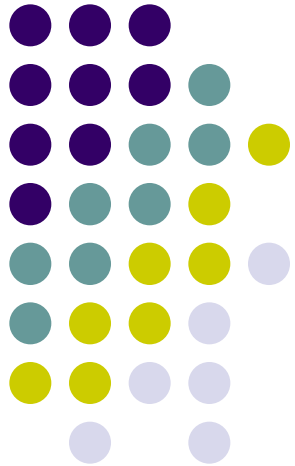
COM220

Aula 16: Interfaces Gráficas em Java

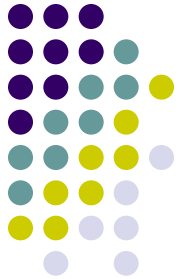
Prof. Laércio Baldochi



Java Foundation Classes - JFC

- Conjunto de tecnologias desenvolvidas para suportar a criação de interfaces gráficas e aprimorar a interação com o usuário em aplicações Java
  - Abstract Window Toolkit (AWT)
  - Swing
  - Java 2D
  - Internationalization

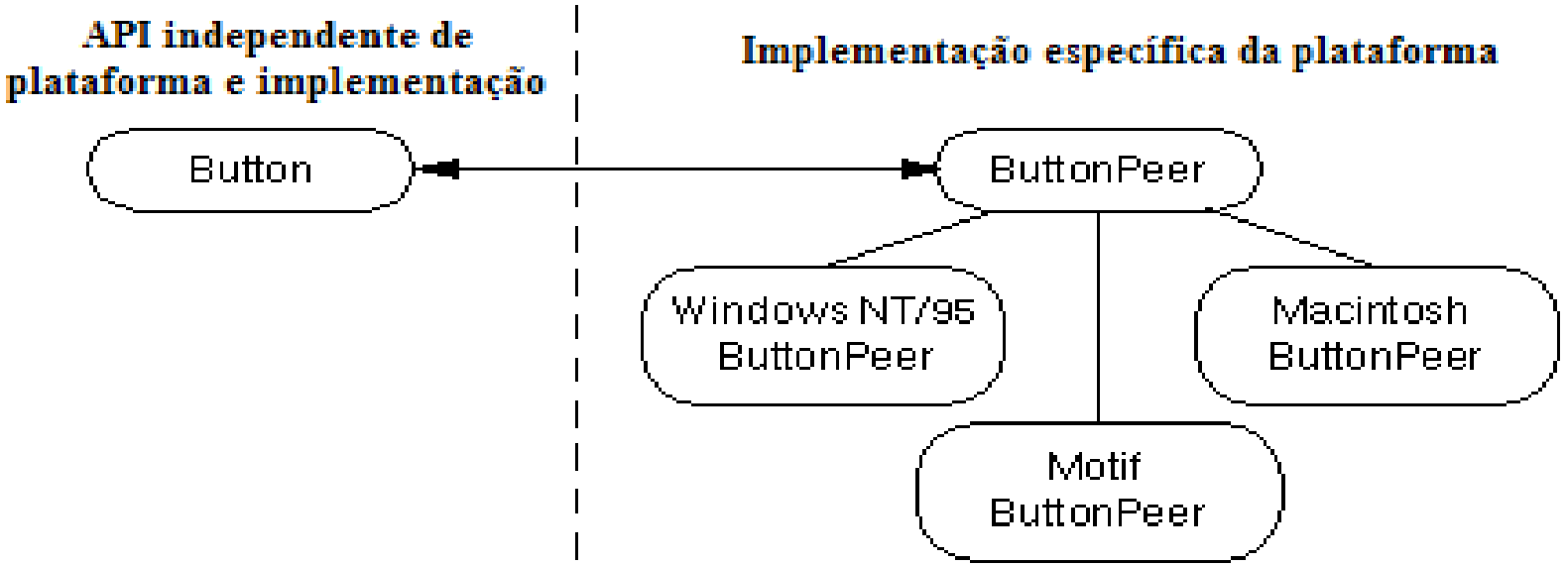
Abstract Window Toolkit



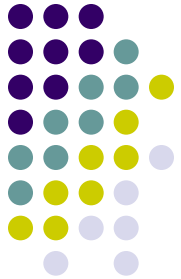
- Desenvolvida inicialmente para suportar a construção de interfaces simples em applets
  - Conjunto restrito de componentes gráficos e classes
    - Gerenciamento de layout
    - Manipulação de eventos
- Componentes gráficos "pesados"
  - Peers
    - Componentes nativos do sistema operacional, específicos de cada plataforma
    - Escondem as peculiaridades de cada plataforma
      - Interface comum de programação

Abstract Window Toolkit

- Arquitetura baseada em peers
  - Apresentou problemas de eficiência e compatibilidade

