Swing Components

1.4 MVC Architecture

Java Swing

Java Swing is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent & lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Difference between AWT and Swing

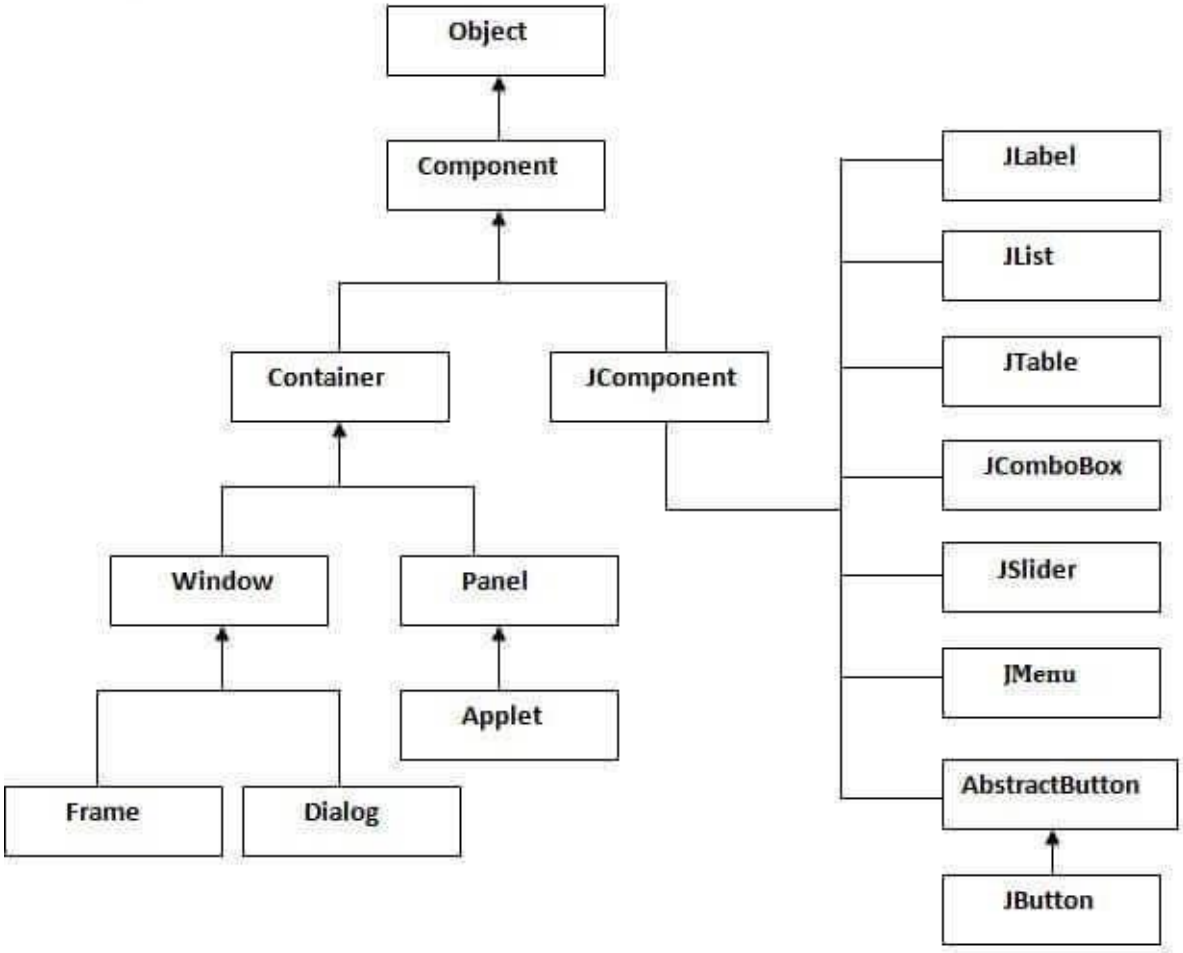
There are many differences between java awt and swing that are given below.

