JScrollBar

O **JScrollBar** permite que o usuário selecione graficamente um valor deslizando um botão dentro de um intervalo limitado.

Para trabalhar com eles, você deve utilizar um `AdjustmentListener` através do método `JScrollbar.addAdjustmentListener()`. Quando ocorrer algum evento de ajuste, o método ouvinte será chamado.

CONSTRUTOR

```
JScrollbar ();
```

Cria uma instância do `JScrollBar` com intervalo de 0 - 100, valor inicial zero e com a orientação vertical.

```
JScrollbar (int orientação);
```

Cria uma instância do `JScrollBar` com intervalo de 0 - 100, valor inicial zero e com a orientação indicada.

PRINCIPAIS MÉTODOS

- Armazenando o valor:

```
int valor = barra.getValue();
```

- Configurações:

```
barra. setBackground (Color BackgroundColor);
barra. setMaximun (int valor);
barra. setMinimun (int valor);
barra. setValue (int valor);
```

EXEMPLO

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

public class Gui extends JFrame implements AdjustmentListener {

  JScrollBar HSelector;
  JScrollBar VSelector;
```

```
  JLabel Report;

  public Gui () {
    setSize(400,400);
    HSelector = new JScrollBar ();
    VSelector = new JScrollBar ();
    Report = new JLabel ();

    setLayout (new BorderLayout ());
    add (Report, BorderLayout.SOUTH);
    add (HSelector, BorderLayout.NORTH);
    add (VSelector,BorderLayout.EAST);

    HSelector.setMaximum (100);
    HSelector.addAdjustmentListener (this);
    VSelector.setMaximum (100);
    VSelector.addAdjustmentListener (this);
    HSelector.setOrientation (JScrollBar.HORIZONTAL);
    Report.setHorizontalAlignment (JTextField.CENTER);
    Report.setText ("H = " + HSelector.getValue() +
                    ", V = " + VSelector.getValue());
  }

  public void adjustmentValueChanged(AdjustmentEvent e) {
    Report.setText ("H = " + HSelector.getValue () +
                    ", V = " + VSelector.getValue ());
  }

  public static void main (String[] args){
        Gui tela = new Gui();
        tela.setVisible(true);
  }
}
```

JSlider

O **JSlider** permite que o usuário selecione graficamente um valor, através do deslizamento de um botão dentro de um intervalo limitado. As marcas da escala (principais e menores) podem ser mostradas, ambas com intervalos variáveis.

CONSTRUTOR

`JSlider ()`
Cria uma instância do `JSlider` com intervalo entre 0-100, valor inicial zero e orientação horizontal

`JSlider (int Orientation)`
Cria uma instância do `JSlider` com intervalo entre 0-100, valor inicial zero e com a orientação especificada

`JSlider (int Min, int Max)`
Cria uma instância do `JSlider` com intervalo entre Min-Max, valor inicial (Min+Max)/2 e orientação horizontal

`JSlider (int Orientation, int Min, int Max)`
Cria uma instância do `JSlider` com intervalo entre Min-Max, valor inicial (Min+Max)/2 e com a orientação especificada

`JSlider (int Orientation, int Min, int Max, int Value)`
Cria uma instância do `JSlider` com intervalo entre Min-Max, valor inicial indicado e com a orientação especificada

PRINCIPAIS MÉTODOS

- Armazenando o valor:

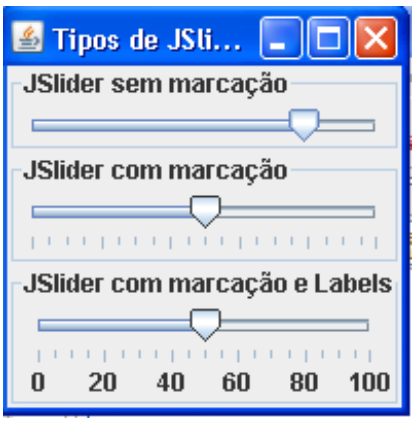
```
int valor = barra.getValue();
```

- Configurações:

