

The background is a solid blue color with a gradient. At the top, there are several wavy, horizontal lines in shades of blue and teal, creating a sense of movement or a horizon line. The word "SWING" is written in a light blue, sans-serif font, positioned on the right side of the image.

SWING

- 
- JAVA provides a rich set of libraries to create Graphical User Interface in a platform independent way. In this PPT , we'll look at SWING GUI controls.

SWING - Overview

- Swing API is a set of extensible GUI Components to ease the developer's life to create JAVA based Front End/GUI Applications. It is build on top of AWT API and acts as a replacement of AWT API, since it has almost every control corresponding to AWT controls. Swing component follows a Model-View-Controller architecture to fulfill the following criterias.
 - A single API is to be sufficient to support multiple look and feel.
 - API is to be model driven so that the highest level API is not required to have data.
 - API is to use the Java Bean model so that Builder Tools and IDE can provide better services to the developers for use.

MVC Architecture

- Swing API architecture follows loosely based MVC architecture in the following manner.
 - Model represents component's data.
 - View represents visual representation of the component's data.
 - Controller takes the input from the user on the view and reflects the changes in Component's data.
 - Swing component has Model as a separate element, while the View and Controller part are clubbed in the User Interface elements. Because of which, Swing has a pluggable look-and-feel architecture.

Swing Features

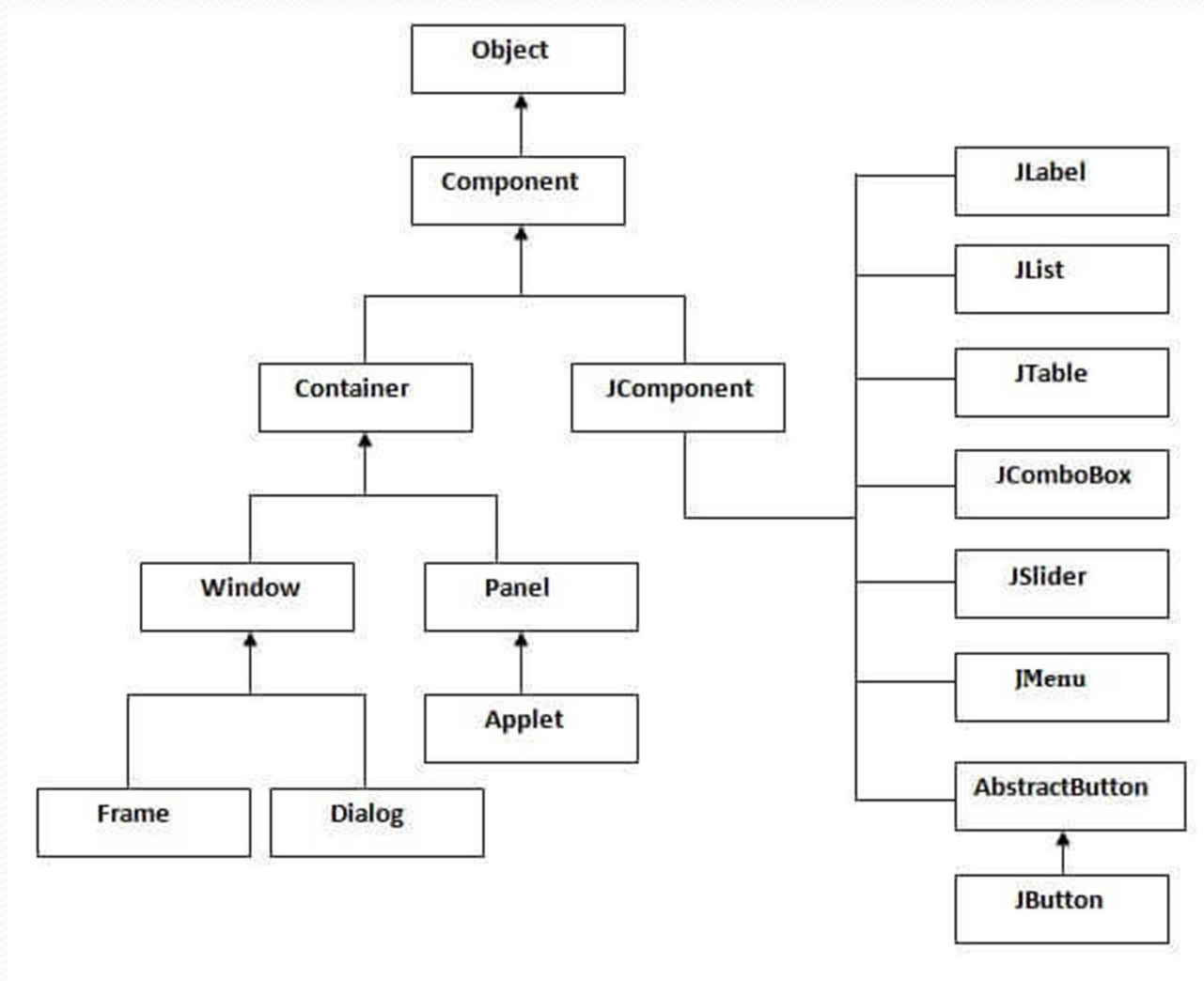
- **Light Weight** – Swing components are independent of native Operating System's API as Swing API controls are rendered mostly using pure JAVA code instead of underlying operating system calls.
- **Rich Controls** – Swing provides a rich set of advanced controls like Tree, TabbedPane, slider, colorpicker, and table controls.
- **Highly Customizable** – Swing controls can be customized in a very easy way as visual appearance is independent of internal representation.
- **Pluggable look-and-feel** – SWING based GUI Application look and feel can be changed at run-time, based on available values.



SWING - Controls

- Every user interface considers the following three main aspects –
 - **UI Elements** – These are the core visual elements the user eventually sees and interacts with. GWT provides a huge list of widely used and common elements varying from basic to complex, which we will cover in this tutorial.
 - **Layouts** – They define how UI elements should be organized on the screen and provide a final look and feel to the GUI (Graphical User Interface). This part will be covered in the Layout chapter.
 - **Behavior** – These are the events which occur when the user interacts with UI elements

Hierarchy of Java Swing classes



Commonly used Methods of Component class

Method	Description
<code>public void add(Component c)</code>	add a component on another component.
<code>public void setSize(int width,int height)</code>	sets size of the component.
<code>public void setLayout(LayoutManager m)</code>	sets the layout manager for the component.
<code>public void setVisible(boolean b)</code>	sets the visibility of the component. It is by default false.

