O que é o JOptionPane?

O JOptionPane é uma classe do pacote javax.swing que fornece métodos para **exibir caixas de diálogo** gráficas. Essas caixas são usadas para:

- Exibir mensagens (informativas, erros, etc.)
- Obter entradas do usuário (como texto, número, etc.)
- Pedir confirmações (sim, não)

Ele é muito útil para criar interfaces gráficas simples e interativas, sem precisar de código muito complexo.

1. Exibindo Mensagens Simples

O JOptionPane pode ser usado para **mostrar mensagens ao usuário**, como mensagens de **informação**, **erro**, **aviso**, etc.

Exemplo: Caixa de Mensagem de Informação

```
import javax.swing.JOptionPane;  // Necessário importar o JOptionPane

public class ExemploMensagem {
   public static void main(String[] args) {
      // Exibe uma caixa de mensagem com título e mensagem
      JOptionPane.showMessageDialog(null, "Bem-vindo ao Java!", "Mensagem", JOptionPane.INFORMATION_MESSAGE);
   }
}
```

- O primeiro parâmetro null indica que a caixa de mensagem não tem uma janela pai (se fosse dentro de uma janela principal, passaríamos um objeto dessa janela).
- O segundo parâmetro é a **mensagem**.
- O terceiro é o **título** da caixa de diálogo.
- O quarto parâmetro é o **tipo de mensagem** (informação, erro, etc.).

Outros tipos de mensagens:

- JOptionPane.ERROR_MESSAGE – Exibe uma mensagem de erro.
- JOptionPane.WARNING_MESSAGE – Exibe uma mensagem de aviso.
- JOptionPane.PLAIN_MESSAGE – Exibe uma mensagem sem ícone.

2. Capturando Entrada do Usuário

O JOptionPane também pode ser utilizado para **capturar entradas do usuário** com caixas de diálogo de **entrada de texto**.

Exemplo: Caixa de Entrada de Texto

```
import javax.swing.JOptionPane;

public class EntradaTexto {
   public static void main(String[] args) {
      // Captura o nome do usuário
      String nome = JOptionPane.showInputDialog(null, "Qual é o seu nome?", "Entrada de Texto", JOptionPane.QUESTION_MESSAGE);

      // Exibe uma mensagem de boas-vindas
      JOptionPane.showMessageDialog(null, "Olá, " + nome + "!", "Saudação", JOptionPane.INFORMATION_MESSAGE);
   }
}
```

- O método showInputDialog exibe uma caixa de texto onde o usuário pode digitar algo. A resposta é capturada como uma **string**.
- O primeiro parâmetro null é para a janela pai, e o segundo é a **mensagem** pedindo a entrada.
- O retorno de showInputDialog é sempre uma **String**. Então, se for necessário, faça a conversão para outro tipo de dado (número, por exemplo).

3. Capturando Números com JOptionPane

O JOptionPane captura apenas **Strings**, mas podemos **converter** para tipos numéricos (como int ou double).

Exemplo: Caixa de Entrada de Número

```
import javax.swing.JOptionPane;

public class EntradaNumero {
   public static void main(String[] args) {
      // Captura a idade do usuário
      String idadeStr = JOptionPane.showInputDialog(null, "Qual a sua idade?", "Entrada de Número", JOptionPane.QUESTION_MESSAGE);

      // Converte a string para um inteiro
      int idade = Integer.parseInt(idadeStr);

      // Exibe a idade
      JOptionPane.showMessageDialog(null, "Você tem " + idade + " anos.", "Idade", JOptionPane.INFORMATION_MESSAGE);
   }
}
```

- A entrada capturada é sempre uma **String**, então a conversão com Integer.parseInt() (ou Double.parseDouble() para números decimais) é necessária.

4. Caixa de Confirmação (Sim ou Não)

Você pode **pedir uma confirmação** do usuário com opções como "Sim", "Não", "Cancelar". Isso é útil quando queremos saber se o usuário deseja continuar uma ação, como excluir um item.

Exemplo: Caixa de Confirmação

```
import javax.swing.JOptionPane;
```

```
public class Confirmacao {
   public static void main(String[] args) {
      // Caixa de confirmação com as opções "Sim" e "Não"
      int resposta = JOptionPane.showConfirmDialog(null, "Você deseja continuar?", "Confirmar", JOptionPane.YES_NO_OPTION);

      // Verifica a resposta do usuário
      if (resposta == JOptionPane.YES_OPTION) {
         JOptionPane.showMessageDialog(null, "Você escolheu continuar!", "Resposta", JOptionPane.INFORMATION_MESSAGE);
      } else {
         JOptionPane.showMessageDialog(null, "Você escolheu não continuar.", "Resposta", JOptionPane.INFORMATION_MESSAGE);
      }
   }
}
```

- JOptionPane.showConfirmDialog retorna um valor inteiro:
  - JOptionPane.YES_OPTION (valor 0) se o usuário escolher "Sim".
  - JOptionPane.NO_OPTION (valor 1) se o usuário escolher "Não".

5. Usando Caixas de Diálogo com Ícones Personalizados

Você pode adicionar ícones personalizados nas caixas de diálogo para torná-las mais informativas ou atraentes.

Exemplo: Caixa com Ícone de Erro

```
import javax.swing.JOptionPane;

public class CaixaErro {
   public static void main(String[] args) {
      // Exibe uma mensagem de erro com um ícone
```

```
        JOptionPane.showMessageDialog(null, "Algo deu errado!", "Erro", JOptionPane.ERROR_MESSAGE);
    }
}
```

Você também pode usar **ícones personalizados**:

```
import javax.swing.*;
import java.net.URL;

public class CaixaComIcone {
    public static void main(String[] args) {
        // Ícone personalizado
        URL url = CaixaComIcone.class.getResource("/caminho/para/o/icone.png");
        ImageIcon icon = new ImageIcon(url);

        // Caixa de mensagem com ícone personalizado
        JOptionPane.showMessageDialog(null, "Mensagem com ícone personalizado", "Mensagem", JOptionPane.INFORMATION_MESSAGE, icon);
    }
}
```

Resumo dos Métodos mais Usados do JOptionPane

Método	**Descrição**
showMessageDialog(Component, String, String, int)	Exibe uma mensagem simples (informação, erro, aviso, etc.)
showInputDialog(Component, String, String, int)	Exibe uma caixa de entrada de texto para o usuário

Método	Descrição
showConfirmDialog(Component, String, String, int)	Exibe uma caixa de confirmação (Sim, Não, Cancelar)
showOptionDialog(Component, Object, String, int, int, Icon, Object[], Object)	Caixa de diálogo com opções personalizadas

Desafios Práticos

1. **Desafio 1**: Crie um programa que pergunte o nome e a idade do usuário, e então exiba uma mensagem dizendo se ele pode votar (idade >= 16).
2. **Desafio 2**: Crie um programa que peça ao usuário para inserir dois números e, em seguida, calcule a soma e exiba o resultado em uma caixa de mensagem.
3. **Desafio 3**: Crie um programa que utilize uma caixa de confirmação para perguntar se o usuário deseja sair do programa, com as opções "Sim" e "Não".

Conclusão

- O **JOptionPane** é uma ferramenta poderosa para criar **interfaces gráficas simples** em Java, sem a complexidade de outras bibliotecas gráficas mais pesadas.
- Ele permite **mostrar mensagens**, **capturar entradas de texto e números**, **pedir confirmações**, e até exibir **ícones personalizados**.

Com isso, você pode criar **aplicações interativas** e **simples interfaces gráficas** em Java, tudo com facilidade e eficiência.

O que é o Swing?

Swing é uma **biblioteca gráfica** do Java que faz parte do pacote javax.swing. Ela foi criada para permitir a criação de **interfaces gráficas ricas**, com **componentes** como botões, campos de texto, listas, menus e muito mais.

Ao contrário de outras bibliotecas gráficas (como o **AWT**, que é mais simples e limitado), o Swing oferece:

- **Maior controle sobre a aparência**.
- **Componentes mais complexos** e **personalizáveis**.
- **Independência de plataforma** (funciona de maneira similar em sistemas operacionais diferentes).

1. Criação de uma Janela Simples com Swing

Para começar, vamos criar uma janela simples com um título e um botão, apenas para entender o básico.

Exemplo Básico:

java
CopiarEditar

```
import javax.swing.JFrame;  // Para criar a janela
import javax.swing.JButton; // Para criar botões

public class JanelaSimples {
    public static void main(String[] args) {
        // Criar a janela (JFrame)
        JFrame janela = new JFrame("Minha Primeira Janela Swing");

        // Definir o tamanho da janela
        janela.setSize(400, 300);

        // Criar um botão
        JButton botao = new JButton("Clique Aqui!");
```

```
        // Adicionar o botão à janela
        janela.add(botao);

        // Definir a operação de fechamento
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Tornar a janela visível
        janela.setVisible(true);
    }
}
```

Explicação do código:

1. **JFrame**: A classe JFrame cria uma janela onde podemos adicionar componentes como botões, textos, etc.
2. **JButton**: A classe JButton cria um botão na interface.
3. **janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)**: Isso garante que a aplicação será fechada quando o usuário clicar no "X" da janela.
4. **janela.setSize(400, 300)**: Define o tamanho da janela (400 pixels de largura e 300 pixels de altura).
5. **janela.setVisible(true)**: Torna a janela visível na tela.

2. Componentes Básicos do Swing

Agora, vamos aprender sobre alguns dos componentes mais comuns que você pode usar em uma interface gráfica Swing:

JButton (Botões)

java

CopiarEditar

```
JButton botao = new JButton("Clique Aqui!");
```

JLabel (Rótulos/Texto)

java

CopiarEditar

```
import javax.swing.JLabel;

JLabel label = new JLabel("Texto de Exemplo");
```

JTextField (Campo de Texto)

java

CopiarEditar

```
import javax.swing.JTextField;

JTextField campoTexto = new JTextField(20);  // 20 é o número de colunas
```

JPasswordField (Campo de Senha)

java

CopiarEditar

```
import javax.swing.JPasswordField;

JPasswordField campoSenha = new JPasswordField(20);
```

JTextArea (Área de Texto Multilinha)

java

CopiarEditar

```
import javax.swing.JTextArea;

JTextArea areaTexto = new JTextArea(5, 20);  // 5 linhas e 20 colunas
```

JComboBox (Caixa de Seleção - Drop Down)

java

CopiarEditar

```
import javax.swing.JComboBox;

String[] opcoes = { "Opção 1", "Opção 2", "Opção 3" };
JComboBox<String> comboBox = new JComboBox<>(opcoes);
```

3. Layouts no Swing

Os **Layouts** no Swing determinam a **disposição** dos componentes dentro de um contêiner (como uma janela ou painel).

Layouts mais comuns:

1. **FlowLayout** (padrão) – Organiza os componentes da esquerda para a direita, linha por linha.
2. **BorderLayout** – Divide o contêiner em cinco áreas (Norte, Sul, Leste, Oeste, Centro).
3. **GridLayout** – Organiza os componentes em uma grade (linhas e colunas).
4. **BoxLayout** – Organiza os componentes em uma linha ou coluna, com ou sem espaçamento.

Vamos usar o **FlowLayout** para nosso primeiro exemplo com componentes.

Exemplo com Layout:

java
CopiarEditar

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

public class ExemploLayout {
    public static void main(String[] args) {
        JFrame janela = new JFrame("Exemplo de Layout");

        // Usando FlowLayout
        janela.setLayout(new FlowLayout());

        // Criando componentes
        JLabel label = new JLabel("Digite seu nome:");
        JTextField campoTexto = new JTextField(20);
        JButton botao = new JButton("Enviar");

        // Adicionando componentes à janela
        janela.add(label);
        janela.add(campoTexto);
        janela.add(botao);
```

```
        // Definindo configurações da janela
        janela.setSize(300, 200);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
    }
}
```

Explicação:

- **FlowLayout** organiza os componentes de maneira sequencial, da esquerda para a direita.
- O **JLabel** exibe um texto, o **JTextField** permite ao usuário digitar um texto, e o **JButton** exibe um botão.
- O layout está aplicando uma disposição simples e organizada.

4. Eventos e Ações

No Swing, os **eventos** são ações que o usuário realiza (como clicar em um botão). Para responder a esses eventos, usamos **Listeners**.

Exemplo com Evento de Clique em Botão

java
CopiarEditar

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

public class ExemploEventos {
    public static void main(String[] args) {
        // Criando a janela
        JFrame janela = new JFrame("Exemplo de Evento");

        // Definindo o layout
        janela.setLayout(new FlowLayout());
```

```
        // Criando componentes
        JButton botao = new JButton("Clique Me");

        // Adicionando o evento de clique ao botão
        botao.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Código a ser executado quando o botão for clicado
                JOptionPane.showMessageDialog(janela, "Botão clicado!");
            }
        });

        // Adicionando o botão à janela
        janela.add(botao);

        // Definindo configurações da janela
        janela.setSize(300, 200);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
    }
}
```

Explicação:

- Usamos addActionListener para associar um **ActionListener** ao botão.
- Quando o botão é clicado, a ação **actionPerformed** é chamada, e o código dentro dessa função é executado (exibindo uma mensagem com JOptionPane).

5. Exemplo Completo de Aplicação Swing

Agora, vamos construir uma **aplicação completa** usando **Swing**. O programa terá:

- Uma janela com **campos de texto** para inserir o nome e a idade do usuário.
- Um **botão** que, ao ser clicado, exibe uma mensagem de boas-vindas personalizada.

Código Completo:

java
CopiarEditar

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

public class AplicacaoSwing {
    public static void main(String[] args) {
        // Criar a janela
        JFrame janela = new JFrame("Aplicação Swing");

        // Layout para organizar os componentes
        janela.setLayout(new GridLayout(3, 2));

        // Criar os componentes
        JLabel labelNome = new JLabel("Digite seu nome:");
        JTextField campoNome = new JTextField(20);

        JLabel labelIdade = new JLabel("Digite sua idade:");
        JTextField campoIdade = new JTextField(20);

        JButton botaoEnviar = new JButton("Enviar");

        // Adicionar componentes à janela
        janela.add(labelNome);
        janela.add(campoNome);
        janela.add(labelIdade);
        janela.add(campoIdade);
        janela.add(botaoEnviar);
```

```
        // Adicionar ação ao botão
        botaoEnviar.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Captura os valores dos campos de texto
                String nome = campoNome.getText();
                String idade = campoIdade.getText();

                // Exibe a mensagem de boas-vindas
                JOptionPane.showMessageDialog(janela, "Olá, " + nome + "! Você
tem " + idade + " anos.");
            }
        });

        // Configurações da janela
        janela.setSize(400, 200);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
    }
}
```

Explicação:

- Usamos o **GridLayout** para organizar os componentes em uma grade 3x2.
- O **botão** possui um **ActionListener** para capturar os dados dos campos de texto e exibir uma mensagem personalizada.