```
barra.setBackground (Color BackgroundColor);
barra.setInverted (boolean Inverted);
barra.setMajorTickSpacing (int Delta);
barra.setMaximum (int Max);
barra.setMinorTickSpacing (int Delta);
barra.setMinimum (int Min);
barra.setValue (int Value);
```

EXEMPLO

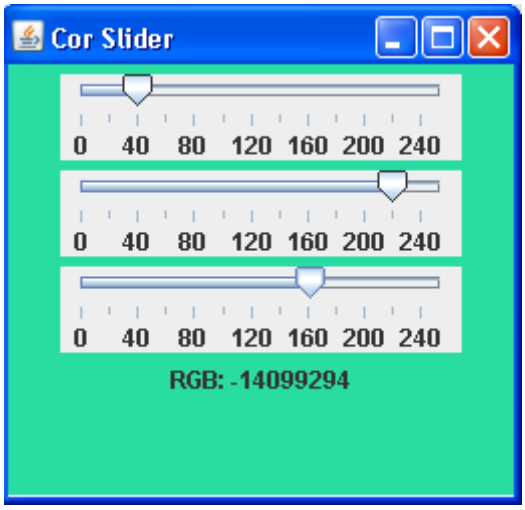
```
import java.awt.*;
import javax.swing.*;

public class Gui extends JFrame {

  public Gui() {
    super("Tipos de JSlider");
    setSize(200,200);
    Container content = getContentPane();
    content.setBackground(Color.white);

    JSlider slider1 = new JSlider();
    slider1.setBorder(BorderFactory.createTitledBorder("JSlider
sem marcação"));
    content.add(slider1, BorderLayout.NORTH);

    JSlider slider2 = new JSlider();
    slider2.setBorder(BorderFactory.createTitledBorder("JSlider
com marcação"));
    slider2.setMajorTickSpacing(20);
    slider2.setMinorTickSpacing(5);
    slider2.setPaintTicks(true);
    content.add(slider2, BorderLayout.CENTER);

    JSlider slider3 = new JSlider();
    slider3.setBorder(BorderFactory.createTitledBorder("JSlider
com marcação e Labels"));
    slider3.setMajorTickSpacing(20);
    slider3.setMinorTickSpacing(5);
    slider3.setPaintTicks(true);
    slider3.setPaintLabels(true);
    content.add(slider3, BorderLayout.SOUTH);
  }

  public static void main(String[] args) {
        Gui tela = new Gui();
        tela.setVisible(true);
  }
}
```



EXEMPLO

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

public class Gui extends JFrame implements ChangeListener {

	private JLabel informa;
	private JPanel mudaCor;
	private JSlider  sRed, sGreen, sBlue;

	public Gui() {
		super("Cor Slider");
		mudaCor = new JPanel(null);
		informa = new JLabel();
		mudaCor.setBackground(Color.white);
		//Cria e ajusta as escalas dos JSliders
		sRed = new JSlider(JSlider.HORIZONTAL, 0, 255, 0);
		sBlue = new JSlider(JSlider.HORIZONTAL, 0, 255, 0);
		sGreen = new JSlider(JSlider.HORIZONTAL, 0, 255, 0);
		sRed.setMinorTickSpacing(20);
		sBlue.setMinorTickSpacing(20);
		sGreen.setMinorTickSpacing(20);
		sRed.setMajorTickSpacing(40);
		sBlue.setMajorTickSpacing(40);
		sGreen.setMajorTickSpacing(40);
		sRed.setPaintTicks(true);
		sGreen.setPaintTicks(true);
		sBlue.setPaintTicks(true);
		sRed.setPaintLabels(true);
		sGreen.setPaintLabels(true);
		sBlue.setPaintLabels(true);
		sRed.addChangeListener(this);
		sGreen.addChangeListener(this);
		sBlue.addChangeListener(this);
		getContentPane().add(mudaCor, "South");
		mudaCor = new JPanel();
		mudaCor.add(sRed);
		mudaCor.add(sGreen);
		mudaCor.add(sBlue);
		mudaCor.add(informa);
		getContentPane().add(mudaCor);
		setSize(260, 250);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	public static void main(String[] args) {
		new Gui().setVisible(true);

	}
```

```
        public void stateChanged(ChangeEvent e) {
                mudaCor.setBackground(new Color(sRed.getValue(),
                sGreen.getValue(),
                sBlue.getValue()));

                informa.setText("RGB: " +
                mudaCor.getBackground().getRGB());
        }
}
```



JProgressBar

Um objeto do tipo **JProgressBar** cria uma barra que mostra o progresso da execução de uma determinada tarefa/atividade.

