Programação II

Prof. Alcemir Rodrigues Santos

Introdução ao Swing

Agenda

- Introdução do Intefaces Gráficas com Java
- Swing
  - MVC
  - Look and Feel
  - Componentes
  - Eventos e Listeners
- Exercícios

Interface Gráfica Java

- Atualmente, o Java suporta, oficialmente, dois tipos de bibliotecas gráficas: **AWT** e **Swing**.
  - A AWT foi a primeira API para interfaces gráficas a surgir no Java
  - Mais tarde, superada pelo Swing (a partir do Java 1.2), que possui diversos benefícios em relação a seu antecessor.
- As bibliotecas gráficas são bastante simples no que diz respeito a conceitos necessários para usá-las.
  - A complexidade no aprendizado de interfaces gráficas em Java reside no tamanho das bibliotecas e no enorme mundo de possibilidades

Interface Gráfica Java

- AWT e Swing são bibliotecas gráficas oficiais inclusas em qualquer JRE ou JDK
- Além destas, existem algumas outras bibliotecas de terceiros
  - sendo a mais famosa, o SWT - desenvolvida pela IBM e utilizada no Eclipse e em vários outros produtos.
  - O GWT (http://www.gwtproject.org/) fornece uma lista enorme de elementos muito utilizados e comuns variando desde os básicos aos mais complexos.

Introdução Swing

- Swing – um conunto de classes de Interface Gráfica de Usuário (GUI)
  - Parte da biblioteca padrão de Java
  - Melhor que a biblioteca anterior: Abstract Window Toolkit (AWT)
- Destaques
  - Um conjunto rico widgets
    - widget: quaisquer elemento GUI (também chamados de componentes)
  - Conteúdos e formas são separados (apoio ao padrão MVC)
  - Controle refinado sobre o comportamento e aparência
  - Independente de plataforma
    - Isola o programador da GUI do sistema operacional

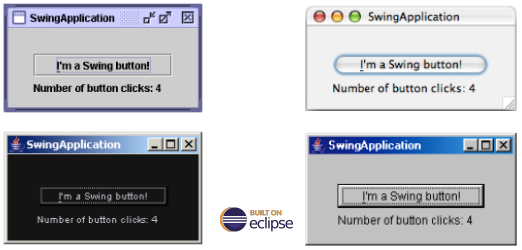
Architecture (MVC)

- A arquitetura da API Swing acompanha a arquitetura MVC da seguinte maneira
  - Um **modelo** representa os dados do componente
  - Uma **visão** respresenta uma representação visual do dados dos componentes
  - Um **controlador** recebe a entrada de um usuário na visão e reflete as mudanças nos dados do componente
- Um componente Swing tem um Modelo como um elemento separado, enquanto as partes de Visão e o Controlador são agrupados como elementos de interface de usuário.
  - Por conta disto, Swing tem uma arquitetura que permite usar diferentes look-and-feel (aparência)

Look-and-Feel

- **Look-and-Feel** (ou LaF) é o nome que se dá à "cara" da aplicação (suas cores, formatos e etc).
  - Por padrão, o Java vem com um look-and-feel próprio,
  - que se comporta exatamente da mesma forma em todas as plataformas suportadas.
- Além do padrão do Java, a partir do JRE 5 da Sun disponibilizou LaF nativos para Windows e Mac OS, além do Motif e GTK.
  - E, fora esses, você ainda pode baixar diversos LaF na Internet ou até desenvolver o seu próprio.

Look-and-Feel



Portabilidade

- As APIs de interface gráfica do Java favorecem, ao máximo, o lema de portabilidade da plataforma Java
- A **aparência** do Swing é único em todas as plataformas onde roda, seja ela Windows, Linux, ou qualquer outra.
  - Isso implica que a aplicação terá exatamente a mesma interface (cores, tamanhos etc) em qualquer sistema operacional.

Portabilidade

- Grande parte da complexidade das classes e métodos do Swing está no fato da API ter sido desenvolvida tendo em mente o máximo de portabilidade possível.
  - Favorece-se, por exemplo, o posicionamento relativo de componentes, em detrimento do uso de posicionamento fixo, que poderia prejudicar usuários com resoluções de tela diferentes da prevista.
- Com Swing, não importa qual sistema operacional, qual resolução de tela, ou qual profundidade de cores: sua aplicação se comportará da mesma forma em todos os ambientes.

