Programação Gráfica

SCC0604 - Programação Orientada a Objetos

Prof. Fernando V. Paulovich
http://www.icmc.usp.br/~paulovic
paulovic@icmc.usp.br

Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)

25 de julho de 2010

Introdução

- Java oferece dois pacotes básicos para a programação de interface gráfica: *AWT* e *Swing*
- O pacote *AWT* foi o pioneiro para a criação de interfaces gráficas Java, sendo o mesmo baseado em "semelhante"
- O pacote *Swing*, substitui o *AWT* na parte de desenhar a interface gráfica com as vantagens de depender menos da plataforma subjacente e de conseguir manter um mesmo padrão entre várias plataformas (as vezes isso é um problema)

Criando um Quadro que pode ser Fechado

- Em um nível mais alto, uma janela em Java é denominada Quadro (Frame)
- Em *AWT*, a classe para implementar um quadro chama **Frame** e em *Swing* chama **JFrame**
- Os quadros são exemplos de contêineres, sendo que os mesmos podem conter outros elementos (botões, campos de texto,etc.)

Criando um Quadro que pode ser Fechado

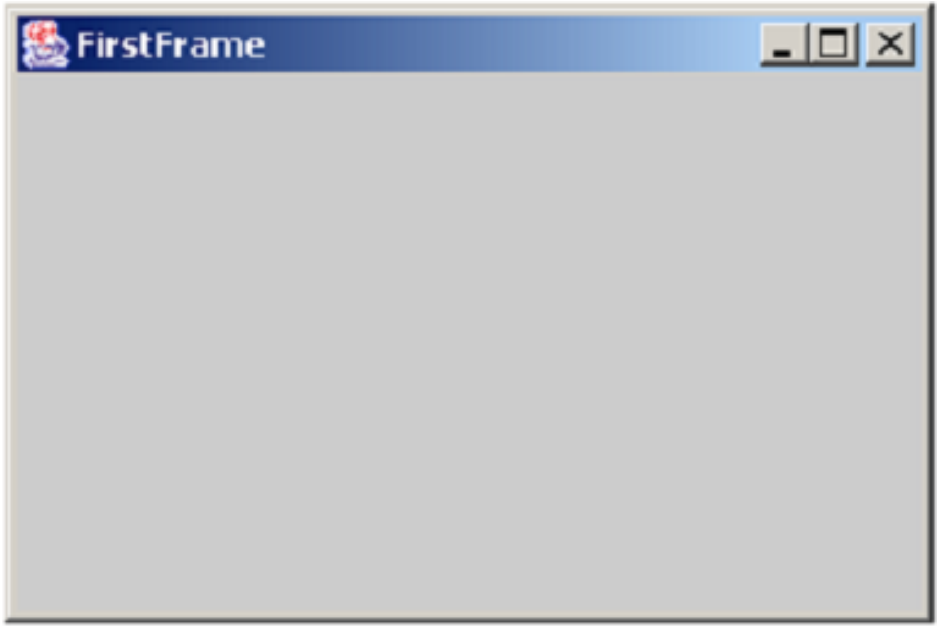
- Para se utilizar o pacote *Swing*, o mesmo deve ser importado: **javax.swing.***
- Para se criar um quadro usando o *Swing*, a classe **JFrame** deve ser derivada em uma nova classe

```
1 public class FirstFrame extends JFrame {
2   ...
3 }
```

Criando um Quadro que pode ser Fechado

```
1  import javax.swing.*;
2
3  public class FirstFrame extends JFrame {
4    public FirstFrame() {
5      setTitle("FirstFrame"); //dá um nome para o quadro
6      setSize(300,200); //informa o tamamho do quadro
7      setLocation(300,300); //posiciona o quadro na tela
8    }
9
10   public static void main(String[] args) {
11     JFrame frame = new FirstFrame();
12     frame.show(); //ou frame.setVisible(true)
13   }
14 }
```

Resultado



Como Encerrar Programas Gráficos

- Temos um problema: não há uma forma de se fechar o programa que criamos - apenas ocultamos o quadro criado!
- Assim, precisamos de alguma forma de sermos notificados de que o quadro foi fechado - isso pode ser feito com base no modelo de eventos *AWT* (**java.awt.event**)
- Para se controlar eventos *AWT* de quados, é necessário criar uma classe que implemente a interface **WindowsListener**

Como Encerrar Programas Gráficos

- Classes que implementam a interface **WindowListener** precisam implementar os seguintes métodos
  - public void windowActivated(WindowEvent e)
  - public void windowClosed(WindowEvent e)
  - public void windowClosing(WindowEvent e)
  - public void windowOpened(WindowEvent e)
  - ...

