JAVA JSPLITPANE

BY BISHWA KARN

- SplitPane is used to divide two components.
- The two components are divided based on the look and feel implementation, and they can be resized by the user.
- If the minimum size of the two components is greater than the size of the split pane, the divider will not allow you to resize it.
- The two components in a split pane can be aligned left to right using JSplitPane.HORIZONTAL_SPLIT, or top to bottom using JSplitPane.VERTICAL_SPLIT.
- When the user is resizing the components the minimum size of the components is used to determine the maximum/minimum position the components can be set to.

Nested Class

Modifier and Type	Class	Description
protected class	JSplitPane.AccessibleJSplitPane	This class implements accessibility support for the JsplitPane class.

Useful Fields

Modifier and Type	Field	Description
static String	BOTTOM	It use to add a Component below the other Component.
static String	CONTINUOUS_LAYOUT_PROPERTY	Bound property name for continuousLayout.
static String	DIVIDER	It uses to add a Component that will represent the divider.
static int	HORIZONTAL_SPLIT	Horizontal split indicates the Components are split along the x axis.
protected int	lastDividerLocation	Previous location of the split pane.
protected Component	leftComponent	The left or top component.
static int	VERTICAL_SPLIT	Vertical split indicates the Components are split along the y axis.
protected Component	rightComponent	The right or bottom component.
protected int	orientation	How the views are split.

Constructors

Constructor	Description
JSplitPane()	It creates a new JsplitPane configured to arrange the child components side-by-side horizontally, using two buttons for the components.
JSplitPane(int newOrientation)	It creates a new JsplitPane configured with the specified orientation.
JSplitPane(int newOrientation, boolean newContinuousLayout)	It creates a new JsplitPane with the specified orientation and redrawing style.
JSplitPane(int newOrientation, boolean newContinuousLayout, Component newLeftComponent, Component newRightComponent)	It creates a new JsplitPane with the specified orientation and redrawing style, and with the specified components.
JSplitPane(int newOrientation, Component newLeftComponent, Component newRightComponent)	It creates a new JsplitPane with the specified orientation and the specified components.

Useful Methods

Modifier and Type	Method	Description
protected void	addImpl(Component comp, Object constraints, int index)	It cdds the specified component to this split pane.
AccessibleContext	getAccessibleContext()	It gets the AccessibleContext associated with this JSplitPane.
int	getDividerLocation()	It returns the last value passed to setDividerLocation.
int	getDividerSize()	It returns the size of the divider.
Component	getBottomComponent()	It returns the component below, or to the right of the divider.
Component	getRightComponent()	It returns the component to the right (or below) the divider.
SplitPaneUI	getUI()	It returns the SplitPaneUI that is providing the current look and feel.
boolean	isContinuousLayout()	It gets the continuousLayout property.
boolean	isOneTouchExpandable()	It gets the oneTouchExpandable property.
void	setOrientation(int orientation)	It gets the orientation, or how the splitter is divided.

```
public class JSplitPaneExample {
    private static void createAndShow() {
     final JFrame frame = new JFrame("JSplitPane Example");
            frame.setSize(300, 300);
            frame.setVisible(true);
            frame.getContentPane().setLayout(new FlowLayout());

            String[] option1 = { "A","B","C","D","E" };
            JComboBox box1 = new JComboBox(option1);
            String[] option2 = {"1","2","3","4","5"};
            JComboBox box2 = new JComboBox(option2);
```

```
    Panel panel1 = new Panel();
    panel1.add(box1);
    Panel panel2 = new Panel();
    panel2.add(box2);
    JSplitPane splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, panel1, panel2);

    // JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT,  panel1, panel2);
    frame.getContentPane().add(splitPane);
}
public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        createAndShow();
      }
    });
} }
```

Java JDesktopPane

- The JDesktopPane class, can be used to create "multi-document" applications.
- A multi-document application can have many windows included in it.
- We do it by making the contentPane in the main window as an instance of the JDesktopPane class or a subclass.
- Internal windows add instances of JInternalFrame to the JdesktopPane instance.
- The internal windows are the instances of JInternalFrame or its subclasses.

