AWT: seguido começando com a letra J. Exemplo: Jbutton

•Fornece componentes mais Interessantes: Opções do painel de seleção de cores, etc.

Hierarquia de Componentes Swing



Classe Swing	Descrição
JButton	Uma implementação de um botão "push".
JComboBox	Um componente que combina um botão ou campo editável e uma lista drop-down.
JComponent	A classe base para todos os componentes Swing, excepto os recipientes de nível superior.
JDialog	A classe principal para a criação de uma janela de diálogo.
JFileChooser	JFileChooser fornece um mecanismo simples para o usuário escolher um arquivo.
Jframe	Uma versão estendida do java.awt.Frame que adiciona suporte para o JFC / Swing arquitetura de componentes.
JLabel	A área de exibição para uma cadeia curta de texto ou uma imagem, ou ambos

Lista das classes de AWT

5. Fundamentos da Orientação a Objetos

Classe Swing	Descrição
Jlist	Um componente que exibe uma lista de objetos e permite que o usuário selecione uma ou mais artigos.
Jmenu	Uma implementação de um menu - uma janela pop-up contendo JMenuItem que é exibido quando o usuário seleciona um item no JMenuBar.
JOptionPane	torna mais fácil para abrir uma caixa de diálogo padrão que solicita aos usuários para um valor ou informa-los de alguma coisa.
Jpanel	é um recipiente leve genérico.
JRadioButton	Uma implementação de um botão de rádio - um item que pode ser selecionado ou não, e que exibe seu estado para o usuário.
JScrollBar	Uma implementação de uma barra de rolagem.
Jslider	Um componente que permite que o usuário graficamente selecionar um valor deslizando um botão dentro de um intervalo limitado.

Classe Swing	Descrição
JTextArea	Uma JTextArea é uma área multi-linha que exibe texto simples.
JTextField J	TextField é um componente leve que permite a edição de uma única linha de texto.
JTree	Um controle que exibe um conjunto de dados hierárquicos como um esboço.
JWindow	A JWindow é um recipiente que pode ser exibido em qualquer lugar da área de trabalho do usuário.
Japplet	Uma versão estendida do java.applet.Applet que adiciona suporte para o JFC / Swing arquitetura de componentes

Lista das classes de AWT

5. Fundamentos da Orientação a Objetos

O que é um Evento?

De forma simplificada, um evento é uma ação executada nos objetos existentes no programa. Por exemplo, sabe quando você posiciona o mouse em cima de uma palavra (Label) e aparece uma legenda explicativa sobre essa palavra? Então, posicionar o mouse em cima da palavra é um evento, e aparecer a legenda explicativa sobre a palavra é uma resposta ao evento.

Todos os objetos estão sujeitos a um evento. Um objeto que recebe eventos e responde a eles é chamado de detector de eventos. Os detectores de eventos são necessários para implementar interfaces que definem formalmente o modo como os eventos serão recebidos e processados.

Cada tipo de evento tem uma espécie de biblioteca que é capaz de reconhecer e responder a eventos, essas "bibliotecas" são denominadas interface do detector de eventos.

Rotinas da interface MouseListener

mouseClicked( )	usuário clicou o mouse.
mousePressed( )	usuário pressionou o botão do mouse (mas ainda não o soltou).
mouseReleased( )	usuário liberou o botão do mouse.
mouseEntered( )	seta do mouse entrou na janela do applet.
mouseExited( )	seta do mouse saiu da janela

5. Fundamentos da Orientação a Objetos

MÉTODOS COMUNS A TODOS OS COMPONENTES

void resize(int width, int height) → tamanho da tela

void move(int x, nt y) → mover componente

void setForeground(Color x) → cor do componente

void setBackground(Color y) → cor de fundo da tela

void disable( ) → desabilitando componente

void enable( ) → habilitando componente

VARIÁVEIS DE COR DEFINIDAS NO JAVA

black	Preto
blue	Azul
cyan	Cyan
darkGray	Cinza escuro
gray	Cinza
green	Verde
lightGray	Cinza claro
magenta	Magenta
orange	Laranja
pink	Rosa
red	Vermelho
white	Branco
yellow	amarelo

5. Fundamentos da Orientação a Objetos

Eventos

Um evento é uma comunicação do mundo externo para o programa que alguma coisa aconteceu. Podemos citar como exemplo o clique ou ainda o movimento do mouse. Uma das coisas mais importantes a se entender sobre o AWT é como é feito o manuseio/tratamento destes eventos. Sem eventos, sua aplicação não poderia responder às ações do usuário.

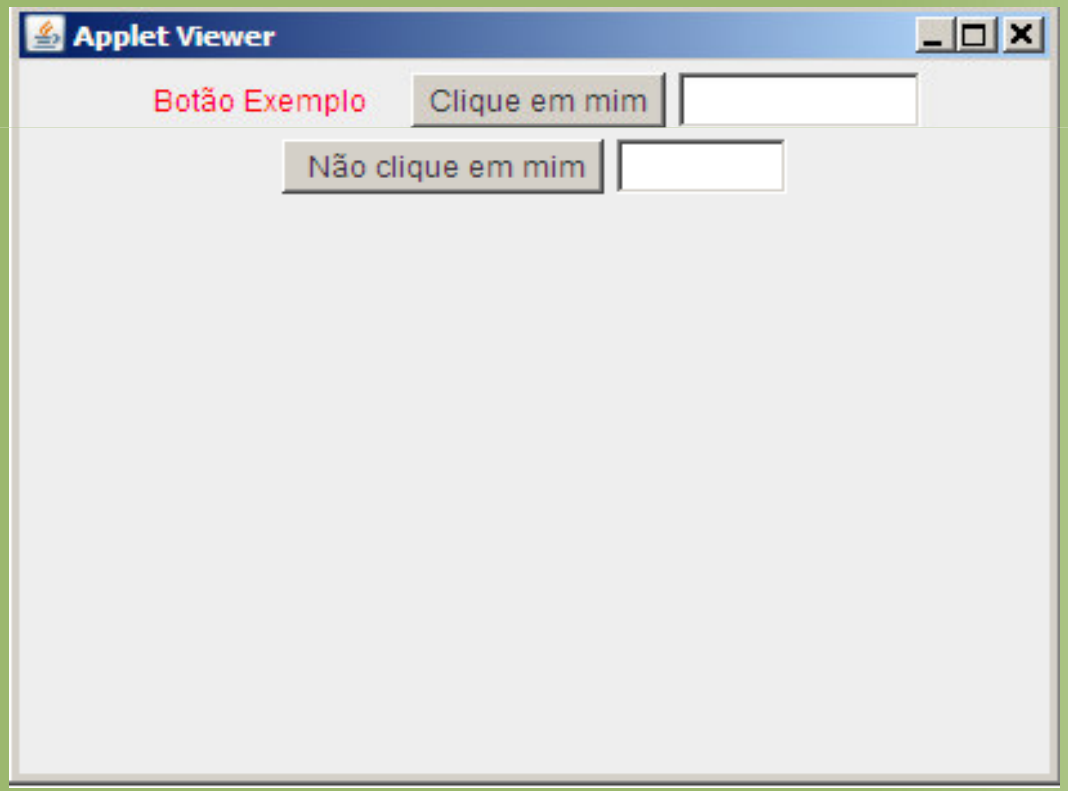
Exemplo de evento clique2:

```
import java.applet.*;   import java.awt.*;
import java.awt.event.*;
public class clique2 extends Applet implements ActionListener
{
    Label titulo;
    TextField caixa1, caixa;
    Button botao,botao1;
    public void init()
    {
          titulo = new Label("Botão Exemplo");
          titulo.setForeground (Color.red); add(titulo);
          botao = new Button("Clique em mim"); add(botao);
          botao.addActionListener(this);
          caixa= new TextField (" ",10); add (caixa);
          botao1 = new Button(" Não clique em mim"); add(botao1);
          botao1.addActionListener(this);
          caixa1= new TextField (" ",6); add (caixa1);
    }
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == botao)
            {
                  caixa.setText("Muito Obrigado!");
            }
        if (e.getSource() == botao1)
            {
                  caixa1.setText("BUMMMM!");
            }
    }
}
```

PRÁTICA2 ◘

Exercício 2.1

Desenvolver compilar e testar um applet usando os objetos Label *titulo;* TextField *caixa1, caixa2*; Button *botao1,botao2;* O programa deverá inserir uma mensagem na caixa1, quando for clicado no botão1 e inserir uma outra mensagem na caixa2, quando for clicado no botão2.



