Swing Continued..

JOptionPane

The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user. The JOptionPane class inherits JComponent class.

JOptionPane class declaration

public class JOptionPane extends JComponent implements Accessible

Common Constructors of JOptionPane class

Constructor	Description
JOptionPane()	It is used to create a JOptionPane with a test message.
JOptionPane(Object message)	It is used to create an instance of JOptionPane to display a message.
JOptionPane(Object message, int messageType	It is used to create an instance of JOptionPane to display a message with specified message type and default options.

Common Methods of JOptionPane class

Methods	Description
JDialog createDialog(String title)	It is used to create and return a new parentless JDialog with the specified title.
static void showMessageDialog(Component parentComponent, Object message)	It is used to create an information-message dialog titled "Message".
static void showMessageDialog(Component parentComponent, Object message, String title, int messageType)	It is used to create a message dialog with given title and messageType.
static int showConfirmDialog(Component parentComponent, Object message)	It is used to create a dialog with the options Yes, No and Cancel; with the title, Select an Option.
static String showInputDialog(Component parentComponent, Object message)	It is used to show a question-message dialog requesting input from the user parented to parentComponent.
void setInputValue(Object newValue)	It is used to set the input value that was selected or input by the user.

JOptionPane Example: showMessageDialog()

```
import javax.swing.*;
public class OptionPaneExample
{
        JFrame f;
```

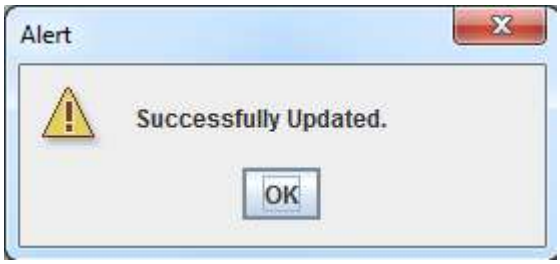
```
        OptionPaneExample()
        {
                f=new JFrame();
                JOptionPane.showMessageDialog(f,"Hello, Welcome to Java.");
        }
         public static void main(String[] args)
        {
                new OptionPaneExample();
        }
}
```

Output:

JOptionPane Example: showMessageDialog()

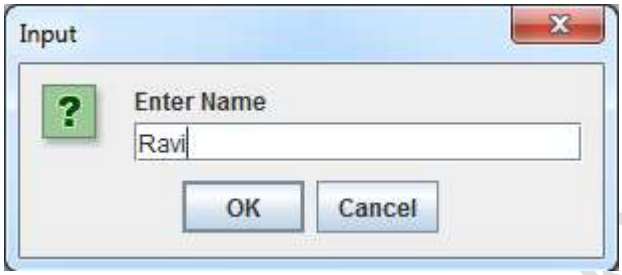
```
import javax.swing.*;
public class OptionPaneExample1
{
        JFrame f;
        OptionPaneExample1()
        {
                f=new JFrame();
                JOptionPane.showMessageDialog(f,"Successfully
Updated.","Alert",JOptionPane.WARNING_MESSAGE);
        }
        public static void main(String[] args)
        {
                new OptionPaneExample1();
        }
}
```


Output:



JOptionPane Example: showInputDialog()

```
import javax.swing.*;
public class OptionPaneExample2
{
        JFrame f;
        OptionPaneExample2()
        {
                f=new JFrame();
                String name=JOptionPane.showInputDialog(f,"Enter Name");
        }
        public static void main(String[] args)
        {
                new OptionPaneExample2();
        }
}
```



JScrollBar

The object of JScrollbar class is used to add horizontal and vertical scrollbar. It is an implementation of a scrollbar. It inherits JComponent class.