Sr. No	Java AWT	Java Swing
1)	AWT components are **platform-dependent**.	Java swing components are **platform-independent**.
2)	AWT components are **heavyweight**.	Swing components are **lightweight**.
3)	AWT **doesn't support pluggable look and feel**.	Swing **supports pluggable look and feel**.
4)	AWT provides **less components** than Swing.	Swing provides **more powerful components**such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5)	AWT **doesn't follows MVC**(Model View Controller) where **model** represents **data**, **view** represents **presentation** and **controller** acts as an **interface between model and view.**	Swing **follows MVC**.

What is JFC

The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.

Hierarchy of Java Swing classes



The methods of Component class are widely used in java swing that are given below.

Sr. No	Method	Description
1	public void add(Component c)	add a component on another component.
2	public void setSize(int width,int height)	sets size of the component.
3	public void setLayout(LayoutManager m)	sets the layout manager for the component.
4	public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

Swing Components

1. JApplet class

As we prefer Swing to AWT. Now we can use JApplet that can have all the controls of swing. The JApplet class extends the Applet class.

Example of EventHandling in JApplet :

```
import java.applet.*;
import javax.swing.*;
import java.awt.event.*;
public class EventJApplet extends JApplet implements ActionListener{
JButton b;
JTextField tf;
public void init(){

tf=new JTextField();
tf.setBounds(30,40,150,20);

b=new JButton("Click");
b.setBounds(80,150,70,40);

add(b);add(tf);
b.addActionListener(this);

setLayout(null);
}

public void actionPerformed(ActionEvent e){
tf.setText("Welcome");
}
}
```

myapplet.html

```
<html>
<body>
<applet code="EventJApplet.class" width="300" height="300">
</applet>
</body>
</html>
```

2. JLabelClass

The class **JLabel** can display either text, an image, or both. Label's contents are aligned by setting the vertical and horizontal alignment in its display area. By default, labels are vertically centered in their display area. Text-only labels are leading edge aligned, by default; image-only labels are horizontally centered, by default.

Class Constructors

Sr. No	Description
1	**JLabel()** Creates a JLabel instance with no image and with an empty string for the title.
2	**JLabel(Icon image)** Creates a JLabel instance with the specified image.
3	**JLabel(Icon image, int horizontalAlignment)** Creates a JLabel instance with the specified image and horizontal alignment.
4	**JLabel(String text)** Creates a JLabel instance with the specified text.
5	**JLabel(String text, Icon icon, int horizontalAlignment)** Creates a JLabel instance with the specified text, image, and horizontal alignment.
6	**JLabel(String text, int horizontalAlignment)** Creates a JLabel instance with the specified text and horizontal alignment.

JLabel Example

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

public class SwingControlDemo {
  private JFrame mainFrame;
  private JLabel headerLabel;
  private JLabel statusLabel;
  private JPanel controlPanel;

  public SwingControlDemo(){
    prepareGUI();
  }
  public static void main(String[] args){
    SwingControlDemo  swingControlDemo = new SwingControlDemo();
    swingControlDemo.showLabelDemo();
  }
  private void prepareGUI(){
    mainFrame = new JFrame("Java Swing Examples");
    mainFrame.setSize(400,400);
    mainFrame.setLayout(new GridLayout(3, 1));

    mainFrame.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent windowEvent){
        System.exit(0);
      }
    });
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("",JLabel.CENTER);
    statusLabel.setSize(350,100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
  }
  private void showLabelDemo(){
    headerLabel.setText("Control in action: JLabel");
    JLabel label  = new JLabel("", JLabel.CENTER);
    label.setText("Welcome to TutorialsPoint Swing Tutorial.");
    label.setOpaque(true);
    label.setBackground(Color.GRAY);
    label.setForeground(Color.WHITE);
```