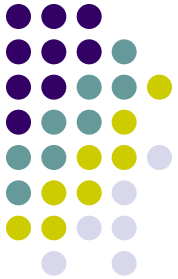


Swing

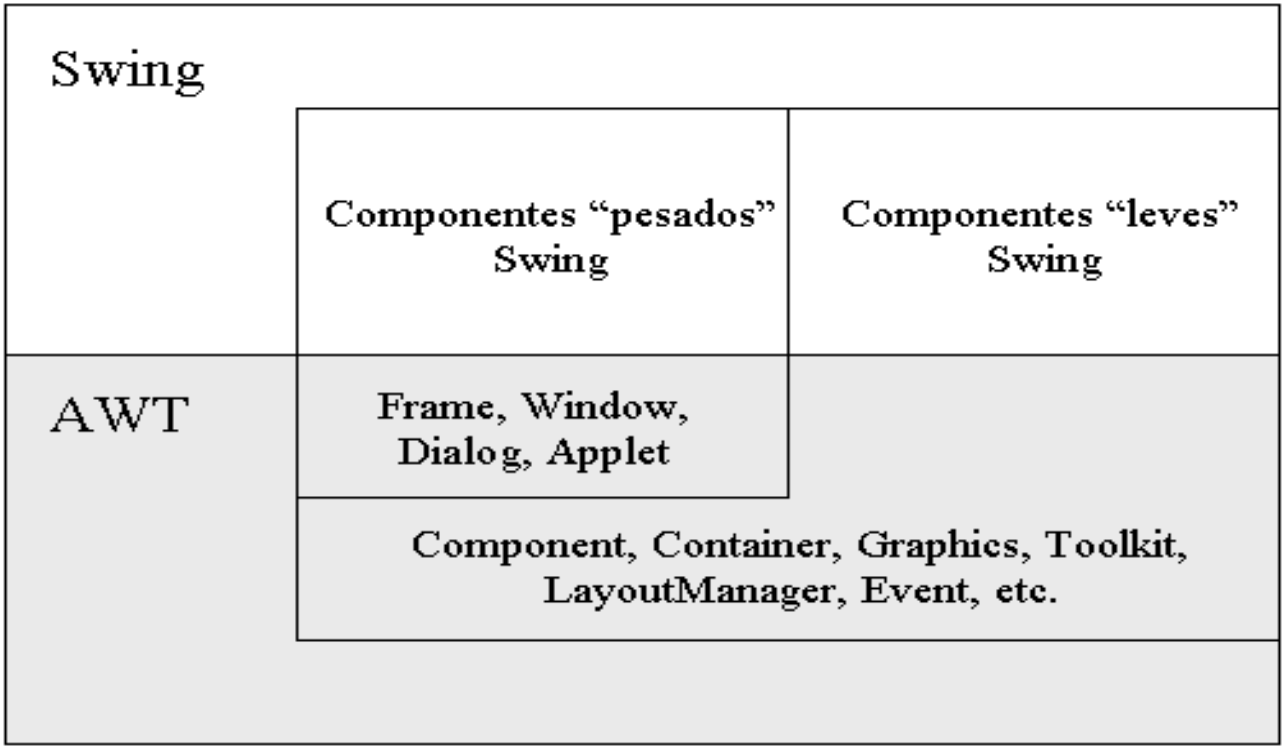


- Principal API da JFC
- Desenvolvido no topo do AWT
  - Soluciona limitações
  - Complementa AWT com novos GUIs
- Possui 4 classes que descendem de componentes AWT
  - JFrame, JDialog, JWindow, JApplet
    - Componentes "pesados"
- Demais componentes são desenhados diretamente na tela
  - Componentes "leves"

