Programação I

Introdução a Desenvolvimento Visual

Samuel da Silva Feitosa

Aula 18

Introdução

- Nesta aula vamos começar os estudos para desenvolver aplicações gráficas em Java.
  - Utilizaremos a biblioteca Java/Swing.
- Veremos diversos formatos de layout e alguns componentes visuais.
  - Estudaremos também o uso de caixas de diálogos.

O que é AWT e Swing? (1)

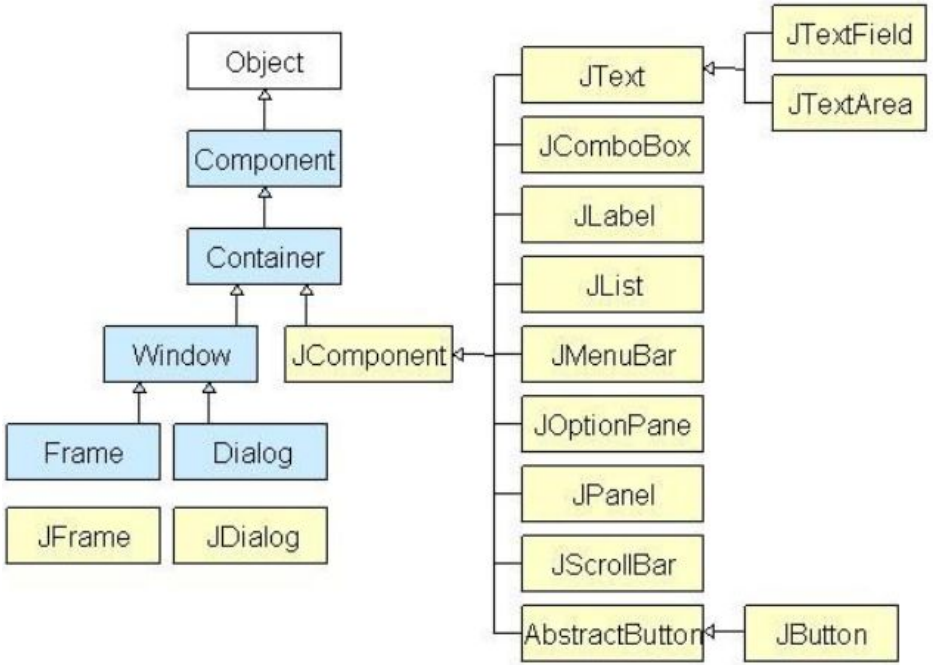
- Java Swing é um **kit de ferramentas** para desenvolver **Interface Gráfica de Usuário (GUI)**, incluindo um grande conjunto de widgets (componentes visuais).
- Permite criar aplicações Java com **janelas**, **campos de texto**, **botões** e outros componentes visuais que são **independentes de plataforma**.

O que é AWT e Swing? (2)

- A biblioteca *Swing* foi construída a partir do Java *Abstract Widget Toolkit* (AWT), uma antiga ferramenta para construção de interfaces gráficas.
- Então é possível usar componentes visuais como janelas, botões e campos de texto da biblioteca sem que seja necessário a criação de componentes do zero.

Componentes Java Swing

- Todos os componentes visuais são instâncias da classe JComponent, e podem ser utilizados nas janelas.



O que é uma classe container?

- Classes **container** são classes que podem ter outros componentes dentro delas.
- Assim, para criar uma GUI precisamos pelo menos um objeto container:
  - **Panel:** É um container puro, com propósito de organizar os componentes na tela.
  - **Frame:** É uma janela completa, com título e ícone.
  - **Dialog:** Pode ser visto como um pop-up, usado para apresentar mensagens na tela.
- Vejamos um exemplo (JFrame).

Exemplo: JFrame (1)

- Código para criar uma janela completa.
    - Importante definir a operação padrão no fechamento da janela.

```
public static void main(String[] args) {
    JFrame frame = new JFrame("Janela de exemplo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 300);

    frame.setVisible(true);
}
```

Exemplo: JFrame (2)

- Adicionando um botão na Janela criada.

```
public static void main(String[] args) {
    JFrame frame = new JFrame("Janela de exemplo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 300);

    // Criando o objeto que representa um botão
    JButton button1 = new JButton("Botão 1");
    // Tratando o evento de clique do botão
    button1.addActionListener(evt -> System.out.println("Clicou!"));
    // Adicionando o botão na tela
    frame.getContentPane().add(button1);

    frame.setVisible(true);
}
```

Exemplo: JFrame (3)

- Criando um método para lidar com os detalhes.

```
public static void main(String[] args) {
    JFrame frame = new JFrame("Janela de exemplo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 300);

    createDefaultWindow(frame);

    frame.setVisible(true);
}

private static void createDefaultWindow(JFrame f) {
    JButton button1 = new JButton("Botão 1");

    button1.addActionListener(evt ->
            JOptionPane.showMessageDialog(null, "Clicou no botão 1"));

    f.getContentPane().add(button1);
}
```

Exemplo: JFrame (4)

- Adicionando dois botões.
  - Note a sobreposição dos elementos visuais.

