Globalcode® The Developers Company
Modelo de Apresentação
Atenção: A Copia integral ou parcial deste material não esta autorizada.

Copyright 2005 Globalcode – The Developers Company, todos os direitos reservados

Globalcode® THE DEVELOPERS COMPANY
Modelo de Apresentação
Atenção: A Copia integral ou parcial deste material não esta autorizada.

Copyright 2005 Globalcode – The Developers Company, todos os direitos reservados

Globalcode® The Developers Company
Modelo de Apresentação
Atenção: A Copia integral ou parcial deste material não esta autorizada.

Copyright 2005 Globalcode – The Developers Company, todos os direitos reservados

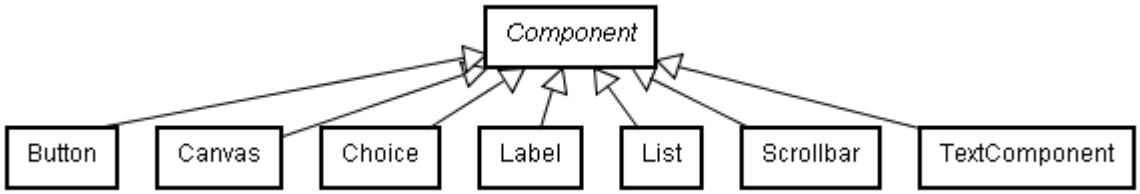
1 Interfaces Gráficas com Java

1.1 AWT (Abstract Window Toolkit)

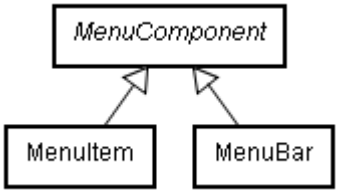
AWT é uma das APIs utilizadas para criação de interfaces gráficas e applets.

Todos os componentes de uma interface gráfica (botões, caixa de combinação etc.) são subclasses de `Component` ou `MenuComponent`.

Um `Component` é um objeto que pode ser representado graficamente em uma tela, além de interagir com o usuário. A classe abstrata `Component` é a superclasse de todos as classes não relacionadas a menu em AWT.



A classe abstrata `MenuComponent` é a super classe de todos os componentes relacionados a menu.



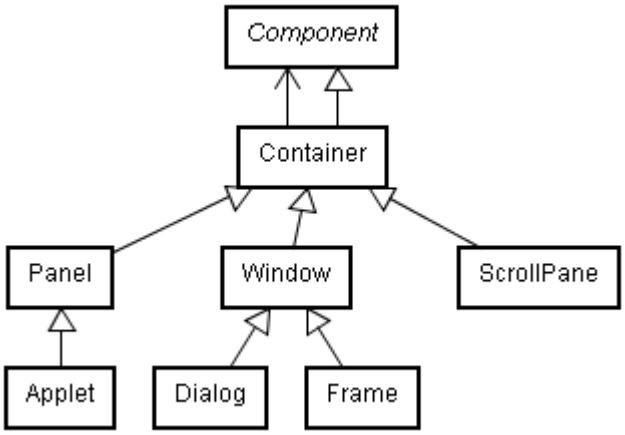
Globalcode The Developers Conference
Modelo de Apresentação
Atenção: A Cópia integral ou parcial deste material não esta autorizada.

Copyright 2005 Globalcode – The Developers Company, todos os direitos reservados

1.1.1 Janelas

Na hierarquia de classes, há uma classe chamada `Container`, que representa um objeto que poderá conter outros, como , por exemplo, `Panel` e `Window` (janelas gráficas) que, por sua vez, podem conter outros componentes.

- **`Panel`**: É o container mais simples de todos, apenas permitindo apenas que outros componentes sejam adicionados. Representa uma "sub-área" de nossa janela, onde a aplicação pode colocar qualquer outro componente, incluindo outros objetos do tipo `Container`.



- **`Applet`**: Este container deve ser visualizado através de um browser, ou então, do utilitário `appletviewer`, que é distribuído juntamente com o J2SE.
- **`Window`**: janelas sem borda ou menu, que depende de outra janela para ser exibida.
- **`Dialog`**: utilizado para apresentações de diálogos.
- **`Frame`**: janelas de aplicações convencionais que contém bordas para redimensionamento, botões de maximizar e minimizar, etc.
- **`ScrollPane`**: janelas de aplicação que possuem barras para navegação (scroll) horizontais ou verticais, podendo ser configuradas para aparecer sempre, nunca ou conforme necessário.