PRÁTICA2 ◘

Exercício 2.2

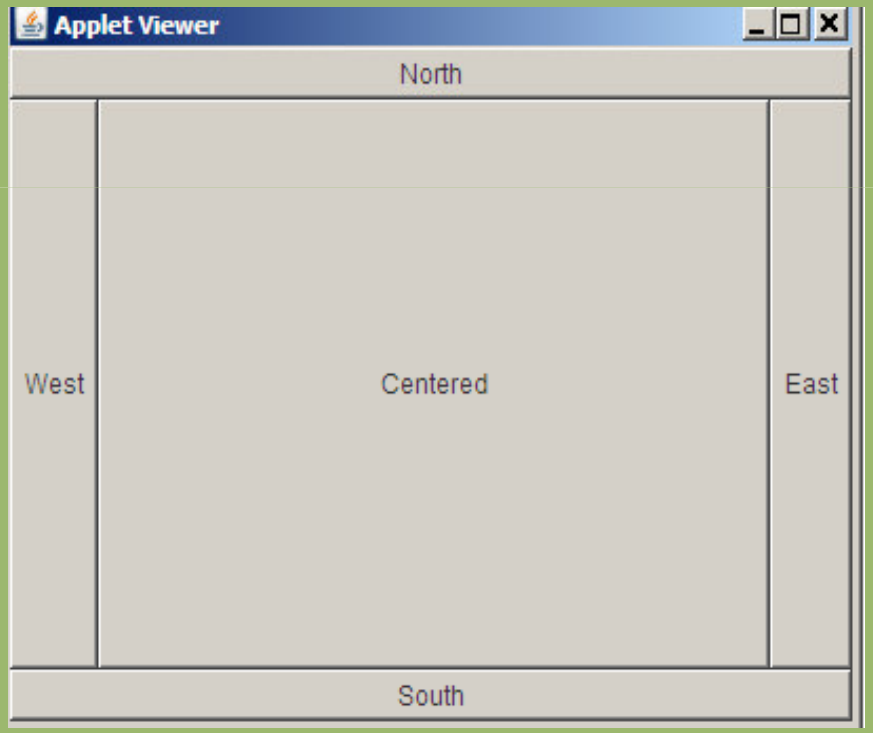
Desenvolver compilar e testar um applet usando os objetos TextField tf; Label *Lab1*; Choice *ch*; Os itens da Choice *são* "Texto"; "Retângulo"; " Círculo".

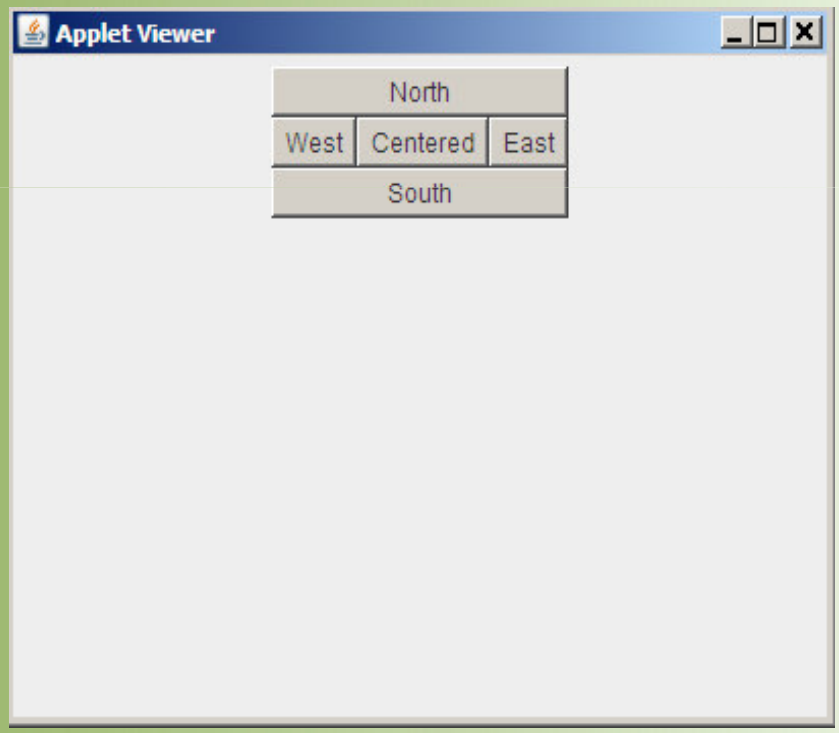
O programa deverá inserir um texto na tela quando a opção for "Texto"; se a opção for "; "Retângulo"; Desenhar um retângula e se a opção for " Círculo", desenhar um círculo.

