

Programação III: Interface Gráficas

Profª. Tainá Isabela

Java Swing

A construção de interfaces gráficas para programas desktop em Java se baseia em duas bibliotecas principais: a AWT (Abstract Window Toolkit) e a Swing.

- Import java.awt.*
- Import java.awt.event.*
- Import javax.swing.*

JFrame

É uma classe do pacote Swing que fornece todas as propriedades, métodos e eventos de que precisamos para construir janelas “padrão Windows”.

A classe JFrame provê um conjunto de métodos que permitem criar e configurar janelas. Abaixo são citados alguns deles:

- `JFrame()`: construtor padrão. Apenas cria uma nova janela.
- `JFrame(String t)`: cria uma janela atribuindo um título a mesma.
- `getTitle()`: obtém o título da janela.

JFrame

- `setTitle(String t)`: atribui um título à janela.
- `isResizable()`: verifica se a janela é redimensionável.
- `setResizable(boolean b)`: especifica se a janela é ou não redimensionável. Caso seja passado true como parâmetro, a janela será redimensionável. Caso o parâmetro passado seja false, a janela não será redimensionável.
- `setSize(int l, int a)`: define o tamanho da janela. Os parâmetros passados são a largura e a altura da janela.
- `setLocation(int x, int y)`: define a posição da janela na tela. O primeiro parâmetro a ser passado é a posição horizontal da janela a partir do lado esquerdo da tela. O segundo parâmetro define a posição vertical a partir da parte superior da tela

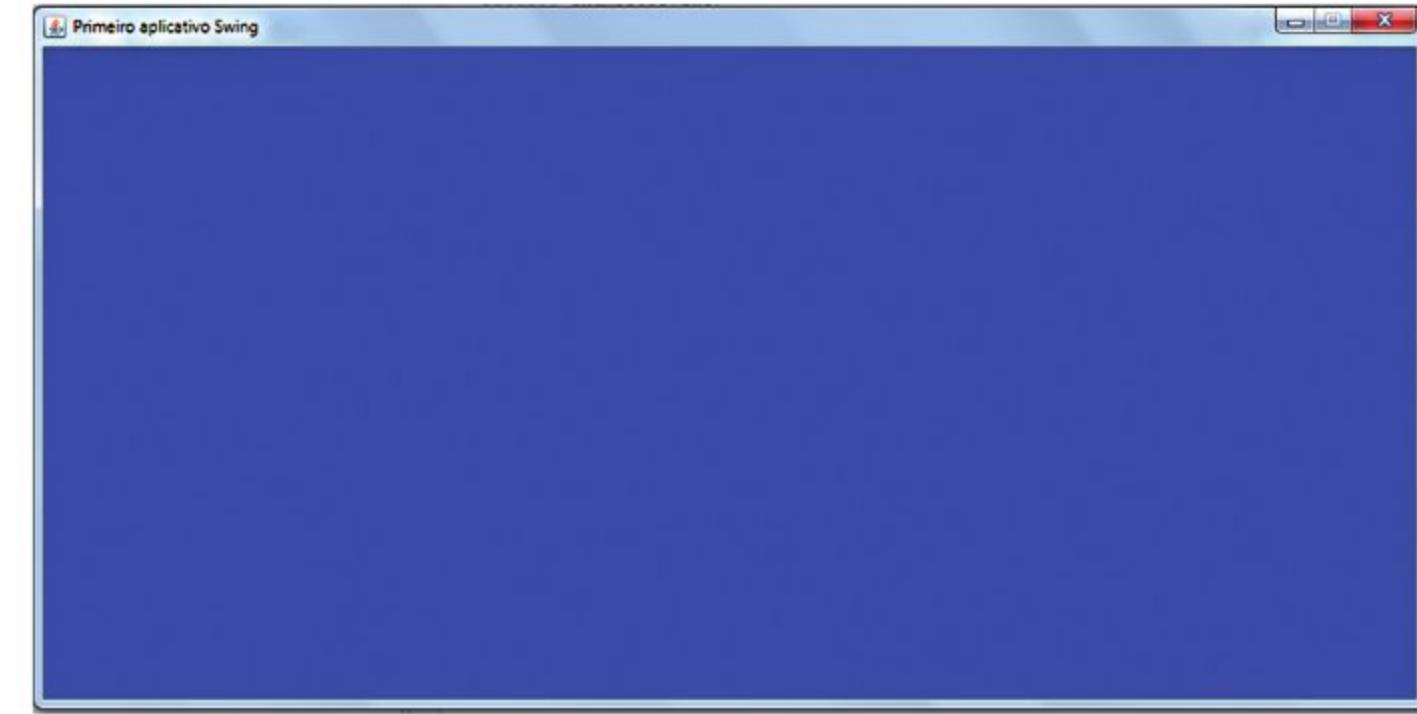
```
package exemplosSwing;

import java.awt.Color;
import javax.swing.JFrame;

public class PrimeiraJanela extends JFrame {

    public PrimeiraJanela() {
        //Título da Janela
        this.setTitle("Primeiro aplicativo Swing");
        //Dimensões da Janela
        this.setSize(1000, 500);
        //Posição:Canto esquerdo superior da tela
        this.setLocation(150, 50);
        //Impedir que a janela seja redimensionada
        this.setResizable(false);
        //Colocar cor de fundo azul na janela
        this.getContentPane().setBackground(Color.blue);
    }

    public static void main(String[] args) {
        // Criar uma instancia do tipo "PrimeiraJanela"
        PrimeiraJanela jan = new PrimeiraJanela();
        //Tornar a janela visivel
        jan.setVisible(true);
    }
}
```