JScrollBar class declaration

public class JScrollBar extends JComponent implements Adjustable, Accessible

Commonly used Constructors:

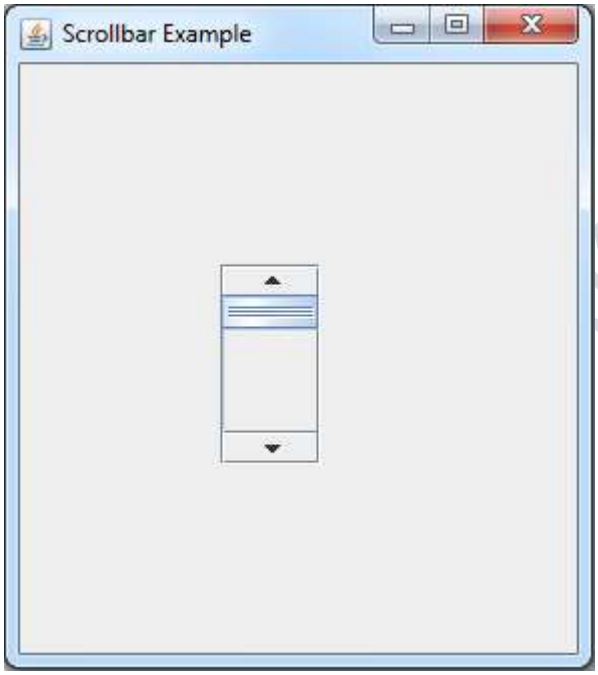
Constructor	Description
JScrollBar()	Creates a vertical scrollbar with the initial values.
JScrollBar(int orientation)	Creates a scrollbar with the specified orientation and the initial values.
JScrollBar(int orientation, int value, int extent, int min, int max)	Creates a scrollbar with the specified orientation, value, extent, minimum, and maximum.

JScrollBar Example

```
import javax.swing.*;
class ScrollBarExample
{
        ScrollBarExample()
```

```
    {
        JFrame f= new JFrame("Scrollbar Example");
        JScrollBar s=new JScrollBar();
        s.setBounds(100,100, 50,100);
        f.add(s);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[])
    {
        new ScrollBarExample();
    }
}
```

Output:



JMenuBar, JMenu and JMenuItem

The JMenuBar class is used to display menubar on the window or frame. It may have several menus. The object of JMenu class is a pull down menu component which is displayed from the menu bar. It inherits the JMenuItem class. The object of JMenuItem class adds a simple labeled menu item. The items used in a menu must belong to the JMenuItem or any of its subclass.