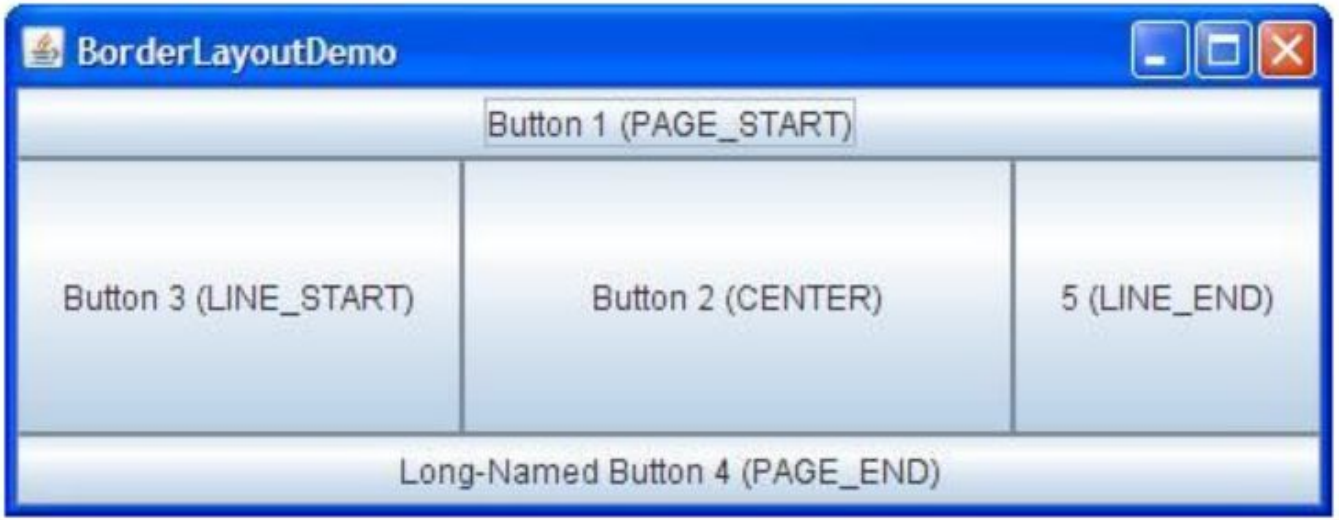
```
private static void createDefaultWindow2(JFrame f) {
    JButton button1 = new JButton("Botão 1");
    JButton button2 = new JButton("Botão 2");
    f.getContentPane().add(button1);
    f.getContentPane().add(button2);
}
```

Gerenciador de Layouts

- O gerenciador de Layout é usado para **organizar** os componentes visuais dentro de um container.
- Existem vários tipos de gerenciadores de layout.
- Veremos alguns deles a seguir.

Java *BorderLayout*

- Permite organizar os componentes visuais em até cinco áreas: topo, base, esquerda, direita e centro.
  - É o gerenciador de layout padrão para todo JFrame.



Exemplo *BorderLayout*

- Adicionando vários botões na tela com o formato de *BorderLayout*.
  - Como é o estilo padrão, não é preciso nenhuma informação extra.

```
private static void createBorderLayoutWindow(JFrame f) {
    JButton btn1 = new JButton("Botão 1 (parte cima)");
    f.getContentPane().add(btn1, BorderLayout.PAGE_START);
    JButton btn2 = new JButton("Botão 2 (centralizado)");
    f.getContentPane().add(btn2, BorderLayout.CENTER);
    JButton btn3 = new JButton("Botão 3 (parte baixo)");
    f.getContentPane().add(btn3, BorderLayout.PAGE_END);
    JButton btn4 = new JButton("BTN4");
    f.getContentPane().add(btn4, BorderLayout.LINE_START);
}
```

Java FlowLayout

- Organiza os componentes em uma única linha, um após o outro.
  - É o gerenciador de layout padrão para todo JPanel.

Exemplo *FlowLayout*

- Adicionando vários botões e definindo o uso de *FlowLayout*.