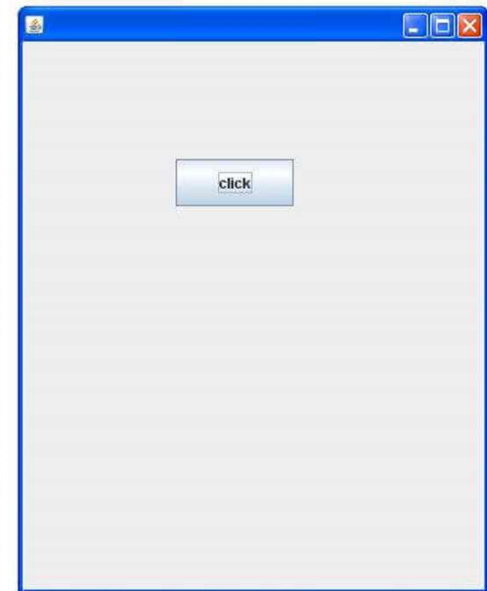
Java Swing Examples

- There are two ways to create a frame:
 - By creating the object of Frame class (association)
 - By extending Frame class (inheritance)

We can write the code of swing inside the main(), constructor or any other method.

Simple Java Swing Example

```
import javax.swing.*;  
public class FirstSwingExample {  
  public static void main(String[] args) {  
    JFrame f=new JFrame();//creating instance of JFrame  
  
    JButton b=new JButton("click");//creating instance of JButton  
    b.setBounds(130,100,100, 40);//x axis, y axis, width, height  
  
    f.add(b);//adding button in JFrame  
  
    f.setSize(400,500);//400 width and 500 height  
    f.setLayout(null);//using no layout managers  
    f.setVisible(true);//making the frame visible  
  }  
}
```



Example of Swing by Association inside constructor

```
import javax.swing.*;
public class Simple {
    JFrame f;
    Simple(){
        f=new JFrame();//creating instance of JFrame

        JButton b=new JButton("click");//creating instance of JButton
        b.setBounds(130,100,100, 40);

        f.add(b);//adding button in JFrame

        f.setSize(400,500);//400 width and 500 height
        f.setLayout(null);//using no layout managers
        f.setVisible(true);//making the frame visible
    }

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

Java JButton

- The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.



JButton class declaration

public class JButton extends AbstractButton implements Accessible

- Commonly used Constructors:

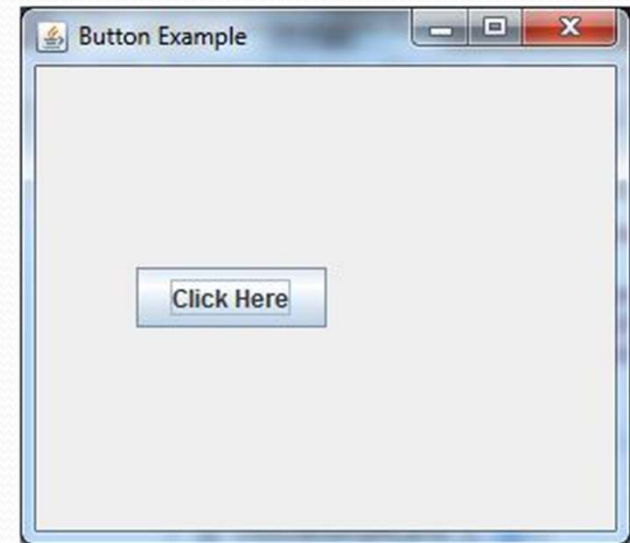
Constructor	Description
JButton()	It creates a button with no text and icon.
JButton(String s)	It creates a button with the specified text.
JButton(Icon i)	It creates a button with the specified icon object.

Commonly used Methods of AbstractButton class:

Methods	Description
<code>void setText(String s)</code>	It is used to set specified text on button
<code>String getText()</code>	It is used to return the text of the button.
<code>void setEnabled(boolean b)</code>	It is used to enable or disable the button.
<code>void setIcon(Icon b)</code>	It is used to set the specified Icon on the button.
<code>Icon getIcon()</code>	It is used to get the Icon of the button.
<code>void setMnemonic(int a)</code>	It is used to set the mnemonic on the button.
<code>void addActionListener(ActionListener a)</code>	It is used to add the <u>action listener</u> to this object.

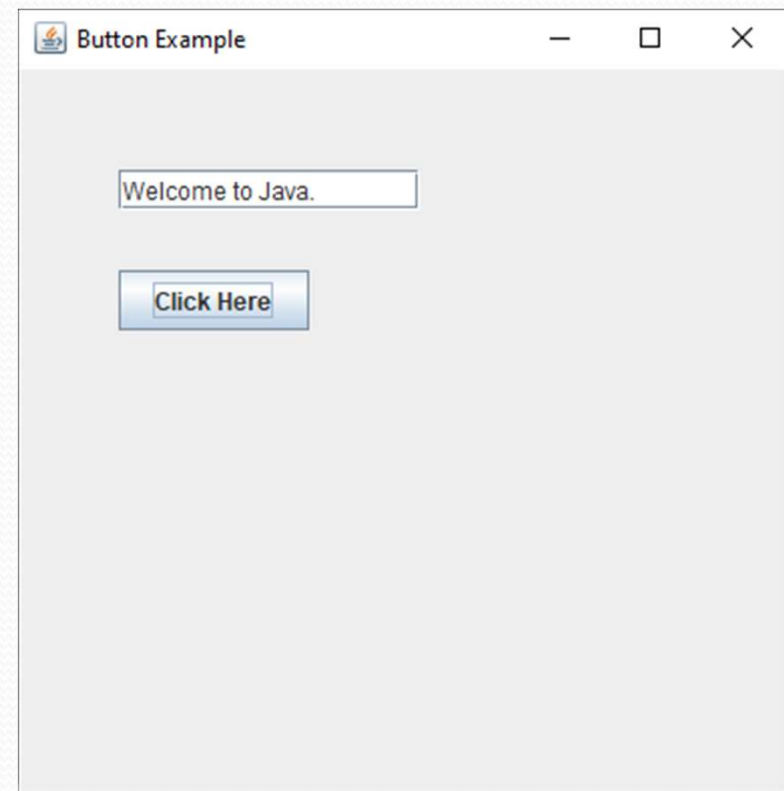
Java JButton Example