Como Encerrar Programas Gráficos

- Por conveniência, vamos estender uma classe que já implementa esses métodos e vamos alterar apenas o método que cuida do fechamento do quadro (**windowClosing()**)
- Essa classe é conhecida como **WindowAdapter** - na verdade a implementação que ela oferece são métodos de corpo vazio

O Código

```
1  import java.awt.event.*;
2
3  class Terminator extends WindowAdapter {
4    public void windowClosing(WindowEvent e) {
5      System.exit(0);
6    }
7  }
```

Conectando o Quadro ao Tratador de Eventos

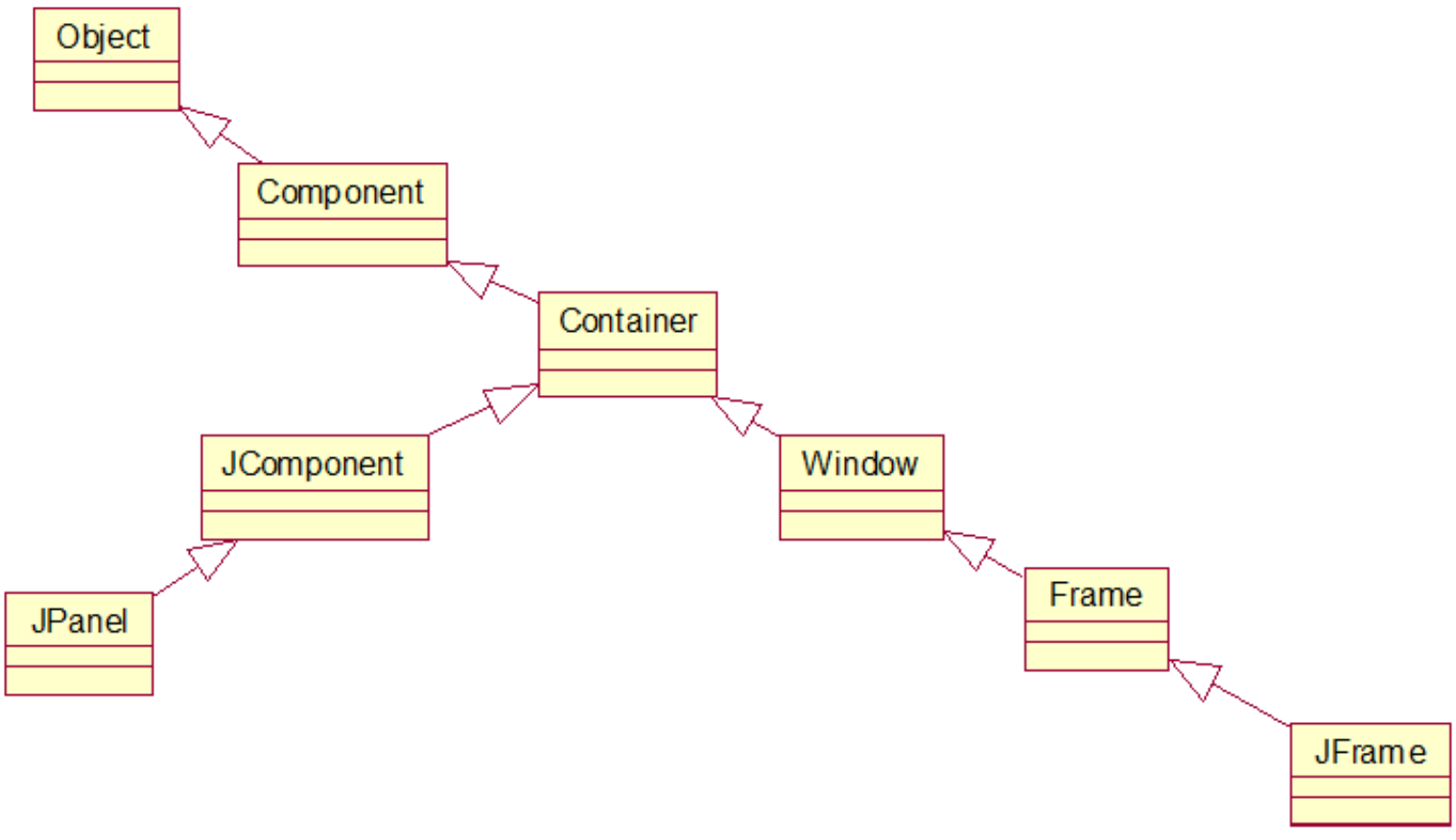
- Após criar essa classe precisamos informar ao quadro qual é o objeto que está controlando os eventos desse quadro por meio do comando **addWindowListener()**

```
1  public class FirstFrame extends JFrame {
2    public FirstFrame() {
3      setTitle("FirstFrame");
4      setSize(300,200);
5      setLocation(300,300);
6      addWindowListener(new Terminator());
7    }
8
9    ...
10 }
```