JLabel e ImageIcon

A classe JLabel é utilizada para criar etiquetas (labels) de textos nas janelas. Ela permite o controle de propriedades do texto a ser utilizado, tais como: alinhamento, tipo de letra, tamanho, cor, etc.

- `JLabel()`: construtor padrão.
- `JLabel(String)`: recebe como parâmetro uma String que será o texto apresentado pelo Label.
- `JLabel(String, int)`: além do texto a ser apresentado, recebe como parâmetro um inteiro que representa o tipo de alinhamento a ser utilizado.

JLabel e ImageIcon

- `JLabel(String, Image)`: além do texto a ser apresentado, recebe como parâmetro um `image` que será o ícone a ser exibido.
- `JLabel(String, Image, int)`: recebe como parâmetros o texto a ser apresentado, o ícone a ser exibido e o tipo de alinhamento a ser utilizado.

Outros dois métodos essenciais para o uso de `JLabels` são o método `getText()` que retorna o texto do Label e o `setText(String)` que especifica (altera) o texto a ser apresentado pelo Label.

```
package javaswing_1;
import java.awt.*;
import javax.swing.*;
public class UsaJLabel_ImageIcon extends JFrame {
    private JLabel label1, label2;
    private ImageIcon icone = new ImageIcon("C:/Users/Giovany/Desktop/AppInstalled.gif");
    public UsaJLabel_ImageIcon() {
        this.setTitle("Labels");
        this.setSize(350, 120);
        this.setLocation(50, 50);
        this.getContentPane().setBackground(new Color(220, 220, 0));

        this.label1 = new JLabel(" Esquerda ", icone, JLabel.LEFT);
        this.label2 = new JLabel(" Direita ", JLabel.RIGHT);

        this.getContentPane().setLayout(new GridLayout(2, 1));
        this.getContentPane().add(this.label1);
        this.getContentPane().add(this.label2);
    }
    public static void main(String[] args) {
        JFrame janela = new UsaJLabel_ImageIcon();
        janela.setUndecorated(true);
        janela.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
    }
}
```



JOptionPane

A classe JOptionPane é utilizada para gerar caixas de diálogo.
Ela nos permite criar vários tipos de caixas de diálogo, a saber:
MessageDialog, ConfirmDialog, InputDialog e OptionDialog.

MessageDialog

Uma MessageDialog é uma caixa de diálogo que apresenta uma mensagem.

ConfirmDialog

Uma ConfirmDialog é uma caixa de diálogo que apresenta uma mensagem e possibilita ao usuário responder uma pergunta.

InputDialog

Uma InputDialog é uma caixa de diálogo que apresenta uma mensagem e possibilita que o usuário digite um texto.

OptionDialog

Uma OptionDialog é uma caixa de diálogo que possibilita a exibição de várias opções para escolha do usuário. Ela apresenta vários botões para o usuário e retorna um número inteiro indicando em qual dos botões o usuário clicou

```
package javaswing_3;
import javax.swing.*;
public class Usa JOptionPane {
    public static void main(String[] args)
    {
        // TODO code application logic here
        String s = JOptionPane.showInputDialog(null, "Digite seu login", "Login no sistema", JOptionPane.QUESTION_MESSAGE);
        if (s == null) return;
        if (JOptionPane.showConfirmDialog(null, "Confirma login ?", "Caixa de confirmação", JOptionPane.OK_CANCEL_OPTION,
                JOptionPane.QUESTION_MESSAGE) == 0)
        {
            JOptionPane.showMessageDialog(null, s, "Login confirmado", JOptionPane.INFORMATION_MESSAGE);
        }
        else
        {
            JOptionPane.showMessageDialog(null, s, "Login NÃO CONFIRMADO", JOptionPane.WARNING_MESSAGE);
            String[] nomes = { "João", "Maria", "Pedro", "Janaina" };
            int resp = JOptionPane.showOptionDialog(null, "Escolha um login padrão", "Login no sistema", 0,
                    JOptionPane.INFORMATION_MESSAGE, null, nomes, nomes[1]);
            if (resp == 1)
            {
                JOptionPane.showMessageDialog(null, "Login " + s + " inválido !!!", "Login confirmado",
                        JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}
```



Tratamento de eventos e JButton

Quando desenvolvemos aplicativos com interface gráfica precisamos programar as respostas que o sistema dará às interações do usuário com a interface gráfica.