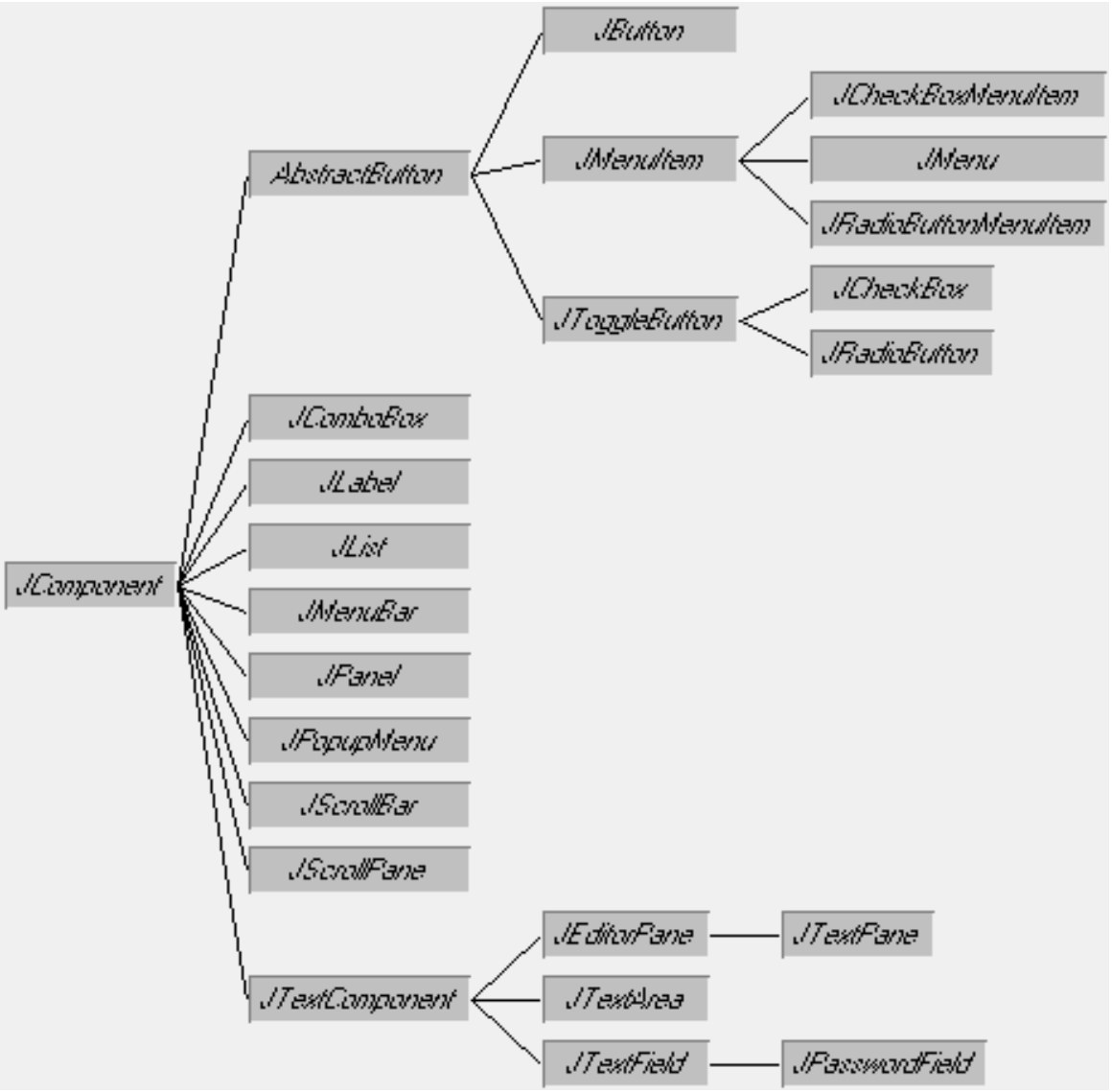
Relação AWT <-> Swing



- Swing utiliza as classes não visuais do AWT
  - Gerenciadores de layout, containeres e manipuladores de eventos

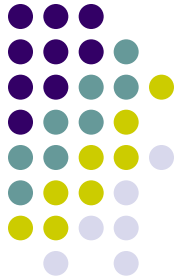


API Swing



- Contém mais de 250 classes
  - 40 são componentes gráficos.
  - Mais importantes estendem a classe JComponent

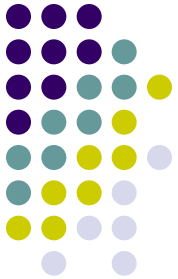
Manipulação de eventos



- Modelo de delegação de tratamento
  - Cada componente deixa a cargo de objetos específicos a manipulação de seus eventos
    - Listeners (escutam os eventos)

- Listener
  - Implementam uma interface comum para cada tipo de evento
  - Devem ser registrados no componente de interesse para receber notificação de eventos

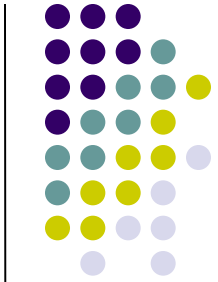
Usando Listeners



```
public class C implements ActionListener{
...
JButton button1 = new JButton("Clique aqui");
button1.addActionListener(this);
....
public void actionPerformed(ActionEvent e) {
      numClicks++;
      label.setText(rotulo + numClicks);
            }
```

- Usando listeners
  - Listener deve ser adicionado ao componente
    - addActionListener
  - Classe deve implementar sua interface associada
    - implements ActionListener()
  - Classe deve declarar os métodos de sua interface associada
    - public void actionPerformed(ActionEvent e)

Uso de listeners
Exemplo 1

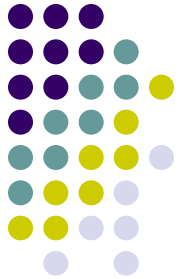


```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class ButtonApplication implements ActionListener {
      private static String rotulo =  "Número de cliques: ";
      private int numClicks  = 0;
       private JLabel label;

      public Component createComponents() {
               label  = new JLabel(rotulo + "0    ");
               JButton button1 = new JButton("Clique aqui");
               //Adicionando listener ao botao
               button1.addActionListener(this);
               label.setLabelFor(button1);
               JPanel pane = new JPanel();
               pane.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));
               pane.add(button1);
               pane.add(label);
               return pane;
      }

      public void actionPerformed(ActionEvent e) {
               numClicks++;
               label.setText(rotulo + numClicks);
      }
```

Uso de listeners
Exemplo 1

```
    public static void main(String[] args) {

            //Cria container e adiciona a ele elementos de interface.
            JFrame frame = new JFrame("Primeira Aplicação Swing");
            ButtonApplication app = new ButtonApplication();

            Component conteudo = app.createComponents();
            frame.getContentPane().add(conteudo, BorderLayout.CENTER);

            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.pack();
            frame.setVisible(true);
    }
}
```

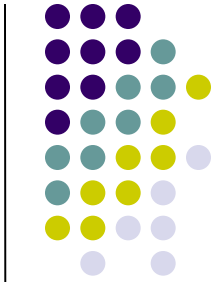
Uso de listeners



- O exemplo 1 ilustra a forma mais simples para associar eventos a componentes Swing
  - Outra possiblidade é colocar todo o código como argumento do método addActionListener

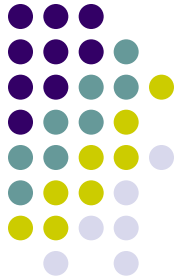
```
button1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        numClicks++;
        label.setText(rotulo + numClicks);
    }
});
```

Uso de listeners
Exemplo 2



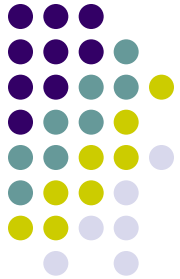
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class ButtonApplication {
    private static String rotulo =  "Número de cliques: ";
    private int numClicks  = 0;

    public Component createComponents() {
        final JLabel label  = new JLabel(rotulo + "0    ");
        JButton button1 = new JButton("Clique aqui");
        //Adicionando listener ao botao
        button1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                numClicks++;
                label.setText(rotulo + numClicks);
            }
        });

        label.setLabelFor(button1);
        JPanel pane = new JPanel();
        pane.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));
        pane.add(button1);
        pane.add(label);
        return pane;
    }
```

Uso de listeners
Exemplo 2



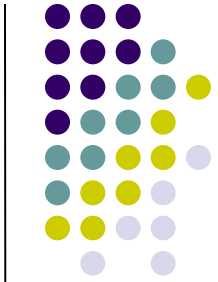
```
    public static void main(String[] args) {

            //Cria container e adiciona a ele elementos de interface.
            JFrame frame = new JFrame("Primeira Aplicação Swing");
            ButtonApplication app = new ButtonApplication();

            Component conteudo = app.createComponents();
            frame.getContentPane().add(conteudo, BorderLayout.CENTER);

            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.pack();
            frame.setVisible(true);
    }
}
```

Tratando eventos do mouse



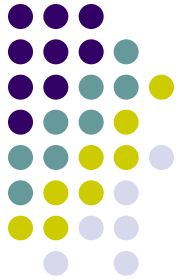
- Para tratar eventos do mouse a linguagem Java disponibiliza
  - Classe mouseEvent
  - Interface mouseListener
    - mouseEntered
    - mouseExited
    - mousePressed
    - mouseReleased
    - mouseClicked

Exercício 1



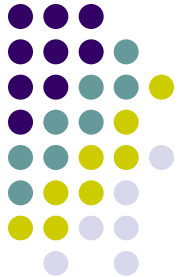
- Adapte o Exemplo 2 de modo que ele passe a tratar os eventos do mouse associados ao objeto button1

Tratando eventos do mouse



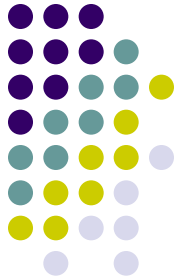
- O tipo do evento está associado ao método executado durante o processo de callback
  - Detalhes do evento são reportados pelo objeto MouseEvent
  - Métodos
    - getButton
    - getClickCount
    - getPoint
    - getWhen
    - getX
    - getY

Exercício 2



- É possível reimplementar o exemplo 2 de modo a permitir a contagem de cliques do mouse, porém **sem** utilizar uma variável contadora?

Exercício 3



- Implemente uma interface com 2 botões
  - Salva
  - Cancela
- Quando houver o clique em um botão, informar num JOptionPane o botão clicado