```
    controlPanel.add(label);

    mainFrame.setVisible(true);
  }
}
```

3. JTextField Example

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

public class SwingControlDemo {
   private JFrame mainFrame;
   private JLabel headerLabel;
   private JLabel statusLabel;
   private JPanel controlPanel;

   public SwingControlDemo(){
      prepareGUI();
   }
   public static void main(String[] args){
      SwingControlDemo  swingControlDemo = new SwingControlDemo();
      swingControlDemo.showTextFieldDemo();
   }
   private void prepareGUI(){
      mainFrame = new JFrame("Java Swing Examples");
      mainFrame.setSize(400,400);
      mainFrame.setLayout(new GridLayout(3, 1));

      mainFrame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent){
            System.exit(0);
         }
      });
      headerLabel = new JLabel("", JLabel.CENTER);
      statusLabel = new JLabel("",JLabel.CENTER);
      statusLabel.setSize(350,100);

      controlPanel = new JPanel();
      controlPanel.setLayout(new FlowLayout());

      mainFrame.add(headerLabel);
      mainFrame.add(controlPanel);
      mainFrame.add(statusLabel);
      mainFrame.setVisible(true);
   }
   private void showTextFieldDemo(){
      headerLabel.setText("Control in action: JTextField");
```

```
      JLabel  namelabel= new JLabel("User ID: ", JLabel.RIGHT);
      JLabel  passwordLabel = new JLabel("Password: ",
JLabel.CENTER);
      final JTextField userText = new JTextField(6);
      final JPasswordField passwordText = new JPasswordField(6);

      JButton loginButton = new JButton("Login");
      loginButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            String data = "Username " + userText.getText();
            data += ", Password: " + new
String(passwordText.getPassword());
            statusLabel.setText(data);
         }
      });
      controlPanel.add(namelabel);
      controlPanel.add(userText);
      controlPanel.add(passwordLabel);
      controlPanel.add(passwordText);
      controlPanel.add(loginButton);
      mainFrame.setVisible(true);
   }
}
```

4. JComboBox Class

The class **JComboBox** is a component which combines a button or editable field and a drop-down list.

Class Constructors

Sr.No.	Constructor & Description
1	**JComboBox()** Creates a JComboBox with a default data model.
2	**JComboBox(ComboBoxModel aModel)** Creates a JComboBox that takes its items from an existing ComboBoxModel.
3	**JComboBox(Object[] items)** Creates a JComboBox that contains the elements in the specified array.

JComboBox Example

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

public class SwingControlDemo {
   private JFrame mainFrame;
   private JLabel headerLabel;
   private JLabel statusLabel;
   private JPanel controlPanel;

   public SwingControlDemo(){
      prepareGUI();
   }
   public static void main(String[] args){
      SwingControlDemo  swingControlDemo = new SwingControlDemo();
      swingControlDemo.showComboboxDemo();
   }
   private void prepareGUI(){
      mainFrame = new JFrame("Java Swing Examples");
      mainFrame.setSize(400,400);
      mainFrame.setLayout(new GridLayout(3, 1));

      mainFrame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent){
            System.exit(0);
         }
      });
      headerLabel = new JLabel("", JLabel.CENTER);
      statusLabel = new JLabel("",JLabel.CENTER);
      statusLabel.setSize(350,100);

      controlPanel = new JPanel();
      controlPanel.setLayout(new FlowLayout());
```

```
      mainFrame.add(headerLabel);
      mainFrame.add(controlPanel);
      mainFrame.add(statusLabel);
      mainFrame.setVisible(true);
   }
   private void showComboboxDemo(){
      headerLabel.setText("Control in action: JComboBox");
      final DefaultComboBoxModel fruitsName = new DefaultComboBoxModel();

      fruitsName.addElement("Apple");
      fruitsName.addElement("Grapes");
      fruitsName.addElement("Mango");
      fruitsName.addElement("Peer");

      final JComboBox fruitCombo = new JComboBox(fruitsName);
      fruitCombo.setSelectedIndex(0);