Controles Swing

- Elementos UI
  - Os principais elementos visuais que o usuário vê e interage.
- Layouts
  - Definem como os elementos da UI devem ser organizados na tela e fornecem uma aparência final para a GUI.
- Comportamento
  - São eventos que ocorrem quando os usuários interagem com os elementos UI

Componentes Swing

- Containers
  - Acomodam e gerenciam outros componentes
  - Podem ser usados como componentes principais ou internos
  - Exemplos: `JFrame` (*componente principal*), `JScrollPane`, `JPanel`.
- Controles Básicos
  - Componentes atômicos
  - Usados para mostrar saída ou capturar alguma entrada
  - Herdam de `JComponent`
  - Exemplos: `JButton`, `JLabel`, `JTextArea`, `JTable`, `JList`

Componentes Swing

- Geralmente todo componente Swing extende a classe AWT correspondente
  - Isso acontece por motivos de retro-compatiblidade
- A biblioteca do Swing está no pacote `javax.swing` (inteira, exceto a parte de acessibilidade, que está em javax.accessibility).

Hello Swing

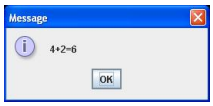
- A classe mais simples do Swing é a JOptionPane que mostra janelinhas de mensagens, confirmação e erros, entre outras.
  - `JOptionPane.showMessageDialog(null, "Minha mensagem!");`

Meu Primeiro Programa Swing

```
public class HelloWorldSwing {
    public static void main(String[] args) {
        // Cria e configura a janela.
        JFrame frame = new JFrame("HelloWorldSwing");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Adiciona uma label "Hello World" ubíqua.
        JLabel label = new JLabel("Hello World");
        frame.getContentPane().add(label);

        // Exibe a janela.
        frame.pack();
        frame.setVisible(true);
    }
}
```


JDialog

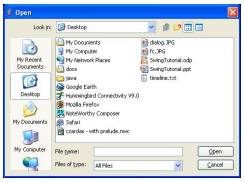
- javax.swing.JDialog:
  - Mais simples e limitado que os frames
  - Tipicamente usado para mostrar uma mensagem curta na tela
  - Também tem uma borda e uma barra de título
  - Pode ter um proprietário
    - Se o proprietário é invisível o diálogo irá ser invisível também
  - Use o método estático do JOptionPane para mostrar uma caixa de diálogo padrão




```
JOptionPane.showMessageDialog(null, "4+2=6");
```


JFileChooser

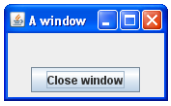
- javax.swing.JFileChooser:
  - Permite que o usuário escolha um (ou muitos) arquivos
  - Permite "abrir" e "salvar"
    - showOpenDialog()
    - showSaveDialog()




```
JFileChooser fc = new FileChooser();
int returnVal = fc.showOpenDialog(null);
if(returnVal == JFileChooser.APPROVE_OPTION)
    System.out.println("File: " +
        fc.getSelectedFile());
```


JFrame

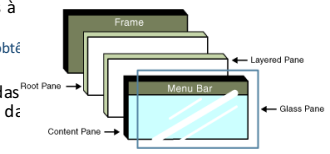
- javax.swing.JFrame:
  - Janela de maior nível, que contém um título e uma borda
  - Geralmente utilizada como a principal janela do programa




```
private JFrame frame;
JFrame frame = new Jframe("Minha Janela");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Mais sobre o JFrame

- Composto com várias camadas
- Os widgets são adicionados à camada "Content Pane"
  - Usar getContentPane() para obté lo
- Outras camadas são utilizadas para customizar a aparência da janela



Containers Internos

- Containers que podem ser utilizados dentro dos componentes macro
- Podem ser compostos de outros componentes internos

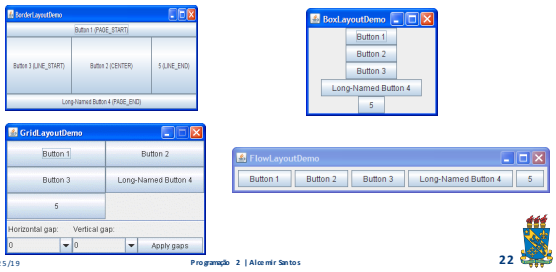
- Exemplos:
  - **JScrollPane**: permite "rolar" os componentes internos
  - **JSplitPane**: separado em dois componentes
  - **JTabbedPane**: pode conter abas que o usuário escolhe qual ver

Layouts

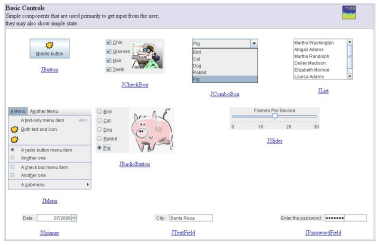
- Cada conteiner tem um gerenciador de layout
  - Este gerenciado determina o tamanho, localização de cada componente interno
- Configurar o gerenciador de layout de um conteiner
  `void setLayout(LayoutManager lm)`

- LayoutManager implementando as classes
  - BorderLayout
  - BoxLayout
  - FlowLayout
  - GridLayout

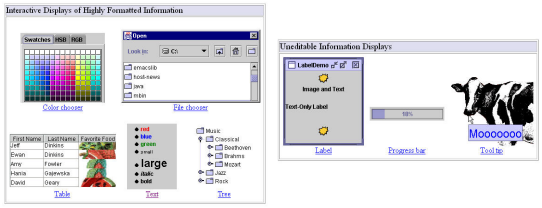
Containers - Layout



Componentes Swing



Componentes Swing



Revendo o primeiro programa Swing

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

public class First {
  public static void main(String[] args) {
    JFrame frame = new JFrame("My First Frame");

    // operation to do when the window is closed.
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    frame.getContentPane().setLayout(new BorderLayout());
    frame.getContentPane().add(new JLabel("I Love Swing"),
       BorderLayout.CENTER);
    frame.pack();
    frame.setVisible(true);
  }
}
```