CONSTRUTOR

`JProgressBar()`
Cria uma barra de progresso horizontal com borda, porém, sem a string de progresso

`JProgressBar(BoundedRangeModel newModel)`
Cria uma barra de progresso horizontal que utiliza um modelo específico para exibir os dados do progresso

`JProgressBar(int orient)`
Cria uma barra de progresso com uma orientação específica, que pode ser:
`JProgressBar.VERTICAL`
`JProgressBar.HORIZONTAL.`

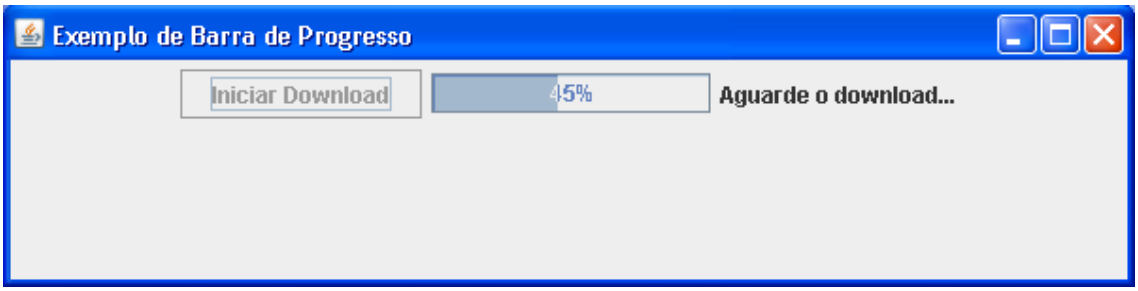
JProgressBar(int min, int max)
Cria uma barra de progresso horizontal com um valor mínimo e máximo

JProgressBar(int orient, int min, int max)
Cria uma barra de progresso com uma orientação específica e com um valor mínimo e máximo

EXEMPLO

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


public class Gui extends JFrame implements ActionListener{
  int interval = 1000;
  int i;
  JProgressBar pb;
  Timer timer;
  JButton button;
  JLabel label;

  public Gui() {
        super ("Exemplo de Barra de Progresso");
        setSize(600,150);
        setLayout(new FlowLayout());

        button = new JButton("Iniciar Download");
        button.addActionListener(this);
        add(button);

        pb = new JProgressBar(0,20);
        pb.setValue(0);
        pb.setStringPainted(true);
        add(pb);

        label = new JLabel();
        add(label);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Cria um timer
        timer = new Timer(interval, new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
              if (i == 20){
                    Toolkit.getDefaultToolkit().beep();
                    timer.stop();
                    button.setEnabled(true);
                    pb.setValue(0);
                    String str = "Download concluido";
```

```
                label.setText(str);
            }
            i = i + 1;
            pb.setValue(i);
        }
        });
    }

        public void actionPerformed(ActionEvent ae) {
                button.setEnabled(false);
                i = 0;
                String str = "Aguarde o download...";
                label.setText(str);
                timer.start();
        }


    public static void main(String[] args) {
        Gui spb = new Gui();
        spb.setVisible(true);
    }
}
```



JComboBox

Um objeto do tipo **JComboBox** cria uma caixa de combinação (lista drop-down) onde o usuário pode escolher apenas uma opção. Esta lista pode ser editável ou não.

CONSTRUTOR

JComboBox()

Cria um `JComboBox` padrão

JComboBox(ComboBoxModel aModel)

Cria um `JComboBox` que utiliza um modelo específico para exibir os dados da lista

JComboBox(Object[] items)

Cria um JComboBox com elementos de um vetor

PRINCIPAIS MÉTODOS

- Adicionar itens

```
caixa.addItem(Object);
```

- Definir qual item já virá selecionado

```
caixa.setSelectedIndex(int); //índice
caixa.setSelectedItem(Object); //item
```

- Retornar o índice do item que está selecionado

```
int n = caixa.getSelectedIndex();
```

- Retornar o item que está selecionado

```
Object n = caixa.getSelectedItem();
```

- Definir se será editável ou não

```
caixa.setEditable(boolean);
```

EXEMPLO

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

public class Gui extends JFrame implements ActionListener {
      JComboBox caixa;
      JPanel painel;
      String[] itens = new String []{ "Branco" , "Vermelho" ,
      "Verde" , "Azul" };
      Color[] cores = { Color.WHITE , Color.RED , Color.GREEN ,
      Color.BLUE };
      public Gui(){
            super("Titulo");
            painel = new JPanel();
            caixa = new JComboBox( itens );
            caixa.addActionListener( this );
            painel.add( caixa );
            this.add( painel );
            this.setBounds(10,10,200,200);
            this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
            this.setVisible( true );
      `
```

```
	public static void main(String[] args) {
		new Gui();
	}

	public void actionPerformed( ActionEvent evento ) {
		int indice = caixa.getSelectedIndex();
		painel.setBackground( cores[ indice ] );
	}
}
```

JList

Um objeto do tipo **JList** cria uma lista suspensa onde o usuário pode selecionar um ou mais itens.