A essa programação do comportamento do sistema de acordo com as ações do usuário dá-se o nome de **tratamento de eventos**.

JButton

Dentre os métodos mais utilizados da classe JButton estão:

- JButton(): construtor que cria um botão sem texto.
- JButton(String): construtor que cria um botão exibindo o texto passado como parâmetro.
- JButton(String, Image): construtor que cria um botão com texto e imagem.

JButton

- `getLabel()`: obtém o texto apresentado pelo botão.
- `setLabel(String)`: define o texto a ser apresentado pelo botão.
- `setEnabled(boolean)`: define se o botão está habilitado (true) ou desabilitado (false).
- `setHorizontalTextPosition()`: define o alinhamento horizontal, que pode ser: LEFT (esquerda) ou RIGHT (direita)

JButton

- `setVerticalTextPosition()`: define o alinhamento vertical que pode ser: TOP (por cima) ou BOTTOM (por baixo).
- `setMnemonic(int)`: define o atalho (combinação de teclas) para acionar o botão (equivalente ao clique sobre o botão).
- `setToolTipText(String)`: possibilita colocar uma mensagem de ajuda no botão.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class UsaJButton extends JFrame implements ActionListener
{
    private JButton b1;
    public UsaJButton()
    {
        // Criando e setando atributos de b1
        b1 = new JButton("Mensagem");
        b1.setHorizontalTextPosition(AbstractButton.RIGHT);
        b1.setBackground(new Color(100, 180, 180));
        b1.setForeground(Color.black);
        b1.setFont(new Font("Scripts", Font.BOLD, 20));
        b1.setEnabled(true);
        b1.setToolTipText("Clique aqui para ver a mensagem");
        b1.setMnemonic(KeyEvent.VK_B); // Alt + B = Clique do mouse
        b1.addActionListener(this);

        this.setTitle("Inserindo botoes na janela");
        this.setSize(350, 100);
        this.setLocation(50, 50);
        this.getContentPane().setBackground(new Color(180, 180, 180));
        this.getContentPane().setLayout(new FlowLayout());
        this.getContentPane().add(this.b1);
    }
}
```

```
// Método declarada na classe ActionListener
public void actionPerformed(ActionEvent e)
{
    if ( e.getSource() == b1 )
        JOptionPane.showMessageDialog(null, "Mensagem", "Botão clicado",
        JOptionPane.INFORMATION_MESSAGE);
}

public static void main(String[] args)
{
    JFrame janela = new UsaJButton();
    janela.setUndecorated(true);
    janela.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
    janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    janela.setVisible(true);
}
}
```

JTextField

A classe JTextField é utilizada para criar caixas de texto. Normalmente, utiliza-se a classe JLabel para apresentar um texto fixo e a classe JTextField para campos de texto a serem digitados pelos usuários.

- JTextField(): construtor padrão utilizado para criar uma caixa de texto vazia.
- JTextField(String): construtor utilizado para criar uma caixa de texto com a String fornecida.
- JTextField(int): construtor utilizado para criar uma caixa de texto com a quantidade de colunas especificada.

JTextField

- JTextField(String, int): construtor utilizado para criar uma caixa de texto com uma determinada String e com a quantidade de colunas especificada.
- getText(): obtém o texto do objeto.
- setText(String): atribui uma String ao objeto.
- getSelectedText(): obtém o texto selecionado no objeto.
- isEditable(): verifica se o componente é editável ou não e retornando um boolean (true ou false)
- setEditable(boolean): especifica se o componente é editável ou não.

JPasswordField

Há uma classe especial para criar caixas de texto próprias para campos de senha: JPasswordField. O funcionamento dessa classe é bastante semelhante ao de JTextField. A diferença básica é que os caracteres digitados são substituídos por outros para ocultar a senha. Assim, os principais métodos de JPasswordField são **idênticos** aos de JTextField.

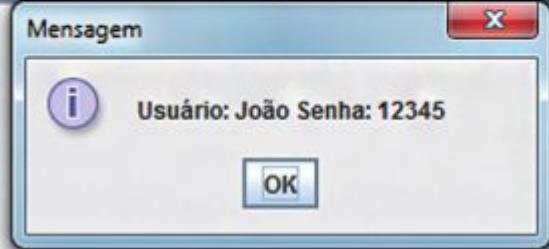
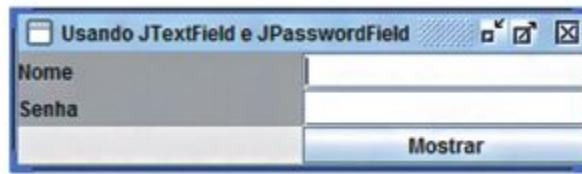
- `setEchoChar(char)`: determina o caractere que será utilizado para esconder a senha.
- `getPassword()`: gera um vetor de char com a senha digitada.
- `getText()`: método obsoleto que retorna a String do objeto.