Cria um frame

Escolhe o BorderLayout como gerenciador

Cria uma label de texto

Especifica CENTER como a posição do layout

Adiciona o labe ao painel de conteúdo

Entrada de dados

- Aprendemos como apresentar os componentes na tela
- Um programa também precisa reagir às ações do usuário
- Exemplos:
  - Quando um usuário pressiona um botão, queremos salvar um arquivo
  - Quando o usuário fechar o programa, queremos perguntar "Você tem certeza?"
  - ...
- Mecanismo Swing: `Events` e `Listeners`

Events, Listeners

- Swing define todas as formas de interfaces de Listener
  - ActionListener, MouseMotionListener, WindowListener, ...
- Existem implementações padrão (vazias) para muitos dos listeners
  - MouseMotionAdapter, WindowAdapter

```
public interface ActionListener extends EventListener {
    public void actionPerformed(ActionEvent e);
}

public interface MouseMotionListener extends EventListener {
    public void mouseDragged(MouseEvent e);
    public void mouseMoved(MouseEvent e);
}
```


Events, Listeners (cont.)

- Um listener é um objeto que implementa a interface Listener
- Se precisamos reagir um evento (em um certo widget) precisamos retistrar um objeto listener naquele widget
  - **addActionListener()** registra um listener de ação no seu receptor:

```
JButton button = new JButton();
ActionListener listener = ...;
button.addActionListener(listener);
```

- Quando um evento ocorre, todos os listeners registrados são notificados
  - O método apropriado do listener (e.g. `actionPerformed()`) é invocado
  - Um objeto descrevendo o evento é passado como um parâmetro

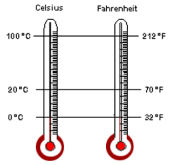
Tratamento de Eventos

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

public class Events implements ActionListener {
   public Events() {
    JFrame frame = new JFrame("Events");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    frame.getContentPane().setLayout(new FlowLayout());
    JButton b = new JButton("Click me!");
    b.addActionListener(this);
    frame.getContentPane().add(b);

    frame.pack();
    frame.setVisible(true);
  }
```

```
public void actionPerformed(ActionEvent e) {
   JOptionPane.showMessageDialog(null, "Thank you");
 }
 public static void main(String[] args) { new Events(); }
}
```

Exercício

Faça um algoritmo em Java usando Swing para realizar a conversão de graus **Fahrenheit** para **Celsius** (ou vice-versa), onde, o usuário irá informar o valor da temperatura em Fahrenheit e o programa deverá exibir o valor convertido em Celsius.



$$\frac{°C}{5} = \frac{°F - 32}{9}$$

Exercício

Faça um programa em Java usando Swing que, dados os coeficientes **a, b** e **c** de uma equação de segundo grau **(ax2 + bx + c = 0),** exibir o **delta** da equação.

Entrada de dados:
- Coeficiente A: **a**
- Coeficiente B: **b**
- Coeficiente C: **c**

$$\Delta = b^2 - 4ac$$

Saída de dados:
- Delta: **d**

Exercício

Faça um algoritmo em Java usando Swing que receba do usuário o **tempo** e a **distância** e, em seguida, calcule e exiba a **velocidade média**.

$$\text{Velocidade Média} = \frac{\text{Distância}}{\text{Tempo}}$$