Layout de Quadros

- A classe **JFrame** contém poucos métodos para configurar os quadros, a maioria é herdado de classes progenitoras
- Dessa forma, na documentação Java (API Java), muita coisa deve ser consultada nas classes progenitoras

Layout de Quadros



Layout de Quadros

- O tamanho e a posição de criação do quadro dependem do monitor que está sendo usado, assim no momento da criação de um quadro, se torna interessante saber qual a resolução do monitor e se basear nela para criar o quadro
- Normalmente, as informações dependentes do sistema podem ser consultadas por meio da classe **java.awt.Toolkit**

Layout de Quadros

```
1  public class FirstFrame extends JFrame {
2    public FirstFrame() {
3      setTitle("FirstFrame");
4      addWindowListener(new Terminator());
5
6      Toolkit tk = Toolkit.getDefaultToolkit();
7      Dimension d = tk.getScreenSize();
8      setBounds(d.width/4, d.height/4, d.width/2, d.height/2);
9    }
10
11   ...
12 }
```

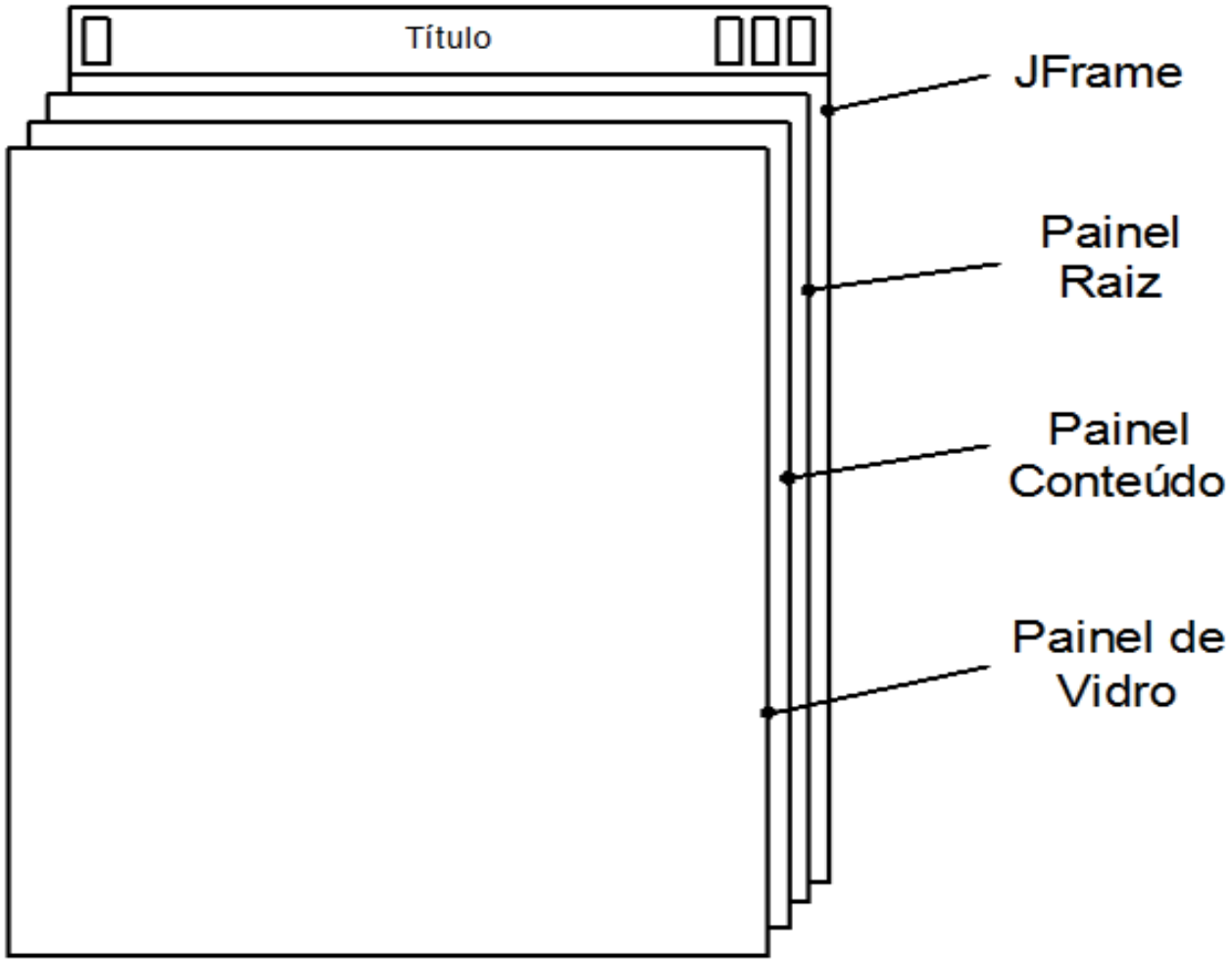
Exibindo Informações em um Quadro

- Como os quadros são projetados apenas para serem contêineres, há uma necessidade de se criar um outro elemento, adicioná-lo ao quadro e desenhar ou escrever sobre esse elemento
- Tal elemento é conhecido como Painel e é implementado na classe **JPanel**
- Um painel é um elemento que tem uma superfície onde é possível desenhar, além de ser também um recipiente que pode receber outros elementos

Exibição de Informações em um Quadro

- A estrutura de um **JFrame** é bem complexa, onde quatro camadas são superpostas para determinadas finalidades