```
package javaswing_1;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class UsaJTextFieldJPasswordField   extends JFrame implements ActionListener{
    private JLabel l1, l2;
    private JTextField t1;
    private JPasswordField p2;
    private JButton b1;
    private JPanel painel;
    public UsaJTextFieldJPasswordField() {
        this.l1 = new JLabel("Nome ", JLabel.LEFT);
        this.t1 = new JTextField();
        this.l2 = new JLabel("Senha ", JLabel.LEFT);
        this.p2 = new JPasswordField();
        this.p2.setEchoChar('*');
        this.painel = new JPanel();
        this.b1 = new JButton("Mostrar");

        this.setTitle("Usando JTextField e JPasswordField");
        this.setSize(350, 100);
        this.setLocation(50, 50);
        this.getContentPane().setBackground(new Color(180, 180, 180));

        this.montarLayout();
    }
}
```

```
private void montarLayout() {
    this.getContentPane().setLayout(new GridLayout(3, 2));
    this.getContentPane().add(this.l1);
    this.getContentPane().add(this.t1);
    this.getContentPane().add(this.l2);
    this.getContentPane().add(this.p2);
    this.getContentPane().add(this.painel);
    this.getContentPane().add(this.b1);
    this.b1.addActionListener(this);
}
public void actionPerformed(ActionEvent e) {
    if ( e.getSource() == this.b1 ) {
        String s = "Usuário: " + this.t1.getText() + " Senha: " + new String(this.p2.getPassword());
        JOptionPane.showMessageDialog(null, s, "Mensagem",
        JOptionPane.INFORMATION_MESSAGE);
    }
}
public static void main(String[] args) {
    JFrame janela = new UsaJTextFieldJPasswordField();
    janela.setUndecorated(true);
    janela.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
    janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    janela.setVisible(true);
}
}
```



Padrões de layout

Os padrões de layout definem como os objetos serão dispostos na janela. O uso de padrões de layout deixa nossas janelas mais genéricas e adaptáveis a modificações.

- FlowLayout
- GridLayout

FlowLayout

O padrão de layout FlowLayout insere os objetos, um após o outro, linha a linha. Assim, quando não é mais possível inserir objetos numa linha, é criada uma nova linha e novos objetos podem ser inseridos.

Esse padrão de layout assemelha-se a um editor de textos, pois quando não existe mais espaço numa dada linha é criada uma nova linha para inserção de mais conteúdos.

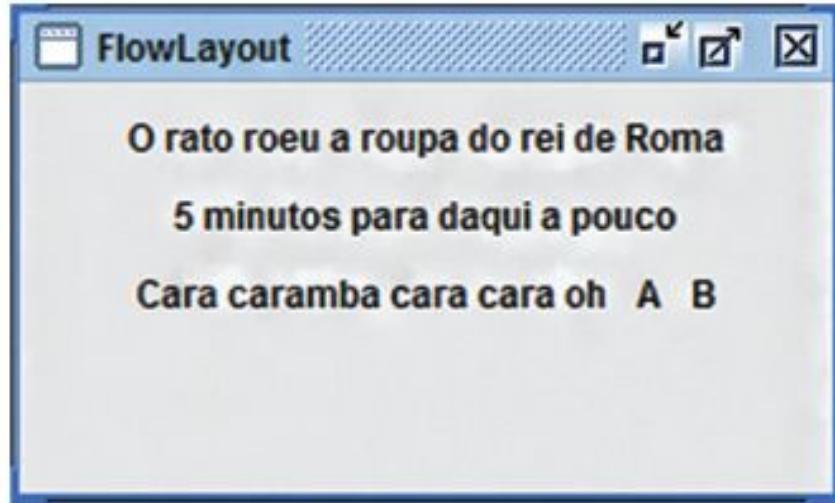
```
package padroesLayout;

import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JRootPane;

public class ExemploFlowLayout extends JFrame {

    public ExemploFlowLayout(){
        FlowLayout layout = new FlowLayout(FlowLayout.CENTER, 10, 10);
        this.setLayout(layout);
        this.add(new JLabel("O rato roeu a roupa do rei de Roma"));
        this.add(new JLabel("5 minutos para daqui a pouco"));
        this.add(new JLabel("Cara caramba cara cara oh"));
        this.add(new JLabel("A"));
        this.add(new JLabel("B"));
    }