      JScrollPane fruitListScrollPane = new JScrollPane(fruitCombo);
      JButton showButton = new JButton("Show");

      showButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            String data = "";
            if (fruitCombo.getSelectedIndex() != -1) {
               data = "Fruits Selected: "
                  + fruitCombo.getItemAt
                  (fruitCombo.getSelectedIndex());
            }
            statusLabel.setText(data);
         }
      });
      controlPanel.add(fruitListScrollPane);
      controlPanel.add(showButton);
      mainFrame.setVisible(true);
   }
}
```

5. JRadioButton Class

The class **JRadioButton** is an implementation of a radio button - an item that can be selected or deselected, and which displays its state to the user.

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

public class SwingControlDemo {
   private JFrame mainFrame;
   private JLabel headerLabel;
   private JLabel statusLabel;
   private JPanel controlPanel;

   public SwingControlDemo(){
      prepareGUI();
   }
   public static void main(String[] args){
```

```
      SwingControlDemo  swingControlDemo = new SwingControlDemo();
      swingControlDemo.showRadioButtonDemo();
   }
   private void prepareGUI(){
      mainFrame = new JFrame("Java Swing Examples");
      mainFrame.setSize(400,400);
      mainFrame.setLayout(new GridLayout(3, 1));

      mainFrame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent){
            System.exit(0);
         }
      });
      headerLabel = new JLabel("", JLabel.CENTER);
      statusLabel = new JLabel("",JLabel.CENTER);
      statusLabel.setSize(350,100);

      controlPanel = new JPanel();
      controlPanel.setLayout(new FlowLayout());

      mainFrame.add(headerLabel);
      mainFrame.add(controlPanel);
      mainFrame.add(statusLabel);
      mainFrame.setVisible(true);
   }
   private void showRadioButtonDemo(){
      headerLabel.setText("Control in action: RadioButton");

      final JRadioButton radApple = new JRadioButton("Apple", true);
      final JRadioButton radMango = new JRadioButton("Mango");
      final JRadioButton radPeer = new JRadioButton("Peer");

      radApple.setMnemonic(KeyEvent.VK_C);
      radMango.setMnemonic(KeyEvent.VK_M);
      radPeer.setMnemonic(KeyEvent.VK_P);

      radApple.addItemListener(new ItemListener() {
         public void itemStateChanged(ItemEvent e) {
            statusLabel.setText("Apple RadioButton: "
               + (e.getStateChange()==1?"checked":"unchecked"));
         }
      });
      radMango.addItemListener(new ItemListener() {
         public void itemStateChanged(ItemEvent e) {
            statusLabel.setText("Mango RadioButton: "
               + (e.getStateChange()==1?"checked":"unchecked"));
         }
      });
      radPeer.addItemListener(new ItemListener() {
         public void itemStateChanged(ItemEvent e) {
            statusLabel.setText("Peer RadioButton: "
               + (e.getStateChange()==1?"checked":"unchecked"));
         }
      });

      //Group the radio buttons.
      ButtonGroup group = new ButtonGroup();
```

```
        group.add(radApple);
        group.add(radMango);
        group.add(radPeer);

        controlPanel.add(radApple);
        controlPanel.add(radMango);
        controlPanel.add(radPeer);

        mainFrame.setVisible(true);
    }
}
```

6. Java JTabbedPane

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.

Commonly used Constructors:

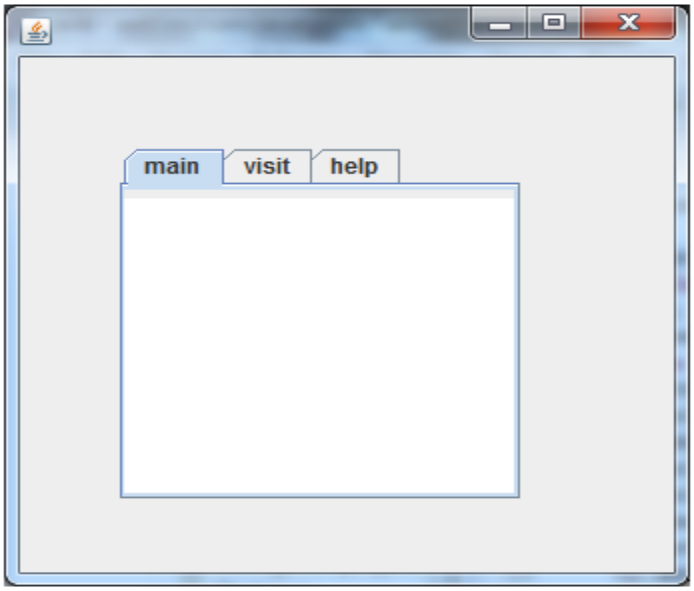
Constructor	Description
JTabbedPane()	Creates an empty TabbedPane with a default tab placement of JTabbedPane.Top.
JTabbedPane(int tabPlacement)	Creates an empty TabbedPane with a specified tab placement.
JTabbedPane(int tabPlacement, int tabLayoutPolicy)	Creates an empty TabbedPane with a specified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

JTabbedPane Example

```
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();
}}
```

Output :



7. JScrollPane

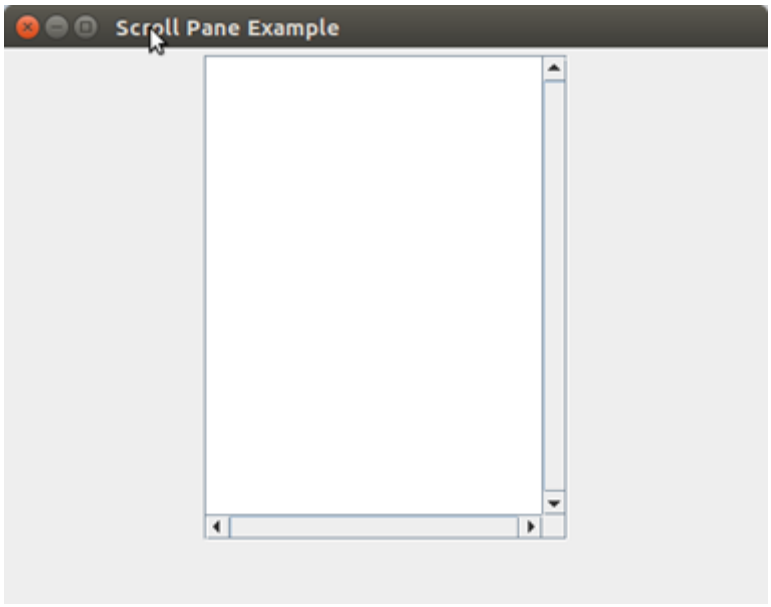
A JscrollPane is used to make scrollable view of a component. When screen size is limited, we use a scroll pane to display a large component or a component whose size can change dynamically.

JScrollPane Example

```
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JtextArea;

public class JScrollPaneExample {
    private static final long serialVersionUID = 1L;

    private static void createAndShowGUI() {

        // Create and set up the window.
        final JFrame frame = new JFrame("Scroll Pane Example");

        // Display the window.
        frame.setSize(500, 500);
```

```
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // set flow layout for the frame
        frame.getContentPane().setLayout(new FlowLayout());

        JTextArea textArea = new JTextArea(20, 20);
        JScrollPane scrollableTextArea = new JScrollPane(textArea);

        scrollableTextArea.setHorizontalScrollBarPolicy(JScrollPane.HORIZO
NTAL_SCROLLBAR_ALWAYS);
        scrollableTextArea.setVerticalScrollBarPolicy(JScrollPane.VERTICAL
_SCROLLBAR_ALWAYS);

        frame.getContentPane().add(scrollableTextArea);
    }
    public static void main(String[] args) {


        javax.swing.SwingUtilities.invokeLater(new Runnable() {

            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

Output:



8. JTree

The JTree class is used to display the tree structured data or hierarchical data. JTree is a complex component. It has a 'root node' at the top most which is a parent for all nodes in the tree. It inherits JComponent class.

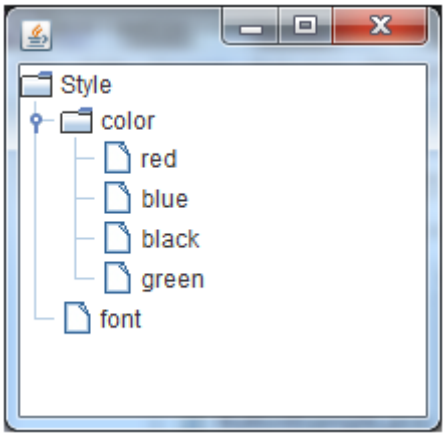
Commonly used Constructors:

Constructor	Description
JTree()	Creates a JTree with a sample model.
JTree(Object[] value)	Creates a JTree with every element of the specified array as the child of a new root node.
JTree(TreeNode root)	Creates a JTree with the specified TreeNode as its root, which displays the root node.

JTree Example