GlobalCode THE DEVELOPERS COMPANY Modelo de Apresentação Atenção: A Copia integral deste material não esta

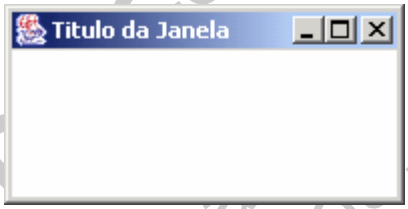
Anotações

1.1.2 Criando uma janela simples

Podemos criar uma janela estendendo a classe `Frame`, configurando seu tamanho e chamando o método `setVisible(true)` para que a janela seja exibida.

Exemplo: Frame1.java

```
package br.com.globalcode.awt;

import java.awt.*;

public class Frame1 extends Frame {

    public Frame1(String title) {
        // Estamos indicando o titulo da janela
        setTitle(title);
        // Configuracao do tamanho da janela
        setSize(200, 100);
        // Fazendo a janela aparecer
        setVisible(true);
    }
    // Ao executar a classe, o metodo main sera executado, criando uma instancia
    // do Frame1 que, por sua vez, fará com que a janela seja exibida.
    public static void main(String args[]) {
        Frame1 janela = new Frame1("Titulo da Janela");
    }
}
```



Importante:

Reparem que estas janelas não tem comportamento padrão quando o usuário clica no botão X localizado no canto superior direito. A janela simplesmente não fecha porque é necessário programar um evento que será executado quando o usuário clicar no botão.

Globalcode® The Developers Company
Modelo de Apresentação
Atenção: A Copia Integral ou parcial deste material não esta autorizada.

Copyright 2005 Globalcode – The Developers Company, todos os direitos reservados

1.1.3 Adicionando um componente ao container

A adição de qualquer componente (ou até mesmo outro `Container`) AWT é feita utilizando o método `add(Component c)` herdado da classe `Container`.

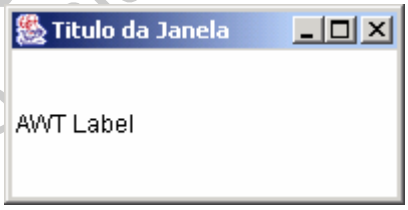
No exemplo abaixo, iremos utilizar `Label`, um dos componentes mais simples e que representa um texto em uma única linha, que não pode ser editado pelo usuário.

Um `Label` pode ser construído da seguinte forma:

```
Label label1 = new Label(“Texto do label”);
```

Exemplo: FrameComLabel.java

```
package br.com.globalcode.awt;

import java.awt.*;

public class FrameComLabel extends Frame {

    public FrameComLabel(String title) {
        // Estamos indicando o titulo da janela
        setTitle(title);
        // Configuracao do tamanho da janela
        setSize(200, 100);

        Label l = new Label("AWT Label");
        add(l);
        // Fazendo a janela aparecer
        setVisible(true);
    }

    // Ao executar a classe, o metodo main sera executado, criando uma instancia
    // do    Frame1, que por sua vez faz com que a janela seja exibida.
    public static void main(String args[]) {
        FrameComLabel janela = new FrameComLabel("Titulo da Janela");
    }
}
```