JMenuBar class declaration

public class JMenuBar extends JComponent implements MenuElement, Accessible

JMenu class declaration

public class JMenu extends JMenuItem implements MenuElement, Accessible

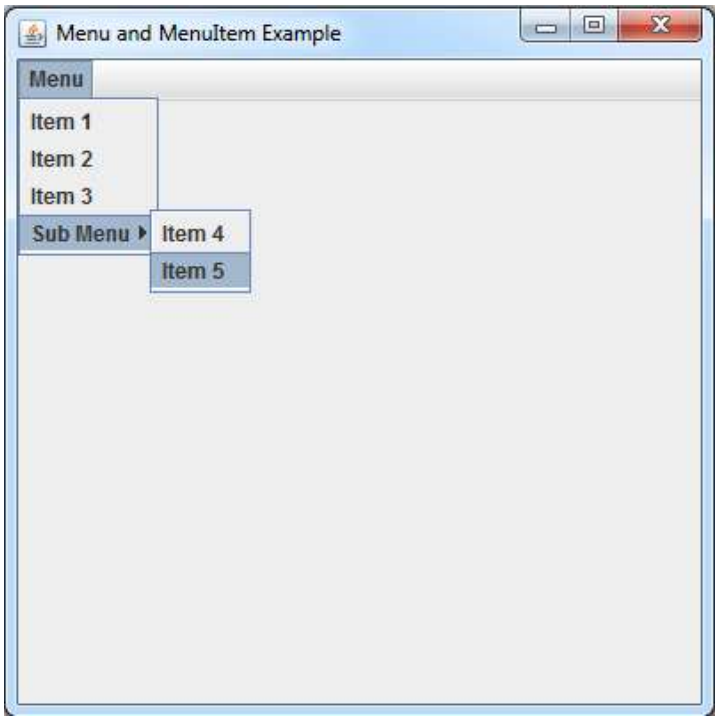
JMenuItem class declaration

public class JMenuItem extends AbstractButton implements Accessible, MenuElement

Java JMenuItem and JMenu Example

```
import javax.swing.*;
class MenuExample
{
	JMenu menu, submenu;
	JMenuItem i1, i2, i3, i4, i5;
	MenuExample()
	{
		JFrame f= new JFrame("Menu and MenuItem Example");
		JMenuBar mb=new JMenuBar();
		menu=new JMenu("Menu");
		submenu=new JMenu("Sub Menu");
		i1=new JMenuItem("Item 1");
		i2=new JMenuItem("Item 2");
		i3=new JMenuItem("Item 3");
		i4=new JMenuItem("Item 4");
		i5=new JMenuItem("Item 5");
		menu.add(i1); menu.add(i2); menu.add(i3);
		submenu.add(i4); submenu.add(i5);
		menu.add(submenu);
		mb.add(menu);
		f.setJMenuBar(mb);
		f.setSize(400,400);
		f.setLayout(null);
		f.setVisible(true);
	}
	public static void main(String args[])
	{
		new MenuExample();
	}
}
```

Output:



JProgressBar

The JProgressBar class is used to display the progress of the task. It inherits JComponent class.

JProgressBar class declaration

public class JProgressBar extends JComponent implements SwingConstants, Accessible

Commonly used Constructors:

Constructor	Description
JProgressBar()	It is used to create a horizontal progress bar but no string text.
JProgressBar(int min, int max)	It is used to create a horizontal progress bar with the specified minimum and maximum value.
JProgressBar(int orient)	It is used to create a progress bar with the specified orientation, it can be either Vertical or Horizontal by using Swing Constants .VERTICAL and Swing Constants. HORIZONTAL constants.
JProgressBar(int orient, int min, int max)	It is used to create a progress bar with the specified orientation, minimum and maximum value.

Commonly used Methods:

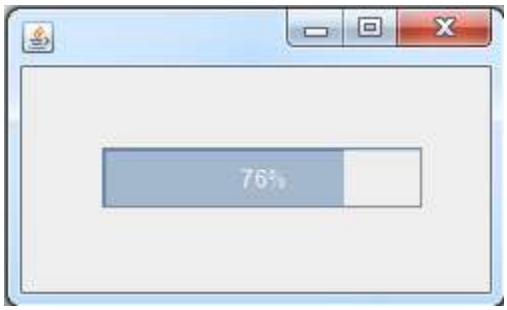
Method	Description
void setStringPainted(boolean b)	It is used to determine whether string should be displayed.
void setString(String s)	It is used to set value to the progress string.
void setOrientation(int	It is used to set the orientation, it may be either vertical or horizontal

orientation)	by using SwingConstants.VERTICAL and SwingConstants.HORIZONTAL constants.
void setValue(int value)	It is used to set the current value on the progress bar.

JProgressBar Example

```
import javax.swing.*;
public class ProgressBarExample extends JFrame
{
	JProgressBar jb;
	int i=0,num=0;
	ProgressBarExample()
	{
		jb=new JProgressBar(0,2000);
		jb.setBounds(40,40,160,30);
		jb.setValue(0);
		jb.setStringPainted(true);
		add(jb);
		setSize(250,150);
		setLayout(null);
	}
	public void iterate()
	{
		while(i<=2000)
		{
			jb.setValue(i);
			i=i+20;
			try
			{
				Thread.sleep(150);
			}
			catch(Exception e)
			{}
		}
	}
	public static void main(String[] args)
	{
		ProgressBarExample m=new ProgressBarExample();
		m.setVisible(true);
		m.iterate();
	}
}
```

Output:



JSlider

The Java JSlider class is used to create the slider. By using JSlider, a user can select a value from a specific range.

Commonly used Constructors of JSlider class

Constructor	Description
JSlider()	creates a slider with the initial value of 50 and range of 0 to 100.
JSlider(int orientation)	creates a slider with the specified orientation set by either JSlider.HORIZONTAL or JSlider.VERTICAL with the range 0 to 100 and initial value 50.
JSlider(int min, int max)	creates a horizontal slider using the given min and max.
JSlider(int min, int max, int value)	creates a horizontal slider using the given min, max and value.
JSlider(int orientation, int min, int max, int value)	creates a slider using the given orientation, min, max and value.

Commonly used Methods of JSlider class

Method	Description
public void setMinorTickSpacing(int n)	is used to set the minor tick spacing to the slider.
public void setMajorTickSpacing(int n)	is used to set the major tick spacing to the slider.
public void setPaintTicks(boolean b)	is used to determine whether tick marks are painted.
public void setPaintLabels(boolean b)	is used to determine whether labels are painted.
public void setPaintTracks(boolean b)	is used to determine whether track is painted.