```
import javax.swing.*;  
public class ButtonExample {  
  public static void main(String[] args) {  
    JFrame f=new JFrame("Button Example");  
    JButton b=new JButton("Click Here");  
    b.setBounds(50,100,95,30);  
    f.add(b);  
    f.setSize(400,400);  
    f.setLayout(null);  
    f.setVisible(true);  
  }  
}
```



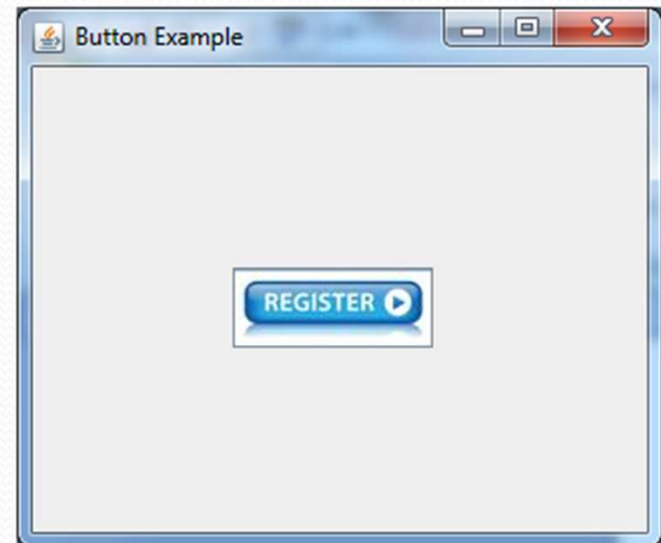
Java JButton Example with ActionListener

```
import java.awt.event.*;
import javax.swing.*;
public class sw2 {
    public static void main(String[] args) {
        JFrame f=new JFrame("Button Example");
        final JTextField tf=new JTextField();
        tf.setBounds(50,50, 150,20);
        JButton b=new JButton("Click Here");
        b.setBounds(50,100,95,30);
        b.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                tf.setText("Welcome to Java.");
            }
        });
        f.add(b);f.add(tf);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```



Example of displaying image on the button:

```
import javax.swing.*;
public class ButtonExample{
    ButtonExample(){
        JFrame f=new JFrame("Button Example");
        JButton b=new JButton(new ImageIcon("D:\\icon.png"));
        b.setBounds(100,100,100, 40);
        f.add(b);
        f.setSize(300,400);
        f.setLayout(null);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        new ButtonExample();
    }
}
```



Java JLabel

- The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

Commonly used Constructors:

Constructor	Description
<code>JLabel()</code>	Creates a JLabel instance with no image and with an empty string for the title.
<code>JLabel(String s)</code>	Creates a JLabel instance with the specified text.
<code>JLabel(Icon i)</code>	Creates a JLabel instance with the specified image.
<code>JLabel(String s, Icon i, int horizontalAlignment)</code>	Creates a JLabel instance with the specified text, image, and horizontal alignment.

Commonly used Methods:

Methods	Description
String getText()	It returns the text string that a label displays.
void setText(String text)	It defines the single line of text this component will display.
void setHorizontalAlignment(int alignment)	It sets the alignment of the label's contents along the X axis.
Icon getIcon()	It returns the graphic image that the label displays.
int getHorizontalAlignment()	It returns the alignment of the label's contents along the X axis.

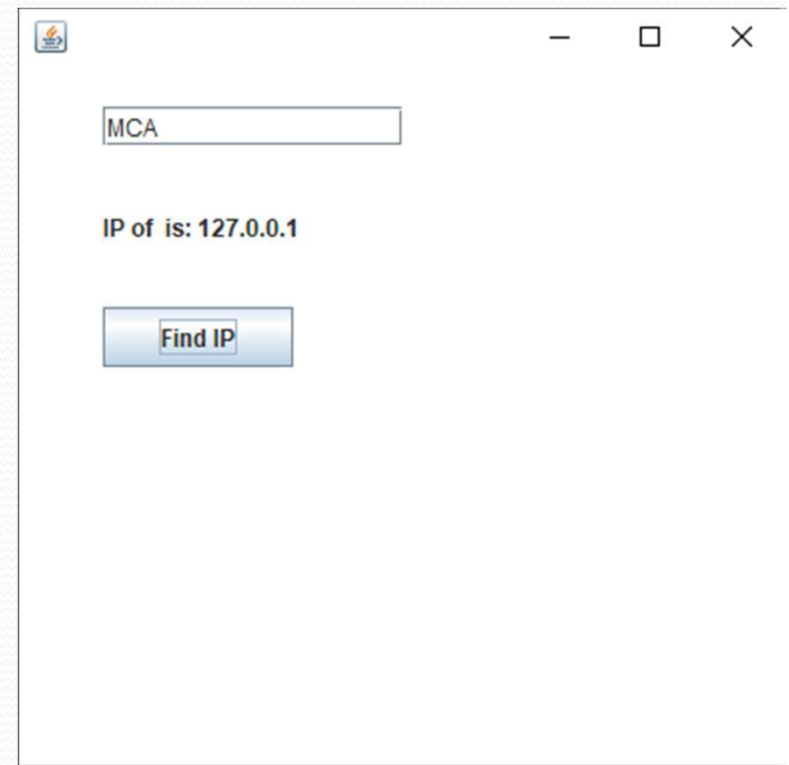
Java JLabel Example

```
import javax.swing.*;  
class LabelExample  
{  
    public static void main(String args[])  
    {  
        JFrame f= new JFrame("Label Example");  
        JLabel l1,l2;  
        l1=new JLabel("First Label.");  
        l1.setBounds(50,50, 100,30);  
        l2=new JLabel("Second Label.");  
        l2.setBounds(50,100, 100,30);  
        f.add(l1); f.add(l2);  
        f.setSize(300,300);  
        f.setLayout(null);  
        f.setVisible(true);  
    }  
}
```



Java JLabel Example with ActionListener

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class LabelExample extends Frame implements ActionListener{
    JTextField tf; JLabel l; JButton b;
    LabelExample(){
        tf=new JTextField();
        tf.setBounds(50,50, 150,20);
        l=new JLabel();
        l.setBounds(50,100, 250,20);
        b=new JButton("Find IP");
        b.setBounds(50,150,95,30);
        b.addActionListener(this);
        add(b);add(tf);add(l);
        setSize(400,400);
        setLayout(null);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        try{
            String host=tf.getText();
            String ip=java.net.InetAddress.getByName(host).getHostAddress();
            l.setText("IP of "+host+" is: "+ip);
        }catch(Exception ex){System.out.println(ex);}
    }
    public static void main(String[] args) {
        new LabelExample();
    }
}
```



Java JTextField

The object of a JTextField class is a text component that allows the editing of a single line text. It inherits JTextComponent class.

JTextField class declaration

- Let's see the declaration for javax.swing.JTextField class.