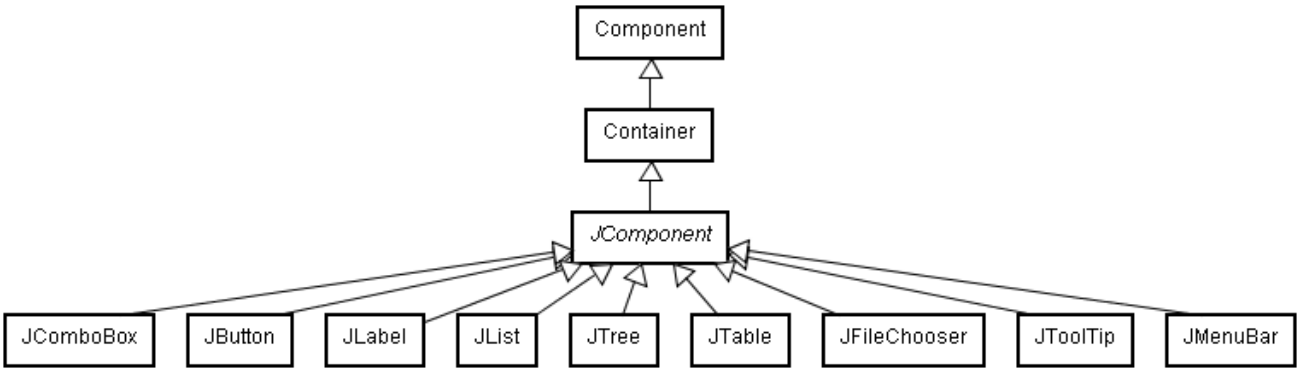
Material de Apresentação GlobalCode - A Cópia integral ou parcial deste material não é autorizada.

Anotações

1.2 Swing

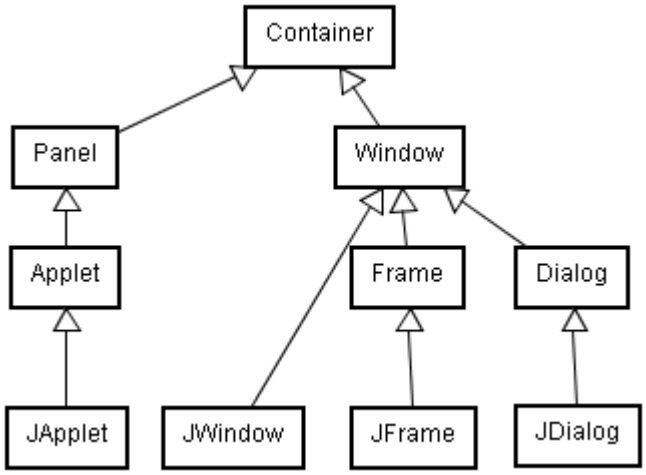
Swing é a API mais recente para desenvolvimento de interfaces gráficas baseada em AWT, mas muito mais rica em componentes.
Esta API não depende de código nativo para representar botões, janelas e outros componentes gráficos, sendo totalmente independente de plataforma e desenhando seus próprios componentes.

A classe `JComponent` é a superclasse de todos os componentes que não são containers de outros, sendo derivada de `Component`, como podemos observar no seguinte diagrama de classes:



1.2.1 Containers

Vejamos o diagrama abaixo:



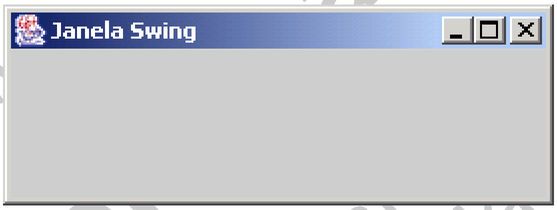
Material para uso exclusivo de cópia integral ou parcial não esta autorizada

Copyright 2005 Globalcode – The Developers Company, todos os direitos reservados

1.2.2 Criando uma janela simples

Exemplo: FrameSwing1.java

```java
package br.com.globalcode.swing;

import javax.swing.*;

public class FrameSwing1 extends JFrame {

    public FrameSwing1() {
        super("Janela Swing");
        setSize(275, 100);
        show();
    }

    public static void main(String args[]) {
        FrameSwing1 t = new FrameSwing1();
    }
}
```



Importante:

Esta janela possui comportamento padrão `HIDE_ON_CLOSE` no botão fechar (X no canto superior direito), ou seja, quando clicamos, a janela é "escondida", mas o processo continua sendo executado. Recomendamos configurar o comportamento de fechamento default, utilizando o seguinte método da classe `JFrame`, que provoca o término da execução.:

```java
setDefaultCloseOperation(EXIT_ON_CLOSE);
```

Exemplo: FrameSwing2.java

```java
package br.com.globalcode.swing;
import javax.swing.*;

public class FrameSwing2 extends JFrame {

  public FrameSwing2() {
    super("Janela Swing");
    // Utilizamos a constante declarada na classe JFrame para definir
    // o comportamento padrao no fechamento da janela
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setSize(275, 100);
    show();
  }
  public static void main(String args[]) {
    FrameSwing2 t = new FrameSwing2();
  }
}
```

Anotações

1.2.3 Adicionando um componente ao container

A manipulação e adição de componentes no `JFrame` é um pouco diferente da que vimos em AWT, pois nunca manipulamos diretamente o `JFrame`.Devemos sempre obter uma referência para o único `Container` contido no `JFrame`, obtido através da chamada ao método `getContentPane()`.