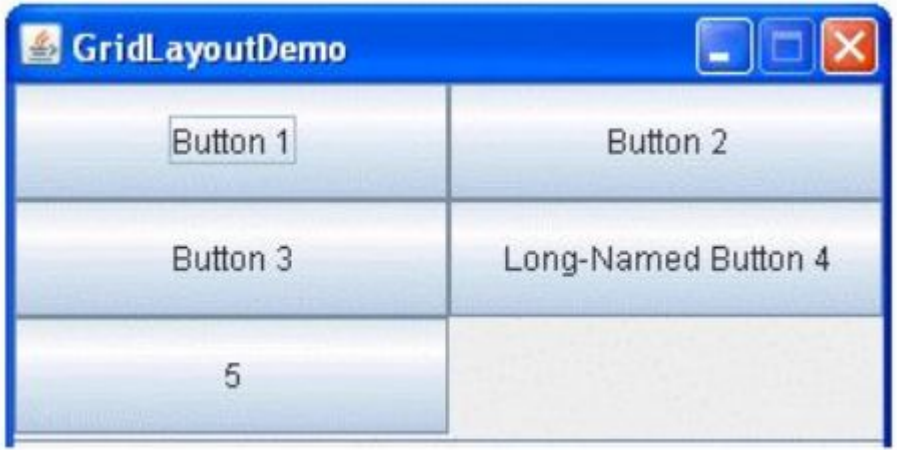
```
private static void createFlowLayoutWindow(JFrame f) {
    FlowLayout layout = new FlowLayout();

    JButton btn1 = new JButton("Botão 1");
    JButton btn2 = new JButton("Botão 2");
    JButton btn3 = new JButton("Botão 3");
    JButton btn4 = new JButton("Botão 4");

    f.getContentPane().setLayout(layout);
    f.getContentPane().add(btn1);
    f.getContentPane().add(btn2);
    f.getContentPane().add(btn3);
    f.getContentPane().add(btn4);
}
```

Java *GridLayout*

- É mais sofisticado que os outros layouts.
  - Possibilita alinhar os componentes dentro de um grid de células.
  - Permite que componentes utilizem mais de uma célula.



Exemplo *GridLayout*

- Adicionando vários botões e definindo o uso de *GridLayout*.
  - *GridLayout* recebe por parâmetro o número de linhas e colunas.

```
private static void createGridLayoutWindow(JFrame f) {
    GridLayout layout = new GridLayout(0, 1);

    JButton btn1 = new JButton("Botão 1");
    JButton btn2 = new JButton("Botão 2");
    JButton btn3 = new JButton("Botão 3");
    JButton btn4 = new JButton("Botão 4");

    f.getContentPane().setLayout(layout);
    f.getContentPane().add(btn1);
    f.getContentPane().add(btn2);
    f.getContentPane().add(btn3);
    f.getContentPane().add(btn4);
}
```

Caixas de Diálogo

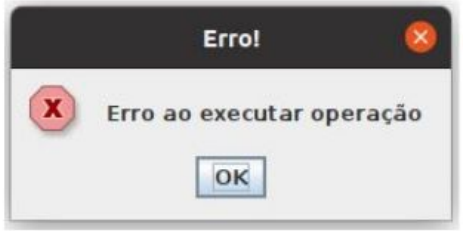
- O Java oferece janelas de diálogo com o usuário através da classe *JOptionPane*.
  - É possível criar **mensagens de alerta**, **telas de confirmação** e **telas para entrada de informação**.
- É um formato bem simples de se comunicar com o usuário.
- Vejamos alguns exemplos.

Exemplo *showMessageDialog*

- Usado para criar uma mensagem informativa ao usuário.

```
JOptionPane.showMessageDialog(null, "Clicou no botão 1");
```


```
JOptionPane.showMessageDialog(null, "Erro ao executar operação",
        "Erro!", JOptionPane.ERROR_MESSAGE);
```



Exemplo *showInputDialog*

- Usado para obter uma informação simples do usuário.

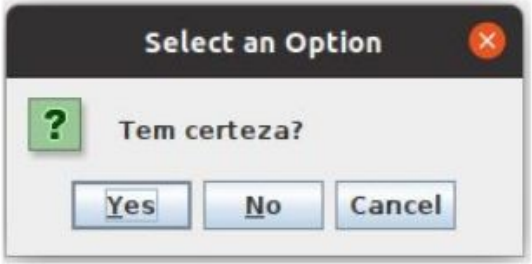
```
String nome = JOptionPane.showInputDialog("Informe o nome:");
```

Exemplo *showConfirmDialog*

- Usado para obter uma informação simples do usuário.

```
int resp = JOptionPane.showConfirmDialog(null, "Tem certeza?");

if (resp == JOptionPane.YES_OPTION) {
    // Trata a resposta
}
```



Considerações Finais

- Existem diversas bibliotecas para construção de interface gráfica em Java.
- Utilizaremos a Swing, que apesar de ser uma das primeiras criadas, ainda é bastante utilizada.
- Nas próximas aulas utilizaremos os recursos do NetBeans para facilitar a construção das janelas.