```
public class JTextField extends JTextComponent implements SwingConstants
```

Commonly used Constructors:

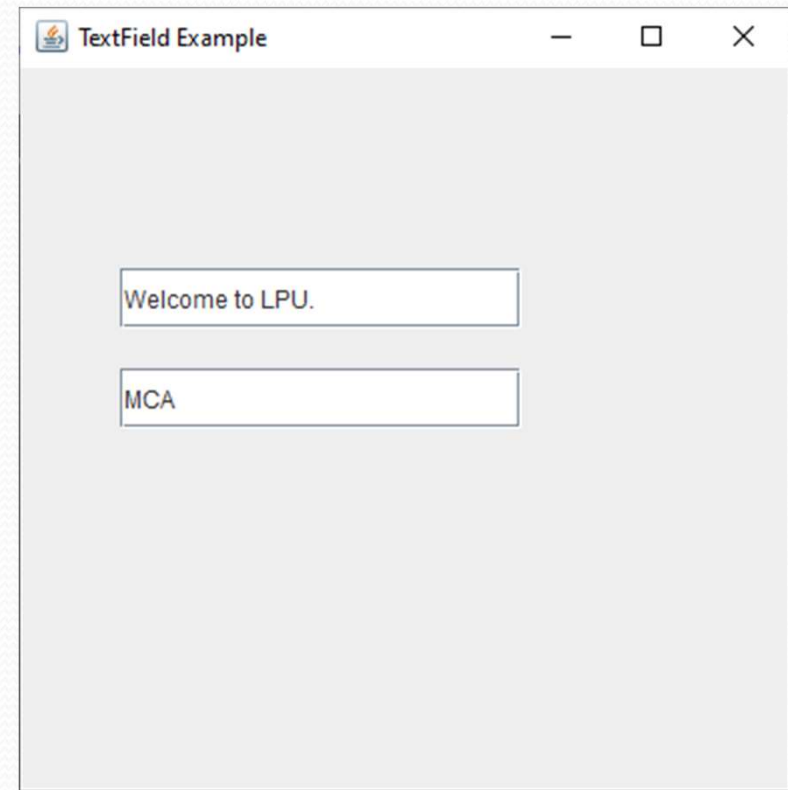
Constructor	Description
<code>JTextField()</code>	Creates a new TextField
<code>JTextField(String text)</code>	Creates a new TextField initialized with the specified text.
<code>JTextField(String text, int columns)</code>	Creates a new TextField initialized with the specified text and columns.
<code>JTextField(int columns)</code>	Creates a new empty TextField with the specified number of columns.

Commonly used Methods:

Methods	Description
<code>void addActionListener(ActionListener l)</code>	It is used to add the specified action listener to receive action events from this textfield.
<code>Action getAction()</code>	It returns the currently set Action for this ActionEvent source, or null if no Action is set.
<code>void setFont(Font f)</code>	It is used to set the current font.
<code>void removeActionListener(ActionListener l)</code>	It is used to remove the specified action listener so that it no longer receives action events from this textfield.

Java JTextField Example

```
import javax.swing.*;
class TextFieldExample
{
    public static void main(String args[])
    {
        JFrame f= new JFrame("TextField Example");
        JTextField t1,t2;
        t1=new JTextField("Welcome to LPU.");
        t1.setBounds(50,100, 200,30);
        t2=new JTextField("MCA");
        t2.setBounds(50,150, 200,30);
        f.add(t1); f.add(t2);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```



Java JTextArea

The object of a JTextArea class is a multi line region that displays text. It allows the editing of multiple line text. It inherits JTextComponent class

JTextArea class declaration

Let's see the declaration for javax.swing.JTextArea class.

- **public class JTextArea extends JTextComponent**

Commonly used Constructors:

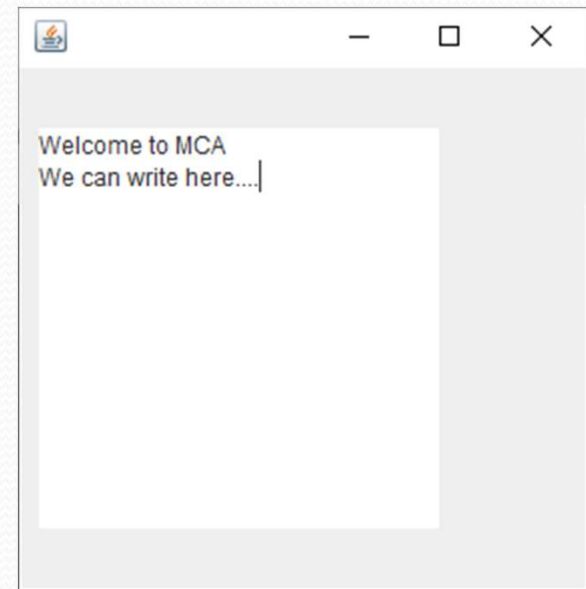
Constructor	Description
<code>JTextArea()</code>	Creates a text area that displays no text initially.
<code>JTextArea(String s)</code>	Creates a text area that displays specified text initially.
<code>JTextArea(int row, int column)</code>	Creates a text area with the specified number of rows and columns that displays no text initially.
<code>JTextArea(String s, int row, int column)</code>	Creates a text area with the specified number of rows and columns that displays specified text.

Commonly used Methods:

Methods	Description
<code>void setRows(int rows)</code>	It is used to set specified number of rows.
<code>void setColumns(int cols)</code>	It is used to set specified number of columns.
<code>void setFont(Font f)</code>	It is used to set the specified font.
<code>void insert(String s, int position)</code>	It is used to insert the specified text on the specified position.
<code>void append(String s)</code>	It is used to append the given text to the end of the document.

Java JTextArea Example

```
import javax.swing.*;
public class texta
{
    texta(){
        JFrame f= new JFrame();
        JTextArea area=new JTextArea("Welcome to MCA");
        area.setBounds(10,30, 200,200);
        f.add(area);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[])
    {
        new texta();
    }
}
```



Java JPasswordField

- The object of a JPasswordField class is a text component specialized for password entry. It allows the editing of a single line of text. It inherits JTextField class.

JPasswordField class declaration

- Let's see the declaration for javax.swing.JPasswordField class.
 - **public class JPasswordField extends JTextField**

Commonly used Constructors:

Constructor	Description
<code>JPasswordField()</code>	Constructs a new <code>JPasswordField</code> , with a default document, null starting text string, and 0 column width.
<code>JPasswordField(int columns)</code>	Constructs a new empty <code>JPasswordField</code> with the specified number of columns.
<code>JPasswordField(String text)</code>	Constructs a new <code>JPasswordField</code> initialized with the specified text.
<code>JPasswordField(String text, int columns)</code>	Construct a new <code>JPasswordField</code> initialized with the specified text and columns.