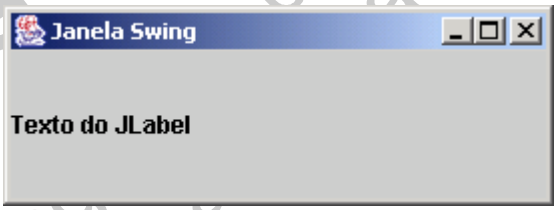
No exemplo abaixo, iremos utilizar o `JLabel`, um dos componentes mais simples e que representa o texto em uma única linha, não podendo ser editado pelo usuário.

Um `JLabel` pode ser construído da seguinte forma:

```
JLabel label1 = new JLabel(“Texto do jlabel”);
```

Exemplo: FrameSwingComLabel.java

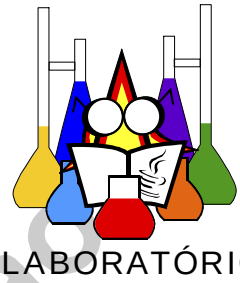
```
package br.com.globalcode.swing;

import java.awt.Container;

import javax.swing.*;

public class FrameSwingComLabel extends JFrame {

    public FrameSwingComLabel() {

        super("Janela Swing");

        // Utilizamos a constante declarada na classe
        // JFrame para definir comportamento padrao
        // no fechamento da janela
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        JLabel label = new JLabel("Texto do JLabel");

        Container c = this.getContentPane();
        c.add(label);

        setSize(275, 100);
        show();
    }

    public static void main(String args[]) {
        FrameSwingComLabel t = new FrameSwingComLabel();
    }
}
```



Globalcode Apresentação
Atenção: A cópia integral ou parcial deste material não está autorizada.

Copyright 2005 Globalcode – The Developers Company, todos os direitos reservados

1.2.4 Laboratório 1



Objetivo:
Praticar a criação de janelas.

Tabela de atividades

Atividade	***OK***
1. Faça o download do arquivo **aj4lab01_01.zip** na URL indicada pelo instrutor(a).	
2. Descompacte o arquivo em seu diretório de trabalho.	
3. Compile e teste as classes `Frame1.java` e `FrameSwing.java`.	

GlobalCode THE DEVELOPERS COMPANY
Modelo de Apresentação
Atenção: A Copia integral ou parcial deste material não esta autorizada.

Anotações

1.3 Principais Componentes Swing

1.3.1 javax.swing.JButton

Representa um botão com rótulo.

Construtores

- **`JButton(String texto):`** o valor da variável texto será apresentado no botão;
- **`JButton(Icon icone):`** o ícone será apresentado no botão.

Observação: Uma das formas de criarmos um ícone é a partir da classe `ImageIcon`, que pode ser construída a partir de uma `String` representando o nome da imagem.
Exemplo: `ImageIcon icone = new ImageIcon("icone.gif")`
Para mais detalhes consulte o Javadoc.

Métodos

- **`void setMnemonic(int mnemonic)`** : Podemos associar atalhos aos botões utilizando este método, que recebe um `int` como parâmetro, representado pelas constantes definidas na classe `KeyEvent`.

Exemplos de constantes:

`KeyEvent.VK_A` => Atalho para A
`KeyEvent.VK_B` => Atalho para B
`KeyEvent.VK_1` => Atalho para 1

Exemplo: TesteJButton.java

```
package br.com.globalcode.swing;
import java.awt.*;
import javax.swing.*;
import java.awt.event.KeyEvent;

public class TesteJButton extends JFrame {

    public TesteJButton() {
        super("Teste JButton");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        Container c = getContentPane();

        JButton botaoOk = new JButton("Ok");
        botaoOk.setMnemonic(KeyEvent.VK_O);
        c.add(botaoOk);

        setSize(200, 50);
        show();
    }
    public static void main(String args[]) {
        TesteJButton t = new TesteJButton();
    }
}
```

GlobalCode The Developers Company
Material de Apresentação
Aviso: A Cópia integral ou parcial deste material não esta autorizada.