PRÁTICA2 ◘

Exercício 2.3

Modificar o código do programa mostrado abaixo inserindo um objeto Panel e aplicar o mesmo Layout da tela anterior ao objeto Panel para que a tela fique como mostra a figura2





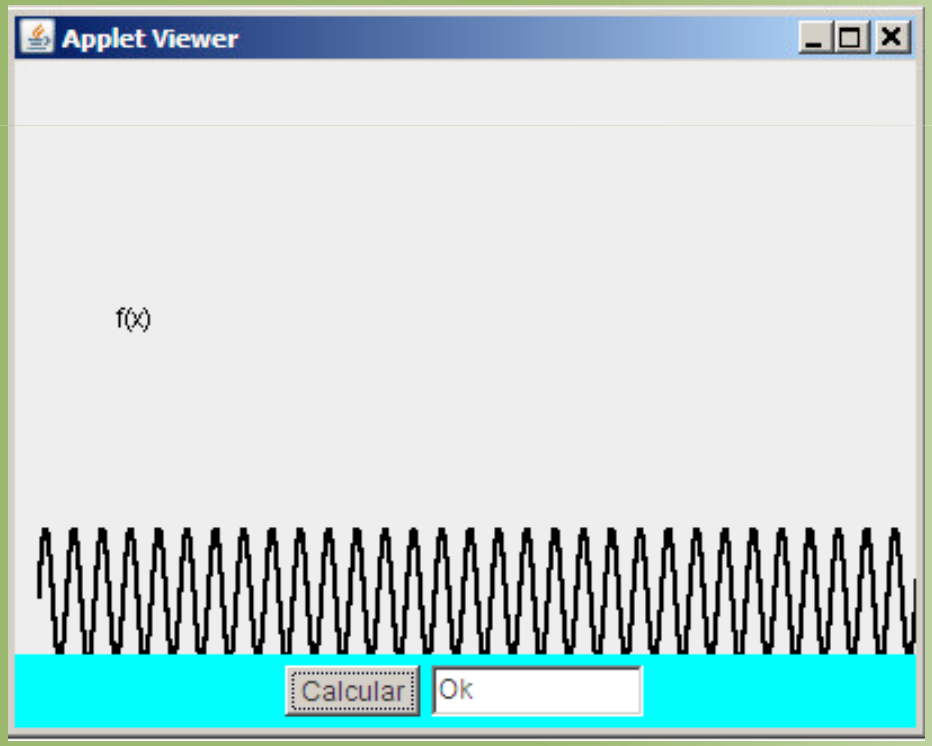
PRÁTICA2 ◘

Exercício 2.4

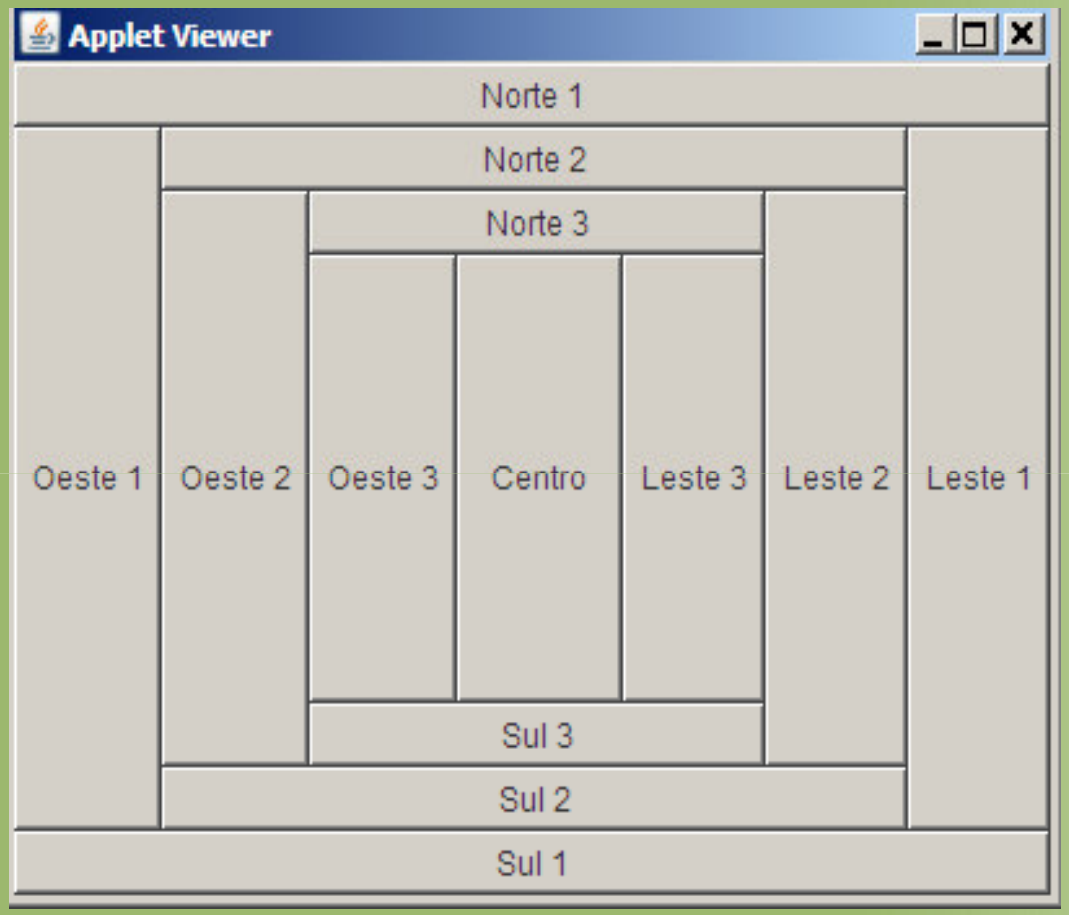
O programa mostrado na figura abaixo plota o gráfico de sem(x) no intervalo de 0 a 200 que é a largura da tela. Modificar o programa para que o mesmo plote o gráfico de duas funções Atraves da condição:

se(i<=100){f1} else {f2}

As funções f1 e f2 pedem ser determinadas.



Exemplo de Layout e Panel




```
public void init ( )
  {
     setLayout(new BorderLayout( ));
     botao1 = new Button("Norte 1");
     botao2 = new Button("Sul 1");
     botao3 = new Button("Leste 1");
     botao4 = new Button("Oeste 1");

     botao5 = new Button("Norte 2");
     botao6 = new Button("Sul 2");
     botao7 = new Button("Leste 2");
     botao8 = new Button("Oeste 2");

     botao9 = new Button("Centro");
     botao10 = new Button("Norte 3");
     botao11 = new Button("Sul 3");
     botao12 = new Button("Leste 3");
     botao13 = new Button("Oeste 3");

     panel1 = new Panel();
     panel1.setLayout(new BorderLayout( ));
     panel2 = new Panel();