- Dentre essas camadas, a de nosso interesse é a camada de conteúdo, pois nela podem ser adicionados outros elementos

Adicionando Informações a um Quadro



Adicionando Informações a um Quadro

- Para se inserir um painel à área de conteúdo de um quadro, usamos o seguinte código

```
1 Container painelConteudo = frame.getContentPane();
2 JPanel p = new JPanel();
3 painelConteudo.add(p);
```

Objetos Gráficos e o Método paintComponent

- Sempre que você precisar de um componente de interface de usuário que seja semelhante a um dos componentes *Swing* básicos, poderá usar herança para criar um nova classe e depois sobrepor ou adicionar métodos para obter a funcionalidade desejada

Objetos Gráficos e o Método paintComponent

- Dessa forma, para desenhar um painel, será necessário
  - Definir uma classe que estenda **JPanel**
  - Sobrepor o método **paintComponent()** dessa classe
- O método **paintComponent()** recebe como parâmetro um objeto **Graphics**, que representa um contexto de dispositivo do Windows ou um contexto gráfico na programação X11

Objetos Gráficos e o Método paintComponent

```
1  class MyPanel extends JPanel {
2    public void paintComponent(Graphics g) {
3      super.paintComponent(g);
4      g.drawString("Isto é um teste.", 75, 100);
5    }
6  }
```

Usando o Panel

```
1  public class FirstFrame extends JFrame {
2    public FirstFrame() {
3    ...
4    Container contentPane = getContentPane();
5      contentPane.add(new MyPanel());
6    }
7    ...
8  }
```