Copyright 2005 Globalcode – The Developers Company, todos os direitos reservados

Exemplo: TesteJButtonComIcone.java

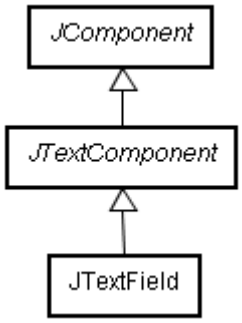
```
package br.com.globalcode.swing;

import java.awt.*;
import javax.swing.*;

public class TesteJButtonComIcone extends JFrame {

    public TesteJButtonComIcone() {
        super("Teste JButton");
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        Container c = getContentPane();

        // Criacao de um icone com uma imagem
        ImageIcon iconeBotao = new ImageIcon("duke.gif");
        // Criacao de um botao com o icone iconeBotao
        JButton botaoIcone = new JButton(iconeBotao);
        // Alteramos a cor de fundo do botao para ficar compativel com a imagem
        botaoIcone.setBackground(Color.WHITE);

        c.add(botaoIcone);
        setSize(80, 150);
        show();
    }

    public static void main(String args[]) {
        TesteJButtonComIcone t = new TesteJButtonComIcone();
    }
}
```

Modelo de Apresentação
Atenção: A Cópia integral ou parcial deste material não está autorizada.

Anotações

1.3.2 javax.swing.JTextField

Esta classe representa um campo de texto digitável, usualmente empregado para campos de cadastro de uma única linha.



Exemplo: TesteJTextField.java

```
package br.com.globalcode.swing;

import java.awt.*;
import javax.swing.*;

public class TesteJTextField extends JFrame {

    public TesteJTextField() {
      super("Teste JTextField");
      this.setDefaultCloseOperation(EXIT_ON_CLOSE);
      Container c = getContentPane();

      JTextField textField = new JTextField();
      c.add(textField);

      setSize(200, 50);
      show();
    }

    public static void main(String args[]) {
      TesteJTextField t = new TesteJTextField();
    }
}
```

Métodos para manipulação do conteúdo de um JTextField

- **`String getText()`:** retorna o valor do texto contido no `JTextField`;
- **`void setText(String texto)`:** atribui o valor da variável `texto` ao `JTextField`;
- **`void setEditable(boolean editable)`:** habilita ou desabilita o componente de texto. Este método é herdado da classe `JTextComponent`.

Globalcode® The Developers Company
Módulo de Apresentação
Atenção: A cópia integral ou parcial deste material não é autorizada.

Copyright 2005 Globalcode – The Developers Company, todos os direitos reservados

1.3.3 javax.swing.JComboBox

Construtores

- **`JComboBox();`**
- **`JComboBox(Object[] itens);`**
- **`JComboBox(Vector itens).`**

Principais métodos

- **`void addItem(Object o):`** adiciona um item na ultima posição;
- **`Object getSelectedItem():`** retorna o item selecionado;
- **`void insertItemAt(Object item, int posicao):`** insere um objeto na posição especificada;
- **`Object getItemAt(int posição):`** retorna o item que estiver na posição especificada ou null se ele não existir;
- **`void removeAllItens():`** remove todos os itens do `JComboBox`;
- **`void removeItemAt(int posicao):`** remove o item que estiver na posição especificada;
- **`void setEnabled(boolean habilitado):`** habilita ou não o `JComboBox`;
- **`void setSelectedItem(Object item):`** configura qual será o item selecionado;
- **`void setSelectedIndex(int posicao):`** configura qual será o item selecionado através da sua posição no `JComboBox`.

O componente `JComboBox` apresenta uma lista com scroll de itens, podendo ser configurado para que cada elemento possa ser editável.

É possível construirmos um `JComboBox` passando como parâmetro um `Vector`, contendo os elementos que queremos exibir no `JComboBox`; o texto exibido será o resultado da chamada ao método `toString` de cada componente.

Neste exemplo, estamos adicionando objetos do tipo `String` ao `JComboBox` através do método `addItem(String)`.

Modelo de Apresentação
Atenção: A Cópia Integral ou parcial deste material não esta autorizada

Anotações