CONSTRUTOR

`JList()`

Cria um `JList` padrão

`JList(Object[] items)`

Cria um `JList` com elementos de um vetor

PRINCIPAIS MÉTODOS

- Definir os itens da lista

  `lista.setListData( Object[] );`

- Desmarcar todos os objetos selecionados

  `lista.clearSelection();`

- Retornar o(s) índice(s) dos item(ns) selecionado(s)

```
int[] n = lista.getSelectedIndices();
int n = lista.getSelectedIndex();
```

- Definir o(s) item(ns) selecionado(s)

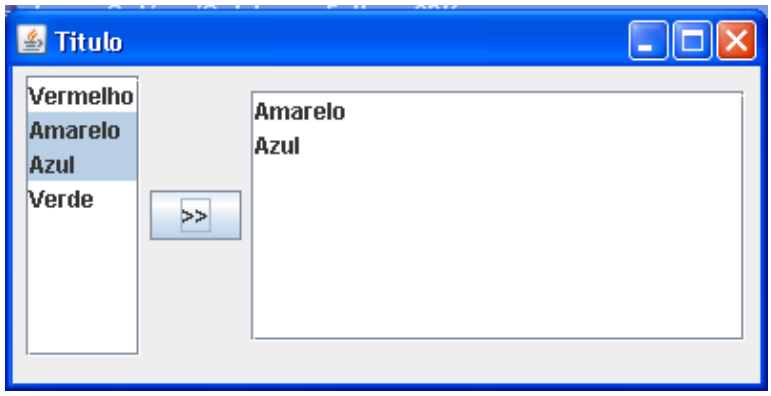
```
lista.setSelectedIndices( Object[] );
lista.setSelectedIndex( Object );
```

- Retornar o(s) item(ns) selecionado(s)

```
Object[] ob = lista.getSelectedValues();
Object ob = lista.getSelectedValue();
```

EXEMPLO

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Gui extends JFrame implements ActionListener {
      JScrollPane barraAnte, barraPos;
      JList listaAnte, listaPos;
      String[] itens = { "Vermelho","Amarelo","Azul","Verde"};
      JPanel painel;
      JButton escolha;
      public Gui(){
            super("Titulo");
            painel = new JPanel();
            listaAnte = new JList( itens );
            listaPos = new JList( );
            escolha = new JButton( ">>" );
            escolha.addActionListener( this );
            barraAnte = new JScrollPane( listaAnte );
            barraPos = new JScrollPane( listaPos );
            painel.add( barraAnte );
            painel.add( escolha );
            painel.add( barraPos );
            add( painel );
            setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
            setBounds( 10 , 10, 400, 200);
            setVisible( true );
      }
      public static void main(String[] args) {
            new Gui();
      }
      public void actionPerformed(ActionEvent evento ) {
            Object[] selecionado;
            selecionado = listaAnte.getSelectedValues();
            listaPos.setListData( selecionado );
      }
}
```



Menus

A classe Swing oferece uma série de componentes para a construção de menus. Os itens de um menu, na verdade, tem seu funcionamento e estrutura semelhantes aos botões (JButton).

Observações:

- Um item de um menu pode ter texto e ícones
- Os itens de um menu podem ser RadioButton ou CheckBox
- Cada item de menu pode ter uma combinação de teclas de atalho associado à ele

Os componentes são:

JMenuItem
JCheckBoxMenuItem
JMenu
JMenuBar

EXEMPLO

```
import javax.swing.*;
import java.awt.event.*;
public class Gui extends JFrame{
      JMenuBar menuBar;
      JMenu menu, submenu;
      JMenuItem menuItem;
      public Gui(){
            setTitle("Teste de Menus");
            setSize(500, 300);
            menuBar = new JMenuBar();
            setJMenuBar(menuBar);

            menu = new JMenu("Arquivos");
            menu.setMnemonic(KeyEvent.VK_A);
            menuBar.add(menu);
```

```
        menuItem = new JMenuItem("Salvar como...");
        menu.add(menuItem);
        menuItem = new JMenuItem("Salvar");
        menu.add(menuItem);
        menuItem = new JMenuItem("Imprimir");
        menu.add(menuItem);
        menu.addSeparator();
        menuItem = new JMenuItem("Fechar");
        menu.add(menuItem);

        menu = new JMenu("Editar");
        menu.setMnemonic(KeyEvent.VK_E);
        menuBar.add(menu);
        menuItem = new JMenuItem("Copiar");
        menu.add(menuItem);
        submenu = new JMenu("Colar");
        menu.add(submenu);
        menuItem = new JMenuItem("Colar");
        submenu.add(menuItem);
        menuItem = new JMenuItem("Colar Especial");
        submenu.add(menuItem)
    }


    public static void main(String[] args) {
            Gui janela = new Gui();
            janela.setVisible(true);
    }

}
```