Fields

Modifier and Type	Field	Description
static int	LIVE_DRAG_MODE	It indicates that the entire contents of the item being dragged should appear inside the desktop pane.
static int	OUTLINE_DRAG_MODE	It indicates that an outline only of the item being dragged should appear inside the desktop pane.

Constructor

Constructor	Description
JDesktopPane()	Creates a new JDesktopPane.

```
public class JDPaneDemo extends JFrame
{
 public JDPaneDemo()
 {
  CustomDesktopPane desktopPane = new CustomDesktopPane();
  Container contentPane = getContentPane();
  contentPane.add(desktopPane, BorderLayout.CENTER);
  desktopPane.display(desktopPane);
   setTitle("JDesktopPane Example");
  setSize(300,350);
  setVisible(true);
 }
```

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

```
class CustomDesktopPane extends JDesktopPane
{
 int numFrames = 3,  x = 30, y = 30;
 public void display(CustomDesktopPane dp)
 {
  for(int  i = 0; i < numFrames ; ++i )
  {  JInternalFrame jframe = new JInternalFrame("Internal Frame " + i ,  true, true, true, true);
     jframe.setBounds(x, y, 250, 85);
     Container c1 = jframe.getContentPane( ) ;
     c1.add(new JLabel("I love my country"));
     dp.add( jframe );
     jframe.setVisible(true);
     y += 85;
  }
 }
}
```

TabbedPane

- The **JTabbedPane** class is used to switch between a group of components by clicking on a tab with a given title or icon.
- It inherits **JComponent** class.

JTabbedPane class declaration

- Declaration for **javax.swing.JTabbedPane** class

public class JTabbedPane **extends** JComponent **implements** Serializable, Accessible, SwingConstants

Constructors

Constructor	Description
JTabbedPane()	Creates an empty TabbedPane with a default tab placement of JTabbedPane.Top.
JTabbedPane(int tabPlacement)	Creates an empty TabbedPane with a specified tab placement.
JTabbedPane(int tabPlacement, int tabLayoutPolicy)	Creates an empty TabbedPane with a specified tab placement and tab layout policy.

```
import javax.swing.*;
public class TabbedPaneExample {
JFrame f;
TabbedPaneExample(){
   f=new JFrame();
   JTextArea ta=new JTextArea(200,200);
   JPanel p1=new JPanel();
   p1.add(ta);
   JPanel p2=new JPanel();
   JPanel p3=new JPanel();
   JTabbedPane tp=new JTabbedPane();
```

```
    tp.setBounds(50,50,200,200);
     tp.add("main",p1);
     tp.add("visit",p2);
     tp.add("help",p3);
     f.add(tp);
     f.setSize(400,400);
     f.setLayout(null);
     f.setVisible(true);
}
public static void main(String[] args) {
     new TabbedPaneExample();
}}
```

JInternalFrame

- With the **JInternalFrame** class you can display a **Jframe-like** window within another window.
- Usually, you add internal frames to a desktop pane.
- The desktop pane, in turn, might be used as the content pane of a **JFrame**.
- The desktop pane is an instance of **JDesktopPane**, which is a subclass of **JLayeredPane** that has added API for managing multiple overlapping internal frames.

```
import javax.swing.*;
   public class JInternalFrameExample {
         public static void main(String[] args) {
                  JFrame.setDefaultLookAndFeelDecorated(true);
                  JFrame frame = new JFrame("JInternalFrame Example");
                  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                  frame.setSize(300,300);
                  JDesktopPane desktop = new JDesktopPane();
                  JInternalFrame if1 = new JInternalFrame("Frame 1", true, true, true, true );
                  if1.setSize(200,200);
                  desktop.add(if1);
                  JInternalFrame if2 = new JInternalFrame("Frame 2", true, true, true, true );
```

```
            if2.setSize(200,200);
            desktop.add(if2);

            if1.setLocation(20,20);
            if1.setVisible(true);
            if2.setLocation(40,40);
            if2.setVisible(true);

            frame.add(desktop);
            frame.setVisible(true);
        }
    }
```