     panel2.setLayout(new BorderLayout( ));

     add("North",botao1);
     add("South",botao2);
     add("East",botao3);
     add("West",botao4);

     add("Center",panel1);
     panel1.add("North",botao5);
     panel1.add("South",botao6);
     panel1.add("East",botao7);
     panel1.add("West",botao8);


     panel1.add("Center",panel2);
     panel2.add("Center",botao9);
     panel2.add("North",botao10);
     panel2.add("South",botao11);
     panel2.add("East",botao12);
     panel2.add("West",botao13);
  }
```

BIBLIOGRAFIA

1. Professor Enrique Ortega (FEA,Unicamp) e Colaboradores Mara Cornélio, Daniel Wada, Carla Lanzotti, Fábio Lemes de Souza Ana Cláudia Scachetti, Aline Gimenez, Mirian Futagawa, Mileine Furlanetti Curso de Programação de Applets em Java 2 (recursos AWT) Disponível em : http://www.unicamp.br/fea/ortega/info/curso/welcome.htm

2. MASSAGO Sadao e SCHÜTZEr Waldeck Tutorial de Programação Java Disponivel em: http://www.dm.ufscar.br/profs/waldeck/curso/java

3. CORCUERA Pedro Interfaces de usuario com AWT y Swing Dpto. Matemática Aplicada y Ciencias de la Computación Universidad de Cantabria corcuerp@unican.es

4. Grupo PET - Informática Apostila de JAVA

5. Java API Documentation. Sun Microsystems, 1995.

5. Site da Sun, http://java.sun.com/j2se/

Fim