Resultado

Texto e Fontes

- A fonte a ser usada para desenho pode ser alterada dentro de um objeto **Graphics** por meio do método **setFont()**, passando um objeto fonte
- Um objeto fonte é configurado passando o nome da fonte, o tipo (negrito, itálico, etc.) e seu tamanho
- Maiores informações sobre a configuração de fontes podem ser obtidas na sessão Texto e Fontes do livro Core Java 2, Volume I - Fundamentos

Texto e Fontes

```
1  public void paintComponent(Graphics g) {
2     super.paintComponent(g);
3
4     Font f = new Font("SansSerif", Font.BOLD, 30);
5     g.setFont(f);
6     g.drawString("Isto é um teste.", 75, 100);
7  }
```

Cores

- É possível modificar as cores usadas para as operações de desenho no contexto gráfico de um componente. Para isso, o método **setColor()** pode ser usado
- O método **setColor()** recebe como parâmetro um objeto do tipo **Color**, que por sua vez já define algumas constantes de cores

Cores

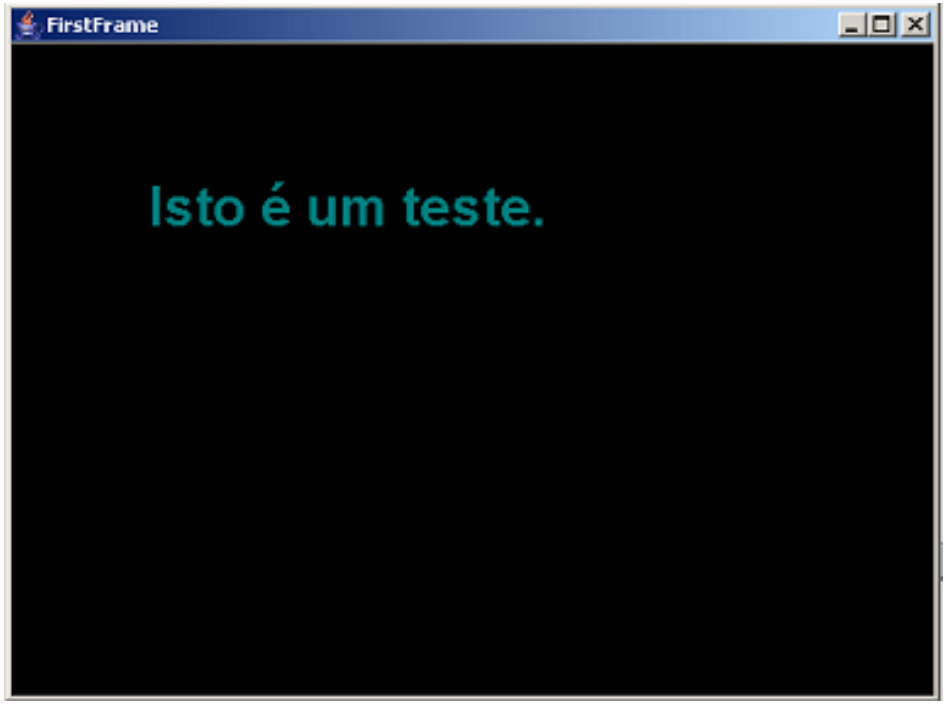
```
1  public class MyPanel extends JPanel {
2
3    public void paintComponent(Graphics g) {
4      super.paintComponent(g);
5
6      g.setColor(new Color(0,128,128));
7
8      Font f = new Font("SansSerif", Font.BOLD, 30);
9      g.setFont(f);
10     g.drawString("Isto é um teste.", 75, 100);
11   }
12 }
```

Cores

- Para se especificar cores de fundo, o método **setBackground()** pode ser utilizado, passando um objeto **Color**

```
1 public void paintComponent(Graphics g) {
2    super.paintComponent(g);
3    this.setBackground(Color.BLACK);
4    ...
5 }
```

Cores



Desenhando Formas Geométricas

- É possível dentro de um painel desenhar formas geométricas como linhas, círculos, quadrados, etc.
- Para saber mais,consulte a seção Como Desenhar Formas Geométricas com Linhas (pág. 262-277) do livro Core Java 2, Volume I - Fundamentos