Java JPasswordField Example

```
import javax.swing.*;  
public class PasswordFieldExample {  
    public static void main(String[] args) {  
        JFrame f=new JFrame("Password Field Example");  
        JPasswordField value = new JPasswordField();  
        JLabel l1=new JLabel("Password:");  
        l1.setBounds(20,100, 80,30);  
        value.setBounds(100,100,100,30);  
        f.add(value); f.add(l1);  
        f.setSize(300,300);  
        f.setLayout(null);  
        f.setVisible(true);  
    }  
}
```



Java JCheckBox

The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on ".It inherits JToggleButton class.

JCheckBox class declaration

- Let's see the declaration for javax.swing.JCheckBox class.
 - **public class JCheckBox extends JToggleButton implements Accessible**

Commonly used Constructors:

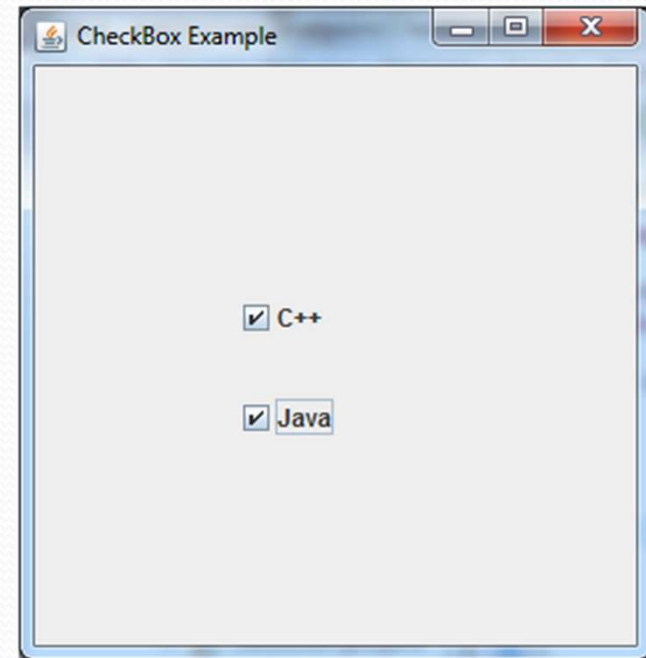
Constructor	Description
JJCheckBox()	Creates an initially unselected check box button with no text, no icon.
JChechBox(String s)	Creates an initially unselected check box with text.
JCheckBox(String text, boolean selected)	Creates a check box with text and specifies whether or not it is initially selected.
JCheckBox(Action a)	Creates a check box where properties are taken from the Action supplied.

Commonly used Methods:

Methods	Description
AccessibleContext getAccessibleContext()	It is used to get the AccessibleContext associated with this JCheckBox.
protected String paramString()	It returns a <u>string</u> representation of this JCheckBox.

Java JCheckBox Example

```
import javax.swing.*;
public class CheckBoxExample
{
    CheckBoxExample(){
        JFrame f= new JFrame("CheckBox Example");
        JCheckBox checkBox1 = new JCheckBox("C++");
        checkBox1.setBounds(100,100, 50,50);
        JCheckBox checkBox2 = new JCheckBox("Java", true);
        checkBox2.setBounds(100,150, 50,50);
        f.add(checkBox1);
        f.add(checkBox2);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[])
    {
        new CheckBoxExample();
    }
}
```



Java JRadioButton

- The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz.

It should be added in ButtonGroup to select one radio button only.

JRadioButton class declaration

Let's see the declaration for javax.swing.JRadioButton class.

- **public class JRadioButton extends JToggleButton implements Accessible**

Commonly used Constructors:

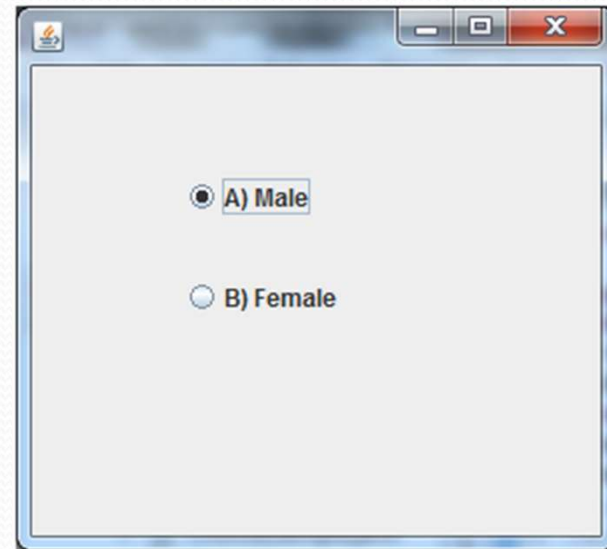
Constructor	Description
JRadioButton()	Creates an unselected radio button with no text.
JRadioButton(String s)	Creates an unselected radio button with specified text.
JRadioButton(String s, boolean selected)	Creates a radio button with the specified text and selected status.

Commonly used Methods:

Methods	Description
<code>void setText(String s)</code>	It is used to set specified text on button.
<code>String getText()</code>	It is used to return the text of the button.
<code>void setEnabled(boolean b)</code>	It is used to enable or disable the button.
<code>void setIcon(Icon b)</code>	It is used to set the specified Icon on the button.
<code>Icon getIcon()</code>	It is used to get the Icon of the button.
<code>void setMnemonic(int a)</code>	It is used to set the mnemonic on the button.
<code>void addActionListener(ActionListener a)</code>	It is used to add the action listener to this object.

Java JRadioButton Example

```
import javax.swing.*;
public class RadioButtonExample {
    JFrame f;
    RadioButtonExample(){
        f=new JFrame();
        JRadioButton r1=new JRadioButton("A) Male");
        JRadioButton r2=new JRadioButton("B) Female");
        r1.setBounds(75,50,100,30);
        r2.setBounds(75,100,100,30);
        ButtonGroup bg=new ButtonGroup();
        bg.add(r1);bg.add(r2);
        f.add(r1);f.add(r2);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new RadioButtonExample();
    }
}
```



Java JComboBox

- The object of Choice class is used to show popup menu of choices. Choice selected by user is shown on the top of a menu. It inherits JComponent class.

JComboBox class declaration

- **public class JComboBox extends JComponent implements ItemSelectable, ListDataListener, ActionListener, Accessible**

Commonly used Constructors:

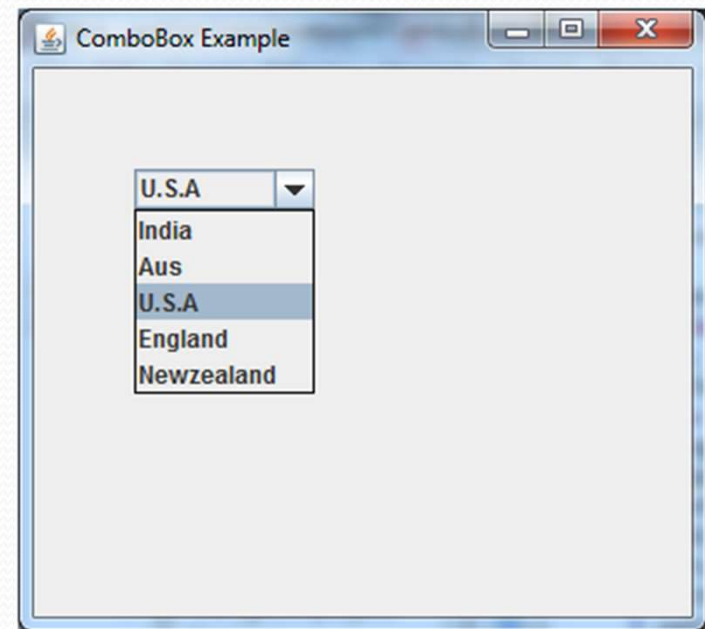
Constructor	Description
JComboBox()	Creates a JComboBox with a default data model.
JComboBox(Object[] items)	Creates a JComboBox that contains the elements in the specified <u>array</u> .
JComboBox(Vector<?> items)	Creates a JComboBox that contains the elements in the specified <u>Vector</u> .

Commonly used Methods:

Methods	Description
<code>void addItem(Object anObject)</code>	It is used to add an item to the item list.
<code>void removeItem(Object anObject)</code>	It is used to delete an item to the item list.
<code>void removeAllItems()</code>	It is used to remove all the items from the list.
<code>void setEditable(boolean b)</code>	It is used to determine whether the JComboBox is editable.
<code>void addActionListener(ActionListener a)</code>	It is used to add the ActionListener .
<code>void addItemListener(ItemListener i)</code>	It is used to add the ItemListener .

Java JComboBox Example

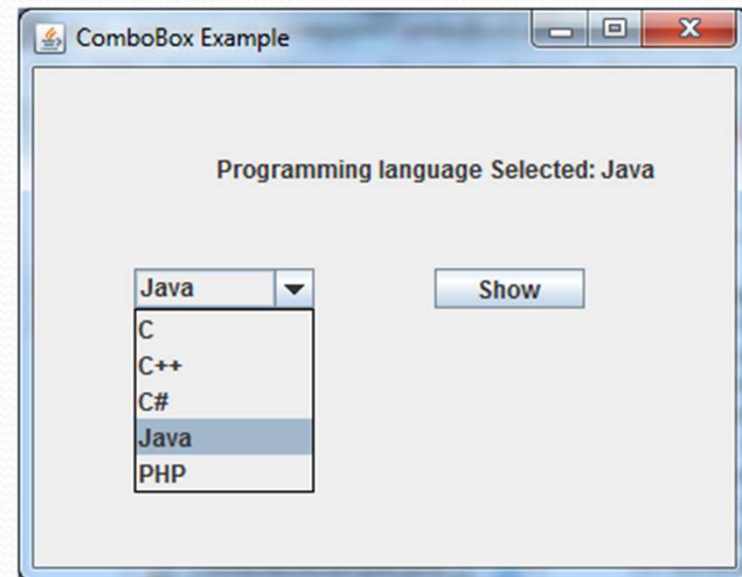
```
import javax.swing.*;
public class ComboBoxExample {
    JFrame f;
    ComboBoxExample(){
        f=new JFrame("ComboBox Example");
        String country[]={"India","Aus","U.S.A","England","Newzealand"};
        JComboBox cb=new JComboBox(country);
        cb.setBounds(50, 50,90,20);
        f.add(cb);
        f.setLayout(null);
        f.setSize(400,500);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new ComboBoxExample();
    }
}
```



```

import javax.swing.*;
import java.awt.event.*;
public class ComboBoxExample {
    JFrame f;
    ComboBoxExample(){
        f=new JFrame("ComboBox Example");
        final JLabel label = new JLabel();
        label.setHorizontalAlignment(JLabel.CENTER);
        label.setSize(400,100);
        JButton b=new JButton("Show");
        b.setBounds(200,100,75,20);
        String languages[]={"C","C++","C#","Java","PHP"};
        final JComboBox cb=new JComboBox(languages);
        cb.setBounds(50, 100,90,20);
        f.add(cb); f.add(label); f.add(b);
        f.setLayout(null);
        f.setSize(350,350);
        f.setVisible(true);
        b.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String data = "Programming language Selected: "
                    + cb.getItemAt(cb.getSelectedIndex());
                label.setText(data);
            }
        });
    }
    public static void main(String[] args) {
        new ComboBoxExample();
    }
}

```



Java JTable

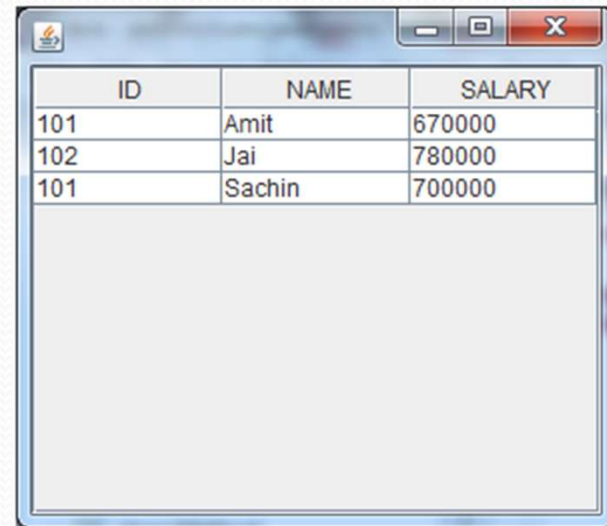
- The JTable class is used to display data in tabular form. It is composed of rows and columns.

Commonly used Constructors:

Constructor			Description
JTable()			Creates a table with empty cells.
JTable(Object[][] columns)	rows,	Object[]	Creates a table with the specified data.

Java JTable Example

```
import javax.swing.*;
public class TableExample {
    JFrame f;
    TableExample(){
        f=new JFrame();
        String data[][]={ {"101","Amit","670000"},
                           {"102","Jai","780000"},
                           {"101","Sachin","700000"}};
        String column[]={"ID","NAME","SALARY"};
        JTable jt=new JTable(data,column);
        jt.setBounds(30,40,200,300);
        JScrollPane sp=new JScrollPane(jt);
        f.add(sp);
        f.setSize(300,400);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new TableExample();
    }
}
```



ID	NAME	SALARY
101	Amit	670000
102	Jai	780000
101	Sachin	700000

Java JList

- The object of JList class represents a list of text items. The list of text items can be set up so that the user can choose either one item or multiple items. It inherits JComponent class.