    public static void main(String[] args) {
        JFrame janela = new ExemploFlowLayout();
        janela.setBounds(250, 50, 280, 170);
        janela.setTitle("FlowLayout");
        janela.setUndecorated(true);
        janela.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
    }
}
```



GridLayout

O padrão de layout GridLayout organiza os objetos como uma tabela, com células de objetos de **mesmo tamanho**.

É um layout flexível, pois uma vez redimensionada a janela ele ajusta automaticamente os objetos de forma que o padrão se mantenha, ou seja, que cada objeto de cada célula seja apresentado com o mesmo tamanho.

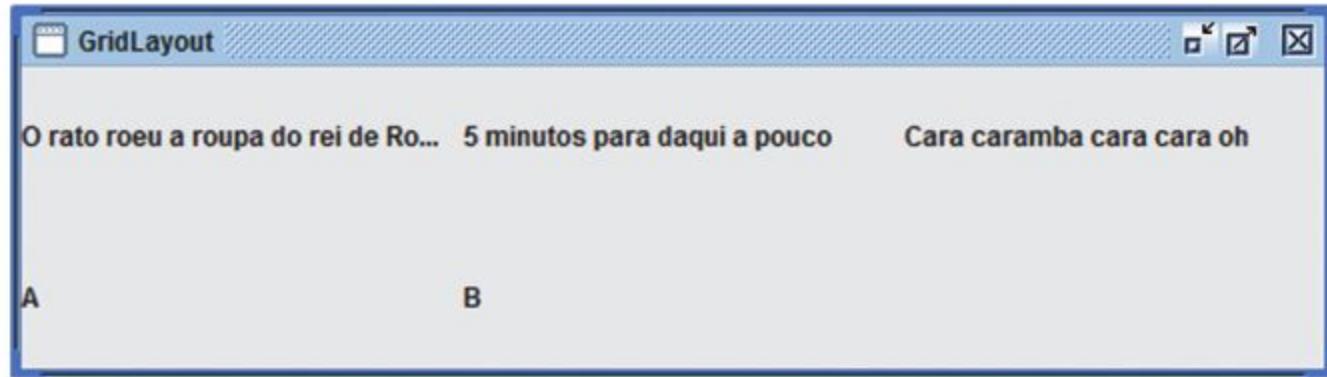
```
package padroesLayout;

import java.awt.GridLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JRootPane;

public class ExemploGridLayout extends JFrame {

    public ExemploGridLayout(){
        GridLayout layout = new GridLayout(2, 3, 10, 10);
        this.setLayout(layout);
        this.add(new JLabel("O rato roeu a roupa do rei de Roma"));
        this.add(new JLabel("5 minutos para daqui a pouco"));
        this.add(new JLabel("Cara caramba cara cara oh"));
        this.add(new JLabel("A"));
        this.add(new JLabel("B"));
    }

    public static void main(String[] args) {
        JFrame janela = new ExemploGridLayout();
        janela.setBounds(250, 50, 600, 170);
        janela.setTitle("GridLayout");
        janela.setUndecorated(true);
        janela.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setVisible(true);
    }
}
```



JComboBox e tratamento de eventos

A classe JComboBox é utilizada para criar caixas que permitem que o usuário selecione apenas um item da sua lista de opções.

Para criar um objeto do tipo JComboBox, é necessário passar um vetor de Strings que indicará as opções de seleção a serem exibidas na caixa.

- `getSelectedIndex`: retorna o índice do item selecionado no combobox. É importante notar que o índice do primeiro elemento é 0.
- `removeItemAt`: dado um índice, elimina do combobox o item que está nessa posição.

JComboBox e tratamento de eventos

- removeAllItems: remove todos os itens do combobox.
- addItem: dado um novo item, insere-o no combobox.
- getSelectedItem: retorna o item selecionado.
- getItemCount: retorna a quantidade de itens do combobox.

```
package javaswing2;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class UsaJComboBox extends JFrame implements ActionListener, ItemListener {
    JLabel label1;
    JTextField tadicinar, t1, t2;
    JComboBox combo;
    JButton novoItem;

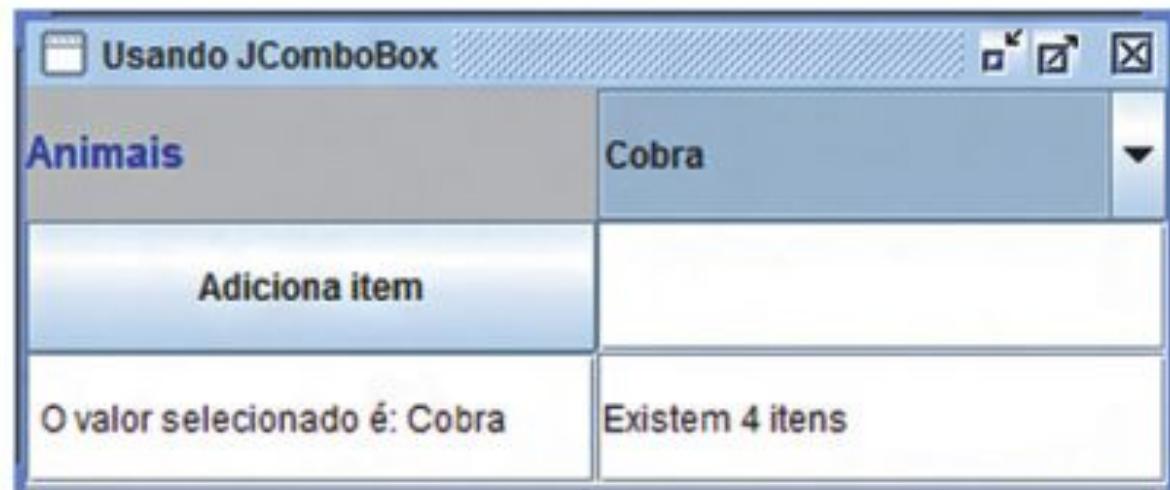
    private void criarComponentesJanela()
    {
        this.label1 = new JLabel("Animais");
        this.label1.setForeground(Color.blue);
        this.label1.setFont(new Font("Arial", Font.BOLD, 15));
        this.novoItem = new JButton("Adiciona item");
        this.t1 = new JTextField();
        this.t2 = new JTextField();
        this.tadicinar = new JTextField();
        String[] animais = { "Leão", "Elefante", "Cobra", "Jaboti" };
        combo = new JComboBox(animais);
    }
}
```

```
public void actionPerformed(ActionEvent e) {
    //adiciona item
    if ( e.getSource() == this.novoItem && this.tadicionar.getText().length() != 0 )
    {
        this.combo.addItem(tadicionar.getText());
        this.tadicionar.setText("");
    }
}
public void itemStateChanged(ItemEvent e) {
    this.t1.setText(" O valor selecionado é: " + this.combo.getSelectedItem().toString());
    this.t2.setText("Existem " + String.valueOf(this.combo.getItemCount()) + " itens");
}
private void setarAtributosJanela()
{
    this.setTitle("Usando JComboBox");
    this.setBounds(50, 50, 400, 170);
    this.getContentPane().setBackground(new Color(190, 190, 190));
    this.getContentPane().setLayout(new GridLayout(3, 2));
    this.getContentPane().add(this.label1);
    this.getContentPane().add(this.combo);
    this.getContentPane().add(this.novoItem);
    this.getContentPane().add(this.tadicionar);
    this.getContentPane().add(this.t1);
    this.getContentPane().add(this.t2);
}
```

```
private void adicionarListeners()
{
    this.novoItem.addActionListener(this);
    this.combo.addItemListener(this);
}

public UsaJComboBox()
{
    this.criarComponentesJanela();
    this.setarAtributosJanela();
    this.adicionarListeners();
}

public static void main(String[] args) {
    JFrame janela = new UsaJComboBox();
    janela.setUndecorated(true);
    janela.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
    janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    janela.setVisible(true);
}
```



JCheckBox

As caixas de opção são criadas a partir da classe JCheckBox e permitem representar uma opção que está ativada (true) ou desativada (false).

Exemplo: Se o checkbox estiver selecionado, o texto será apresentado em negrito e, caso contrário, será apresentado em fonte plana. Para verificar se um objeto JCheckBox está selecionado, utilizamos o método isSelected().

```
package javaswing2;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class UsaJCheckBox extends JFrame implements ItemListener {
    JLabel label1;
    JCheckBox c1;

    private void criarComponentesJanela()
    {
        this.label1 = new JLabel("O rato roeu");
        this.label1.setFont(new Font("Arial", Font.PLAIN, 20));
        this.c1 = new JCheckBox("Negrito");
    }

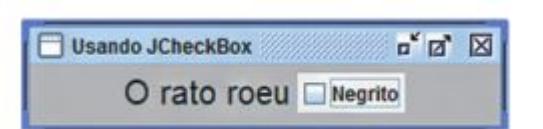
    private void setarAtributosJanela()
    {
        this.setTitle("Usando JCheckBox");
        this.setBounds(250, 50, 300, 70);
        this.getContentPane().setBackground(new Color(190, 190, 190));
        this.getContentPane().setLayout(new FlowLayout(FlowLayout.CENTER));
        this.getContentPane().add(this.label1);
        this.getContentPane().add(this.c1);
    }
}
```

```
private void adicionarActionListeners()
{
    this.c1.addItemListener(this);
}

public UsaJCheckBox() {
    this.criarComponentesJanela();
    this.setarAtributosJanela();
    this.adicionarActionListeners();
}

public void itemStateChanged(ItemEvent e)
{
    if ( e.getSource() == this.c1)
    {
        if ( this.c1.isSelected() )
            this.label1.setFont(new Font("Arial", Font.BOLD, 20));
        else
            this.label1.setFont(new Font("Arial", Font.PLAIN, 20));
    }
}

public static void main(String[] args) {
    JFrame janela = new UsaJCheckBox();
    janela.setUndecorated(true);
    janela.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
    janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    janela.setVisible(true);
}
```



JRadioButton e ButtonGroup

A classe JRadioButton é utilizada para criar “botões de rádio”. Esses botões são graficamente muito semelhantes aos checkbox.

A diferença entre radiobutton e checkbox está no fato que os primeiros são usados em conjuntos de forma que apenas um elemento do conjunto pode estar selecionado em um dado momento.

Mas, para haver esse controle, os JRadioButton devem estar em um grupo representado pela classe ButtonGroup.

Um objeto da classe ButtonGroup não é visual, ou seja, não tem impacto na interface gráfica, mas sem ele perdemos a garantia de que apenas um botão de rádio está selecionado.

JRadioButton e ButtonGroup

- setMnemonic: permite que uma combinação de teclas tenha o mesmo efeito do clique sobre um objeto JRadioButton.
- isSelected: determina se um botão de rádio está selecionado.
- setSelected: recebe um parâmetro booleano (true ou false) que faz com que o botão de rádio seja selecionado ou não.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class UsaJRadioButton extends JFrame implements ItemListener {
    JLabel label1, label2;
    JTextField t1, t2, t3;
    JRadioButton radio1, radio2, radio3;
    ButtonGroup radiogroup;
    private void criarComponentesJanela()
    {
        this.label1 = new JLabel("Números");
        this.label2 = new JLabel("Resultado: ");
        this.t1 = new JTextField(5);
        this.t2 = new JTextField(5);
        this.t3 = new JTextField(5);
        this.t3.setEditable(false);
        this.radio1 = new JRadioButton("+");
        this.radio2 = new JRadioButton("-");
        this.radio3 = new JRadioButton("*");
        this.radiogroup = new ButtonGroup();
    }
}
```

```
private void setarAtributosButtonGroup()
{
    this.radiogroup.add(this.radio1);
    this.radiogroup.add(this.radio2);
    this.radiogroup.add(this.radio3);

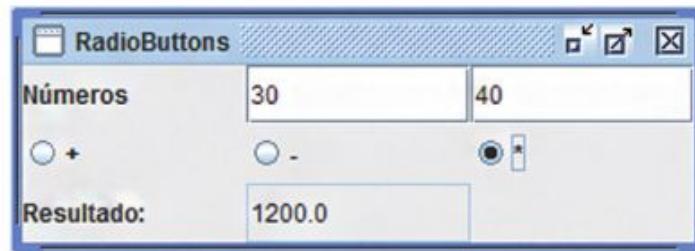
}

private void setarAtributosJanela()
{
    this.setTitle("RadioButtons");
    this.setBounds(230, 50, 340, 120);
    this.getContentPane().setLayout(new GridLayout(3,3));
    this.getContentPane().add(this.label1);
    this.getContentPane().add(this.t1);
    this.getContentPane().add(this.t2);
    this.getContentPane().add(this.radio1);
    this.getContentPane().add(this.radio2);
    this.getContentPane().add(this.radio3);
    this.getContentPane().add(this.label2);
    this.getContentPane().add(this.t3);
    this.getContentPane().add(new JPanel());
}
```

```

private void adicionarItemListeners() {
    this.radio1.addItemListener(this);
    this.radio2.addItemListener(this);
    this.radio3.addItemListener(this);
}
public UsaJRadioButton() {
    this.criarComponentesJanela();
    this.setarAtributosJanela();
    this.setarAtributosButtonGroup();
    this.adicionarItemListeners();
}
public void itemStateChanged(ItemEvent e) {
    try {
        float n1 = Float.parseFloat(t1.getText());
        float n2 = Float.parseFloat(t2.getText());
        float result = 0;
        if (e.getSource() == this.radio1) result = n1 + n2;
        if (e.getSource() == this.radio2) result = n1 - n2;
        if (e.getSource() == this.radio3) result = n1 * n2;
        t3.setText(" " + result);
    } catch (NumberFormatException erro) { this.t3.setText("Erro"); }
}
public static void main(String[] args) {
    JFrame janela = new UsaJRadioButton();
    janela.setUndecorated(true);
    janela.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
    janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    janela.setVisible(true);
}
}

```



JMenuBar, JMenu e JMenuItem

Para que nossa janela conte com uma barra de menu, é necessário, inicialmente, instanciar uma barra de menus, ou seja, uma JMenuBar e associá-la à janela.

Para acrescentar um menus na barra de Menu, precisamos instanciar um objeto da classe JMenu e depois adicioná-lo à barra de menu.

Para acrescentar itens a um menu, precisamos instanciar objetos da classe JMenuItem e adicioná-los ao menu.