```
import javax.swing.*;
import javax.swing.tree.DefaultMutableTreeNode;
public class TreeExample {
JFrame f;
TreeExample(){
   f=new JFrame();
   DefaultMutableTreeNode style=new DefaultMutableTreeNode("Style");

   DefaultMutableTreeNode color=new DefaultMutableTreeNode("color");

   DefaultMutableTreeNode font=new DefaultMutableTreeNode("font");
   style.add(color);
   style.add(font);
   DefaultMutableTreeNode red=new DefaultMutableTreeNode("red");
   DefaultMutableTreeNode blue=new DefaultMutableTreeNode("blue");
   DefaultMutableTreeNode black=new DefaultMutableTreeNode("black")
;
   DefaultMutableTreeNode green=new DefaultMutableTreeNode("green"
);
   color.add(red); color.add(blue); color.add(black); color.add(green);

   JTree jt=new JTree(style);
   f.add(jt);
   f.setSize(200,200);
   f.setVisible(true);
}
public static void main(String[] args) {
   new TreeExample();
}}
```

Output:



9. JProgressBar

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

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 SwingConstants.VERTICAL and SwingConstants.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:

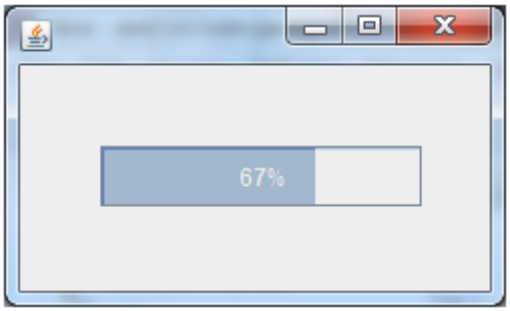
Method	Description
void setStringPainted(boolean b)	It is used to determine whether string should be displayed.
void setString(String s)	It is used to set value to the progress string.
void setOrientation(int orientation)	It is used to set the orientation, it may be either vertical or horizontal by using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



10. JTable

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

Commonly used Constructors:

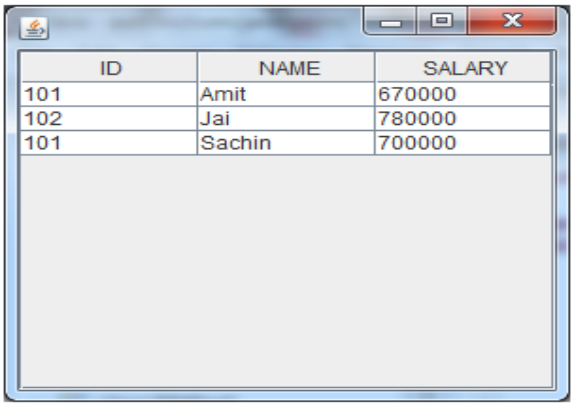
Constructor	Description
JTable()	Creates a table with empty cells.
JTable(Object[][] rows, Object[] columns)	Creates a table with the specified data.

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();
}
}
```

Output:



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 seperate 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.