Commonly used Constructors:

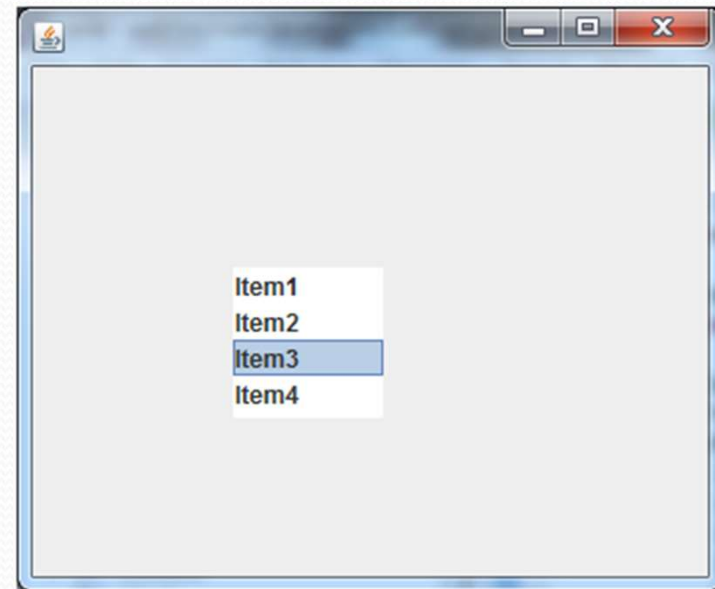
Constructor	Description
<code>JList()</code>	Creates a JList with an empty, read-only, model.
<code>JList(ary[] listData)</code>	Creates a JList that displays the elements in the specified array.
<code>JList(ListModel<ary> dataModel)</code>	Creates a JList that displays elements from the specified, non-null, model.

Commonly used Methods

Methods	Description
Void addListSelectionListener(ListSelectionListener listener)	It is used to add a listener to the list, to be notified each time a change to the selection occurs.
int getSelectedIndex()	It is used to return the smallest selected cell index.
ListModel getModel()	It is used to return the data model that holds a list of items displayed by the JList component.
void setListData(Object[] listData)	It is used to create a read-only ListModel from an array of objects.

Java JList Example

```
public class ListExample
{
    ListExample(){
        JFrame f= new JFrame();
        DefaultListModel<String> l1 = new DefaultListModel<>();
        l1.addElement("Item1");
        l1.addElement("Item2");
        l1.addElement("Item3");
        l1.addElement("Item4");
        JList<String> list = new JList<>(l1);
        list.setBounds(100,100, 75,75);
        f.add(list);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[])
    {
        new ListExample();
    }
}
```



Java 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.

Common Constructors of JOptionPane class

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

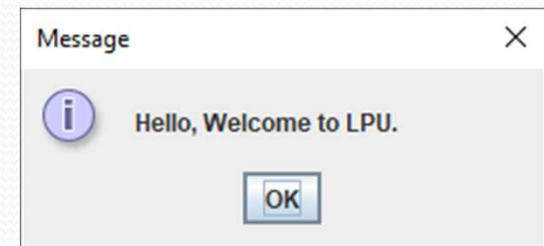
Common Methods of JOptionPane class

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.

Java JOptionPane Example: showMessageDialog()

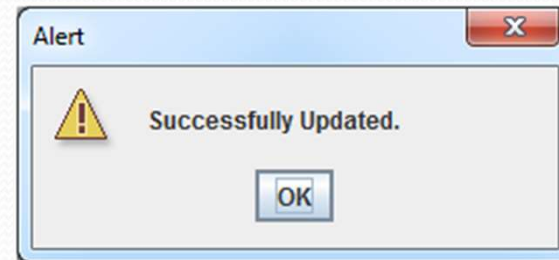
```
import javax.swing.*;

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



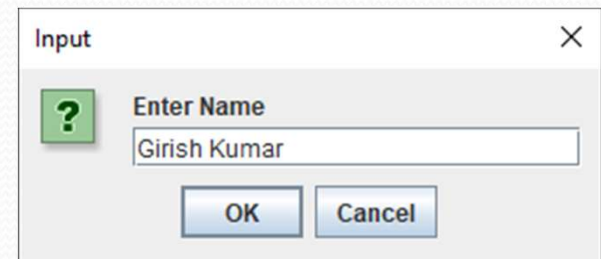
Java JOptionPane Example: showMessageDialog()

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



Java JOptionPane Example: showInputDialog()

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



Java JOptionPane Example: showConfirmDialog()

```
import javax.swing.*;
import java.awt.event.*;
public class OptionPaneExample extends WindowAdapter{
    JFrame f;
    OptionPaneExample(){
        f=new JFrame();
        f.addWindowListener(this);
        f.setSize(300, 300);
        f.setLayout(null);
        f.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        f.setVisible(true);
    }
    public void windowClosing(WindowEvent e) {
        int a=JOptionPane.showConfirmDialog(f,"Are you sure?");
        if(a==JOptionPane.YES_OPTION){
            f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
    }
    public static void main(String[] args) {
        new OptionPaneExample();
    }
}
```



Java JColorChooser

- The JColorChooser class is used to create a color chooser dialog box so that user can select any color. It inherits JComponent class.

Commonly used Constructors

Constructor	Description
JColorChooser()	It is used to create a color chooser panel with white color initially.
JColorChooser(color initialcolor)	It is used to create a color chooser panel with the specified color initially.

Commonly used Methods

Method	Description
<code>void addChooserPanel(AbstractColorChooserPanel panel)</code>	It is used to add a color chooser panel to the color chooser.
<code>static Color showDialog(Component c, String title, Color initialColor)</code>	It is used to show the color chooser dialog box.

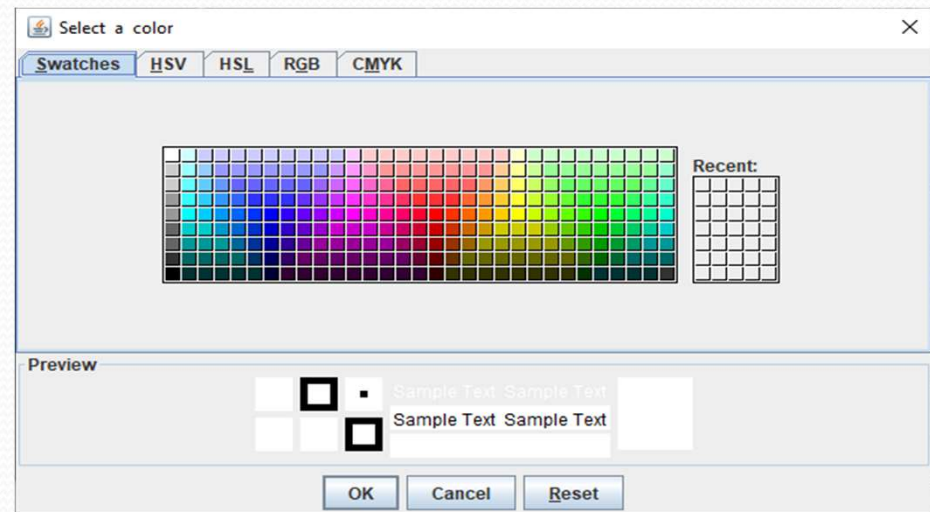
Java JColorChooser Example

```
import java.awt.Color;

import javax.swing.JColorChooser;

public class Main {
    public static void main(String[] args) {
        // Display a color chooser dialog
        Color color = JColorChooser.showDialog(null, "Select a color", null);

        // Check if user selected a color
        if (color == null) {
            System.out.println("we cancelled or closed the color chooser");
        } else {
            System.out.println("we selected color: " + color);
        }
    }
}
```



Java JProgressBar

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

Commonly used Constructors

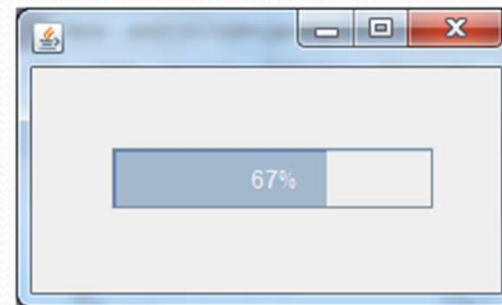
Constructor	Description
<code>JProgressBar()</code>	It is used to create a horizontal progress bar but no string text.
<code>JProgressBar(int min, int max)</code>	It is used to create a horizontal progress bar with the specified minimum and maximum value.
<code>JProgressBar(int orient)</code>	It is used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

Commonly used Methods

Method	Description
<code>void setStringPainted(boolean b)</code>	It is used to determine whether string should be displayed.
<code>void setString(String s)</code>	It is used to set value to the progress string.
<code>void setOrientation(int orientation)</code>	It is used to set the orientation, it may be either vertical or horizontal by using <code>SwingConstants.VERTICAL</code> and <code>SwingConstants.HORIZONTAL</code> constants.
<code>void setValue(int value)</code>	It is used to set the current value on the progress bar.

Java 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();
    }
}
```



Java 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
<code>public void setMinorTickSpacing(int n)</code>	is used to set the minor tick spacing to the slider.
<code>public void setMajorTickSpacing(int n)</code>	is used to set the major tick spacing to the slider.
<code>public void setPaintTicks(boolean b)</code>	is used to determine whether tick marks are painted.
<code>public void setPaintLabels(boolean b)</code>	is used to determine whether labels are painted.
<code>public void setPaintTracks(boolean b)</code>	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);
}

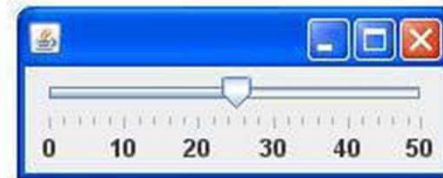
public static void main(String s[]) {
    SliderExample1 frame=new SliderExample1();
    frame.pack();
    frame.setVisible(true);
}
}
```



Java JSlider Example: painting ticks

```
import javax.swing.*;
public class SliderExample extends JFrame{
    public SliderExample() {
        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[]) {
        SliderExample frame=new SliderExample();
        frame.pack();
        frame.setVisible(true);
    }
}
```



```
import javax.swing.JFrame;
import javax.swing.JSlider;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;
import java.awt.FlowLayout;
```

```
class JSliderTest extends JFrame implements ChangeListener{
```

```
    JSlider slider1,slider2;
    JButton jb1,jb2;
```

```
    JSliderTest()
    {
        setTitle("JSlider");
        setLayout(new FlowLayout());
        setJSlider();
        setSize(700, 250);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
```

```
    private void setJSlider()
    {
        slider1 = new JSlider();
        slider1.addChangeListener(this);
        add(new JLabel("slider1"));
        add(slider1);
        add(jb1 = new JButton("DefaultValue"));

        slider2 = new JSlider(100,200);
        slider2.addChangeListener(this);
        add(new JLabel("slider2"));
        add(slider2);
        add(jb2 = new JButton("DefinedValue"));
    }
```

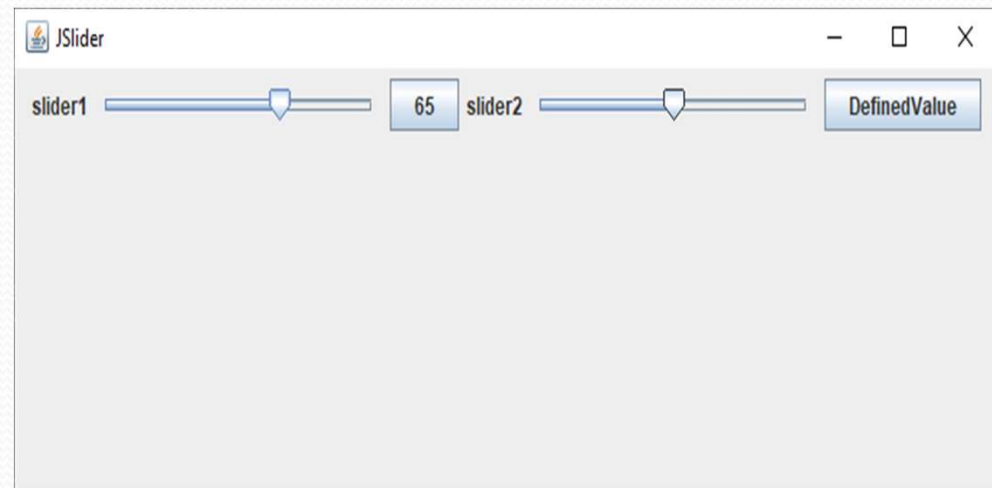
```
    public void stateChanged(ChangeEvent ev)
    {
        if(ev.getSource() == slider1)
        {
            jb1.setText(""+slider1.getValue());
        }
        if(ev.getSource() == slider2)
        {
            jb2.setText(""+slider2.getValue());
        }
    }
}
```

```
public class Main {

    public static void main(String[] args) {

        JSliderTest js = new JSliderTest();

    }
}
```





Continue . . .