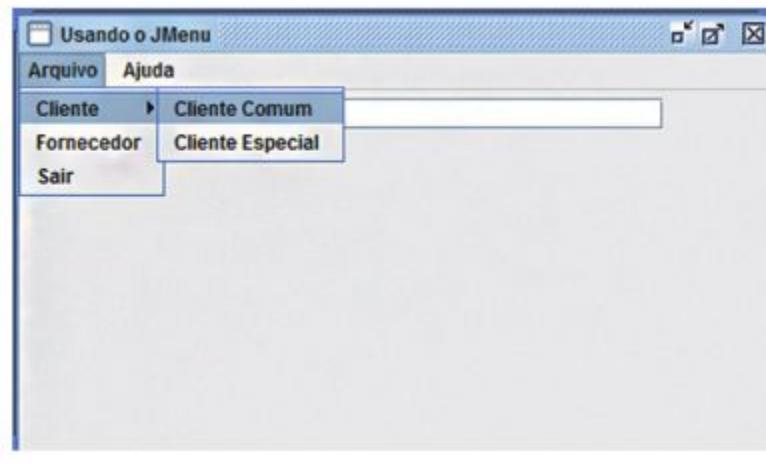
```
package javaswing2;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class UsaJMenu extends JFrame implements ActionListener {

    JMenuBar menuBar;
    JTextField t1;
    JMenu menuArquivo, menuCliente;
    JMenuItem miAjuda, miClienteEspecial, miClienteComum, miFornecedor, miSair;

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == miClienteEspecial) {
            t1.setText("Escolhido o item Cliente Especial");
        } else if (e.getSource() == miClienteComum) {
            t1.setText("Escolhido o item Cliente Comum");
        } else if (e.getSource() == miFornecedor) {
            t1.setText("Escolhido o item Fornecedor");
        } else if (e.getSource() == miAjuda) {
            t1.setText("Escolhido o menu Ajuda");
        } else if (e.getSource() == miSair) {
            System.exit(0);
        } else {
            t1.setText("Item escolhido inválido");
        }
    }
}
```

```
public UsaJMenu() {
    this.setBounds(150, 50, 480, 280);
    this.setTitle("Usando o JMenu");
    this.getContentPane().setLayout(new FlowLayout(FlowLayout.CENTER));
    this.t1 = new JTextField(30);
    this.getContentPane().add(t1);
    this.menuBar = new JMenuBar();
    this.setJMenuBar(this.menuBar);
    this.menuArquivo = new JMenuItem("Arquivo");
    this.menuCliente = new JMenuItem("Cliente");
    this.miClienteComum = new JMenuItem("Cliente Comum");
    this.miClienteEspecial = new JMenuItem("Cliente Especial");
    this.menuArquivo.add(menuCliente);
    this.menuCliente.add(this.miClienteComum);
    this.menuCliente.add(this.miClienteEspecial);
    this.miFornecedor = new JMenuItem("Fornecedor");
    this.menuArquivo.add(this.miFornecedor);
    this.miSair = new JMenuItem("Sair");
    this.menuArquivo.add(this.miSair);
    this.miAjuda = new JMenuItem("Ajuda");
    this.menuBar.add(this.menuArquivo);
    this.menuBar.add(this.miAjuda);
    this.miAjuda.addActionListener(this);
    this.miClienteComum.addActionListener(this);
    this.miClienteEspecial.addActionListener(this);
    this.miFornecedor.addActionListener(this);
    this.miSair.addActionListener(this);
}
```

```
public static void main(String[] args) {
    JFrame janela = new UsaJMenu();
    janela.setUndecorated(true);
    janela.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
    janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    janela.setVisible(true);
}
}
```



Momento Questionário



Atividades

1. Crie uma janela (JFrame) com: Um título personalizado. Um JLabel exibindo uma mensagem de boas-vindas. Um JButton com texto “Fechar”. Quando o botão for clicado, a janela deve se fechar.
2. Desenvolva uma janela contendo: Um JLabel e um JTextField para o nome de usuário. Um JLabel e um JPasswordField para senha. Um JButton “Entrar”. Ao clicar em “Entrar”, exiba uma JOptionPane com uma mensagem de boas-vindas que inclua o nome digitado.
3. Implemente uma interface com: Dois campos JTextField para números.
 - a. Um JComboBox com as opções “Somar”, “Subtrair”, “Multiplicar” e “Dividir”.
 - b. Um JCheckBox para escolher se o resultado deve aparecer em uma JOptionPane. Um botão “Calcular”.
 - c. Ao clicar, mostre o resultado conforme a operação escolhida e a opção do checkbox.

ETAPA 4 - INTERFACES

Crie uma janela principal com:

- Campos para título e descrição;
- Campos específicos para tipo selecionado (Livro ou Filme)
- JComboBox para escolher o tipo
- Botão “Adicionar”
- Botão “Listar”
- Campo de busca + botão “Filtrar”
- **JTextArea** para exibir resultados

Ao adicionar, valide os campos e exiba mensagens de erro com JOptionPane em caso de exceções.

Personalize a GUI usando a **cor exclusiva do aluno** para:

- Fundo de painel lateral ou botões principais
- Título ou barra superior

ETAPA 4 – INTERFACES

Adicione dois botões na GUI:

- “Exportar Dados” → grava os itens em um arquivo escolhido com JFileChooser
- “Importar Dados” → lê e carrega os itens do arquivo