Java JSlider Example

```
import javax.swing.*;
public class SliderExample1 extends JFrame
{
   public SliderExample1()
        {
                JSlider slider = new JSlider(JSlider.HORIZONTAL, 0, 50, 25);
```

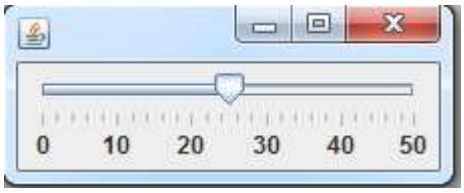
```
                JPanel panel=new JPanel();
                panel.add(slider);
                add(panel);
    }
    public static void main(String s[])
        {
                SliderExample1 frame=new SliderExample1();
                frame.pack();
                frame.setVisible(true);
    }
}
```

Output:

JSlider Example: painting ticks

```
import javax.swing.*;
public class SliderExample2 extends JFrame
{
        public SliderExample2()
        {
                JSlider slider = new JSlider(JSlider.HORIZONTAL, 0, 50, 25);
                slider.setMinorTickSpacing(2);
                slider.setMajorTickSpacing(10);
                slider.setPaintTicks(true);
                slider.setPaintLabels(true);
                JPanel panel=new JPanel();
                panel.add(slider);
                add(panel);
        }
        public static void main(String s[])
        {
                SliderExample2 frame=new SliderExample2();
                frame.pack();
                frame.setVisible(true);
         }
}
```

Output:



Dialog

The JDialog control represents a top level window with a border and a title used to take some form of input from the user. It inherits the Dialog class. Unlike JFrame, it doesn't have maximize and minimize buttons.

JDialog class declaration

public class JDialog extends Dialog implements WindowConstants, Accessible, RootPaneContai ner

Commonly used Constructors:

Constructor	Description
JDialog()	It is used to create a modeless dialog without a title and without a specified Frame owner.
JDialog(Frame owner)	It is used to create a modeless dialog with specified Frame as its owner and an empty title.
JDialog(Frame owner, String title, boolean modal)	It is used to create a dialog with the specified title, owner Frame and modality.

JDialog Example

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class DialogExample
{
    private static JDialog d;
    DialogExample()
    {
        JFrame f= new JFrame();
        d = new JDialog(f , "Dialog Example", true);
        d.setLayout( new FlowLayout() );
        JButton b = new JButton ("OK");
        b.addActionListener ( new ActionListener()
        {
            public void actionPerformed( ActionEvent e )
            {
                DialogExample.d.setVisible(false);
            }
```

```
            });
            d.add( new JLabel ("Click button to continue."));
            d.add(b);
            d.setSize(300,300);
            d.setVisible(true);
        }
        public static void main(String args[])
        {
            new DialogExample();
        }
}
```

Output:

JPanel

The JPanel is a simplest container class. It provides space in which an application can attach any other component. It inherits the JComponents class. It doesn't have title bar.

JPanel class declaration

public class JPanel extends JComponent implements Accessible

Commonly used Constructors:

Constructor	Description
JPanel()	It is used to create a new JPanel with a double buffer and a flow layout.
JPanel(boolean isDoubleBuffered)	It is used to create a new JPanel with FlowLayout and the specified buffering strategy.
JPanel(LayoutManager layout)	It is used to create a new JPanel with the specified layout manager.

JPanel Example

```
import java.awt.*;
```

```
import javax.swing.*;
public class PanelExample
{
	PanelExample()
	{
		JFrame f= new JFrame("Panel Example");
		JPanel panel=new JPanel();
		panel.setBounds(40,80,200,200);
		panel.setBackground(Color.gray);
		JButton b1=new JButton("Button 1");
		b1.setBounds(50,100,80,30);
		b1.setBackground(Color.yellow);
		JButton b2=new JButton("Button 2");
		b2.setBounds(100,100,80,30);
		b2.setBackground(Color.green);
		panel.add(b1); panel.add(b2);
		f.add(panel);
		f.setSize(400,400);
		f.setLayout(null);
		f.setVisible(true);
	}
	public static void main(String args[])
	{
		new PanelExample();
	}
}
```


Output: