Swing

Swing is a framework or API that is used to create GUI (or) window-based applications.It is an advanced version of AWT (Abstract Window Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and 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

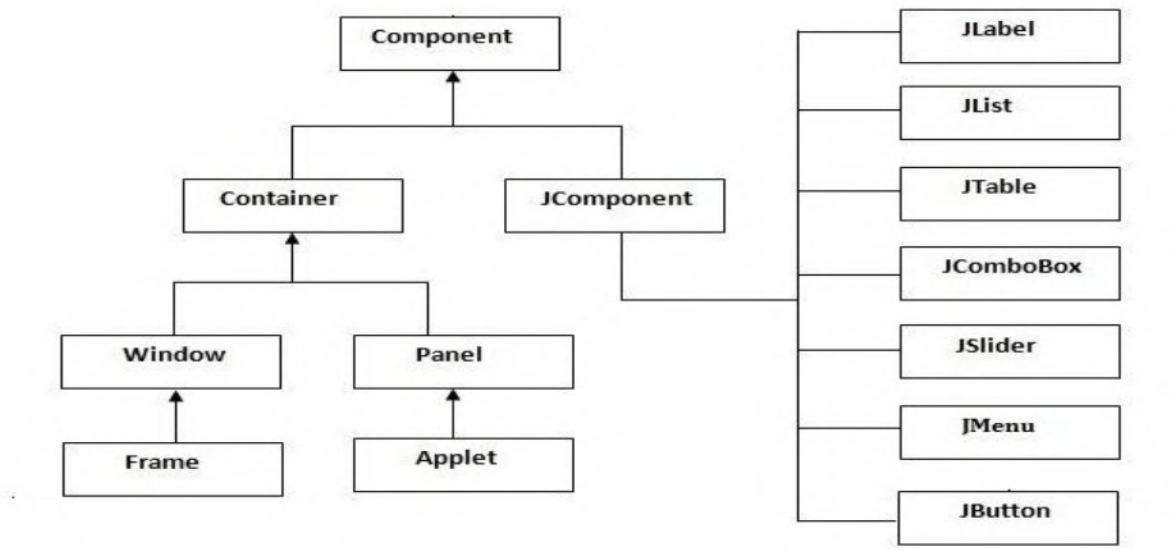
There are many differences between java awt and swing that are given below.

No.	AWT	Swing
1)	AWT components are **platform-dependent**.	Swing components are **platform-independent**.
2)	AWT components are **heavyweight**.	Swing components are **lightweight**.
3)	AWT provides **less components** than Swing.	Swing provides **more powerful components** such as tables, lists, scrollpanes, colorchooser and` etc.
4)	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**.

Commonly used Methods of Component class

Method	Description
add(Component c)	inserts a component on this component.
setSize(int width,int height)	sets the size (width and height) of the component.
setLayout(LayoutManager m)	defines the layout manager for the component.
setVisible(boolean status)	changes the visibility of the component, by default false.
setTitle(String text)	Sets the title for component

Swing Hierarchy



To create simple swing example, you need a frame.

- In swing, we use **JFrame class** to create a frame.

There are two ways to create a frame in swing.

- By extending JFrame class (inheritance)

  Ex:

  ```
  class Example extends JFrame
  {
    ……..
  }
  ```

- By creating the object of JFrame class (association)

  Ex:

  ```
  class Example
  {
  JFrame obj=new JFrame();
    ……..
  }
  ```

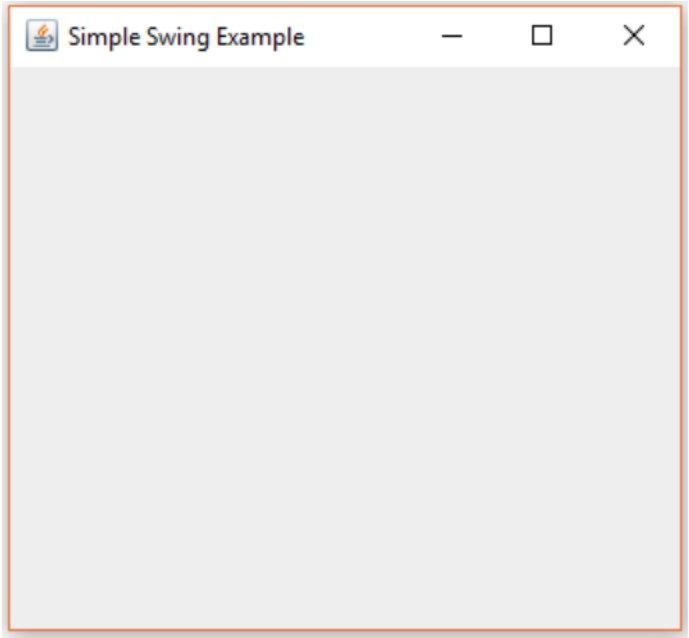
A Simple Swing Example

We can write the code of swing inside the main() or constructor.

In Main() Method:

SwingExample.java

```
import javax.swing.*;
public class SwingExample
{
public static void main(String[] args)
{
   JFrame f=new JFrame("Simple Swing Example");
   f.setSize(300,300);
   f.setLayout(null);
   f.setVisible(true);
}
}
```



(OR)

In Constructor()

```
import javax.swing.*;
class SwingExample extends JFrame
{
SwingExample()
{
setSize(300,300);//frame size 300 width and 300 height
setLayout(null);//no layout manager
setVisible(true);//now frame will be visible, by default not visible
setTitle("SwingExample ");//Set Title
}
public static void main(String args[]){
SwingExample f=new SwingExample();
}
}
```

Components of Swing:

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.

Syntax:

```
JButton b=new JButton("Text");
(Or)
JButton b1,b2;
b1=new JButton("Text");
b.setBounds(50,100,80,30);
```

JLabel:

The 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.

Syntax:

```
JLabel l1=new JLabel("Text");
(or)
JLabel l1,l2;
l1=new JLabel("Text");
```

JTextField:

The JTextField class is a text component that allows the editing of a single line text.

Syntax:

```
JTextField t1=new JTextField("Text");
(or)
JTextField t1,t2;
t1=new JTextField("Text");
```

JTextArea :

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

Syntax:

```
JTextArea t1=new JTextArea("Text");
(or)
JTextArea t1,t2;
t1=new JTextArea("Text");
```

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".

Syntax:

```
JCheckBox c1=new JCheckBox("Text");
(or)
JCheckBox c1,c2;
c1=new JCheckBox("Text");
```

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 one or more items from list of items.

Syntax:

```
DefaultListModel<String> l1 = new DefaultListModel<>();
l1.addElement("Item1");
l1.addElement("Item2");
l1.addElement("Item3");
l1.addElement("Item4");
JList list = new JList<>(l1);
```

JPasswordField:

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

Syntax:

```
JPasswordField pwd = new JPasswordField();
pwd.setBounds(100,50,80,30);
```

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.

Syntax:

```
ButtonGroup bg=new ButtonGroup();
JRadioButton r1=new JRadioButton("Male");
JRadioButton r2=new JRadioButton("Female");
bg.add(r1);bg.add(r2);
```

JComboBox:

The JComboBox class is used to show popup menu of items. Item selected by user is shown on the top of a menu.(like Choice class in AWT)

Syntax:

```
String country[]={"India","Aus","U.S.A","England","Newzealand"};
JComboBox cb=new JComboBox(country);
cb.setBounds(50, 50,90,20);
```

JTable:

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

Syntax:

```
String data[][]=    { {“521",“Madhu",“43400"}, {“512",“Hari",“54500"},
                          {“509",“Ganesh","70000"}};
String column[]={"ID","NAME","SALARY"};
JTable jt=new JTable(data,column);
jt.setBounds(30,40,200,300);
```

JPanel:

The JPanel is a simplest container class. It provides space in which an application can attach any other component.

Syntax:

```
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);
panel.add(b1);
```

JDialog:

The JDialog control represents a top level window with a border and a title used to take some form of input from the user.

- Unlike JFrame, it doesn't have maximize and minimize buttons.

Syntax:

```
JFrame f= new JFrame();
JDialog d=new JDialog(f , "Dialog", true);
JButton b = new JButton ("OK");
d.add(b);
```

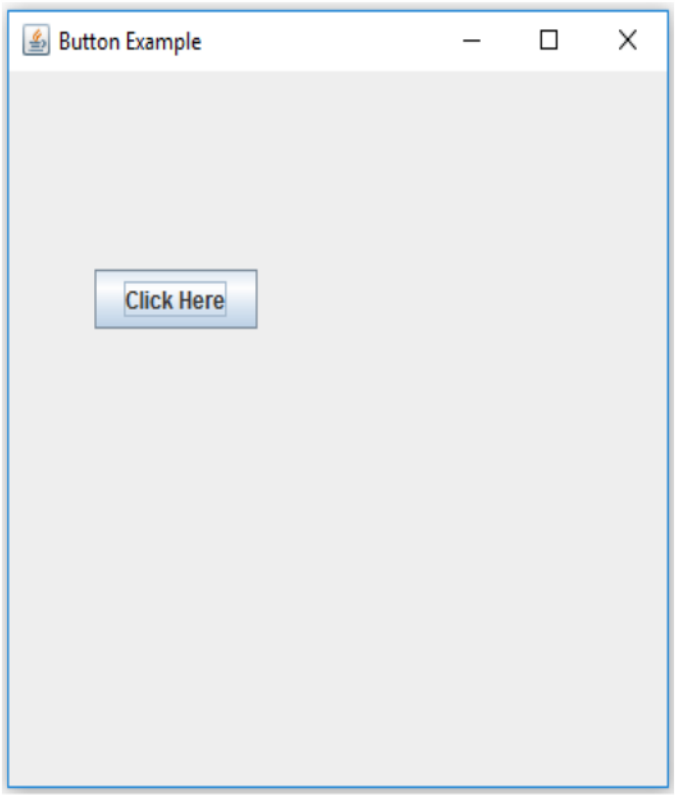
Example: An example for **JButton** Component in swing.

JButtonExample.java

```
import javax.swing.*;
public class JButtonExample
{
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);
}
}
```

Execution:

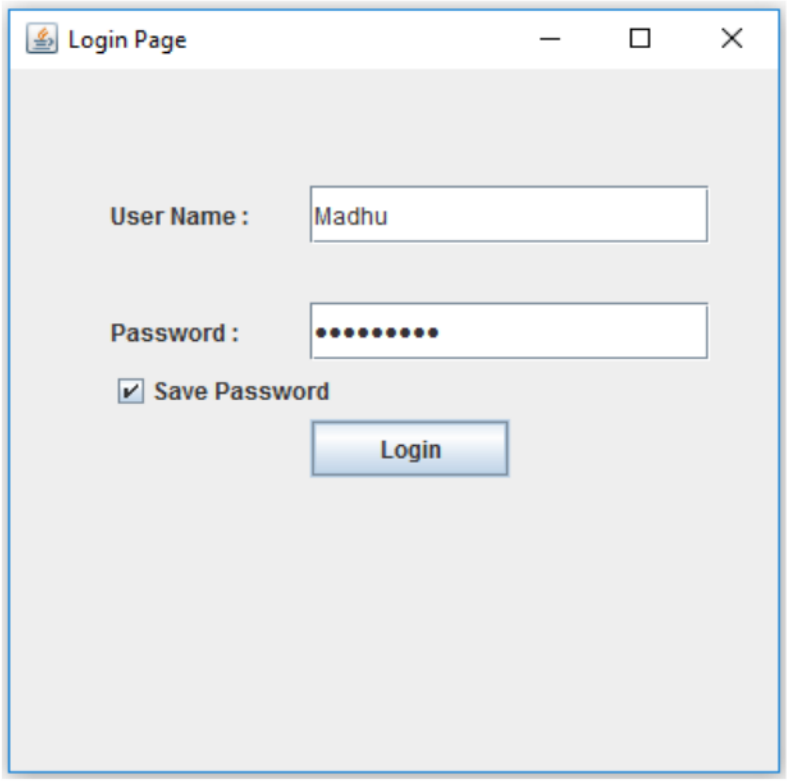
```
D:/>javac JButtonExample.java

D:/>java JButtonExample
```



Example: An example for Login page which includes **JLabel,JTextField,JPasswordField, JCheckBox and JButton** Components.

LoginExample.java

```
import javax.swing.*;
class LoginExample extends JFrame
{
public static void main(String args[])
{
   JLabel l1,l2;
   JTextField t1;
   JPasswordField p1;
   JCheckBox cb;
   JButton b;
   JFrame f=new JFrame("Login Page");
   l1=new JLabel("User Name :");
   l1.setBounds(50,60,100,30);
   t1=new JTextField("");
   t1.setBounds(150,60, 200,30);
   l2=new JLabel("Password :");
   l2.setBounds(50,120,100,30);
   p1=new JPasswordField("");
   p1.setBounds(150,120, 200,30);
   cb=new JCheckBox("Save Password");
   cb.setBounds(50,150,200,30);
   b=new JButton("Login");
   b.setBounds(150,180,100,30);
   f.add(l1);f.add(l2);
   f.add(t1);f.add(p1);
   f.add(cb);f.add(b);
   f.setSize(400,400);
   f.setLayout(null);
   f.setVisible(true);
}
}
```



Execution:

```
D:/>javac LoginExample.java
D:/>java LoginExample
```

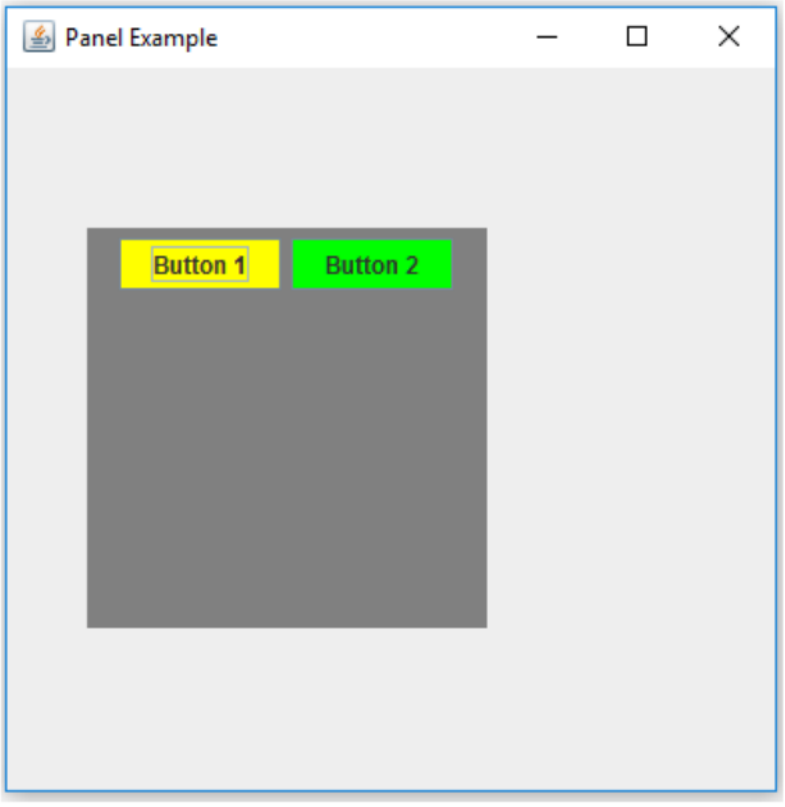
Example: An example for **JPanel** Class component of swing

JPanelExample.java

```
import java.awt.*;
import javax.swing.*;
public class JPanelExample
{
  public static void main(String args[])
      {
      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);
      }
}
```

Execution:

```
D:/> javac JPanelExample.java
D:/> java JPanelExample
```



Example: An example for **JRadioButton** Class component of swing

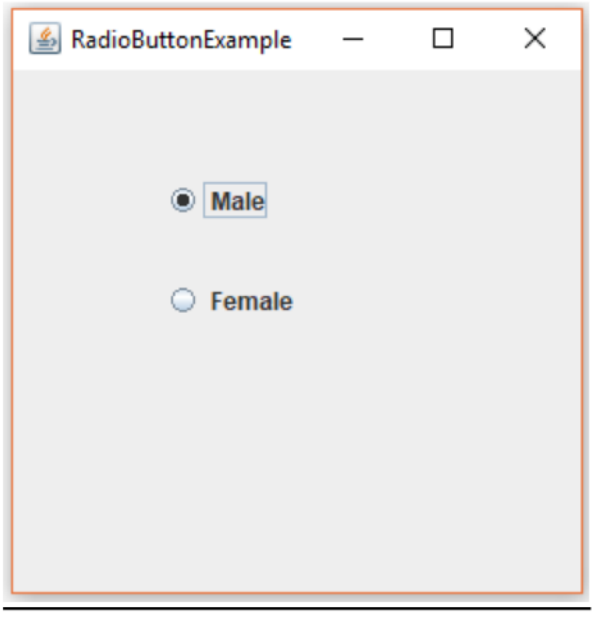
JRadioButtonExample.java

```
import javax.swing.*;
public class JRadioButtonExample
{
public static void main(String[] args)
{
JFrame f=new JFrame();
JRadioButton r1=new JRadioButton(" Male");
JRadioButton r2=new JRadioButton(" 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);
f.setTitle("RadioButtonExample");
}
}
```

Output:

Javac JRadioButtonExample.java

Java JRadioButtonExample



Example: An example for **JList** Class component of swing

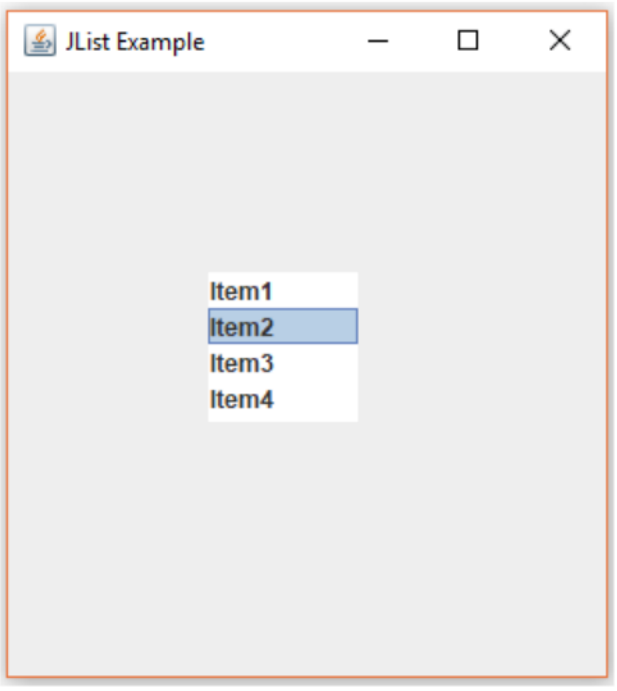
JListExample.java

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

Output:

Javac JListExample.java

Java JListExample



Example: An example for **JTable** Class component of swing

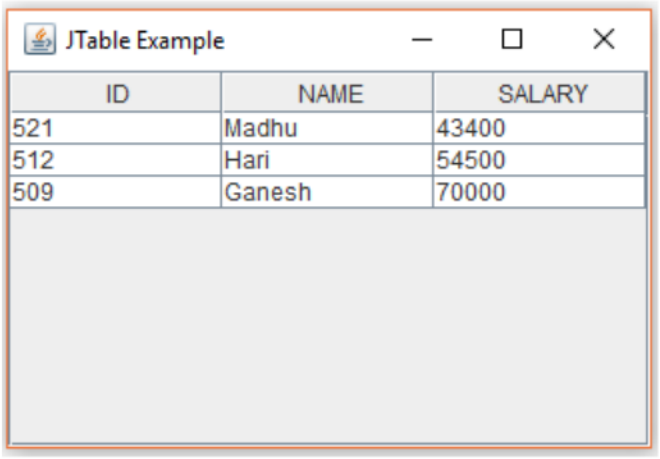
JTableExample.java

```
import javax.swing.*;
public class JTableExample
{
public static void main(String[] args)
{
   JFrame f=new JFrame();
   String data[][]={{"521","Madhu","43400"},
                    {"512","Hari","54500"},
                    {"509","Ganesh","70000"}};
   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);
   f.setTitle("JTable Example");
}
}
```

Output:

Javac JTableExample.java

Java JTableExample



Example: An example for **JOptionPane** Class component of swing

JOptionPaneExample.java

```
import javax.swing.*;
public class JOptionPaneExample
{
public static void main(String[] args)
{
   JFrame f=new JFrame();
   JOptionPane.showMessageDialog(f,"Hello, Welcome to Swing.");
}
}
```

Output:

Javac JOptionPaneExample.java

Java JOptionPaneExample

Example: An example for **JTextArea** Class component of swing

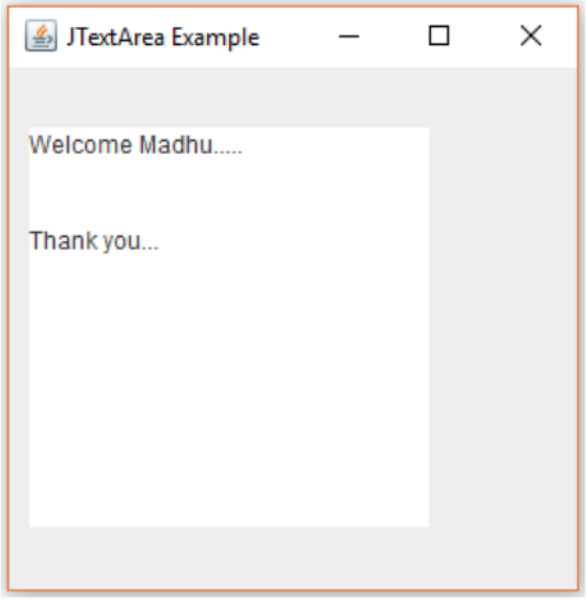
JTextAreaExample.java

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

Output:

Javac JTextAreaExample.java

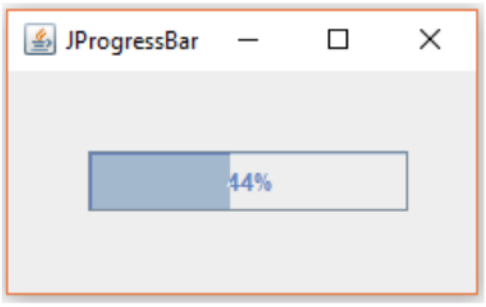
Java JTextAreaExample



Example: An example for **JProgressBar** Class component of swing

JProgressBarExample.java

```
import javax.swing.*;
public class JProgressBarExample extends JFrame
{
JProgressBar jb;
int i=0,num=0;
JProgressBarExample()
{
jb=new JProgressBar(0,2000);
jb.setBounds(40,40,160,30);
jb.setValue(0);
jb.setStringPainted(true);
add(jb);
setSize(250,150);
setLayout(null);
setTitle("JProgressBar");
}
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)
{
   JProgressBarExample m=new JProgressBarExample();
   m.setVisible(true);
   m.iterate();
}
}
```



Output:

Javac JProgressBarExample.java

Java JProgressBarExample

Applet

An **applet** is a Java program that runs in a Web browser. (or) Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

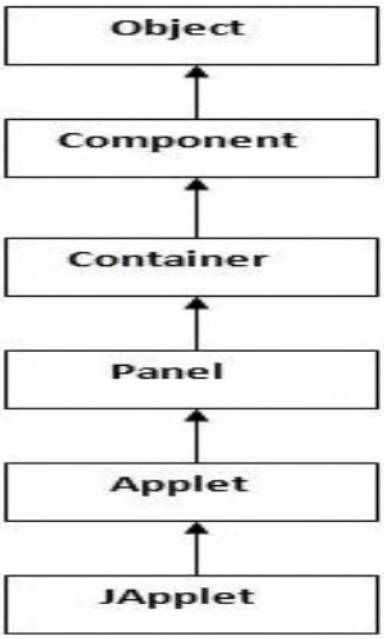
Any applet in Java is a class that extends the **java.applet.Applet** class.

Advantage of Applet

There are many advantages of applet. They are as follows:

- It works at client side so less response time.
- Secured
- It can be executed by browsers running under many platforms, including Linux, Windows, Mac Os etc.

Hierarchy of Applet :



As displayed in the diagram, Applet class extends Panel. Panel class extends Container, which is the subclass of Component. Where Object class is base class for all the classes in java.

JApplet class is extension of Applet class.

Lifecycle of Applet:

There are 5 lifecycle methods of Applet, Those are

public void init(): is used to initialized the Applet. It is invoked only once.

public void start(): is invoked after the init() method or browser is maximized. It is used to start the Applet.

public void paint(Graphics g): is invoked immediately after the start() method, and this method helps to create Applet's GUI such as a colored background, drawing and writing.

public void stop(): is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.

public void destroy(): is used to destroy the Applet. It is invoked only once.

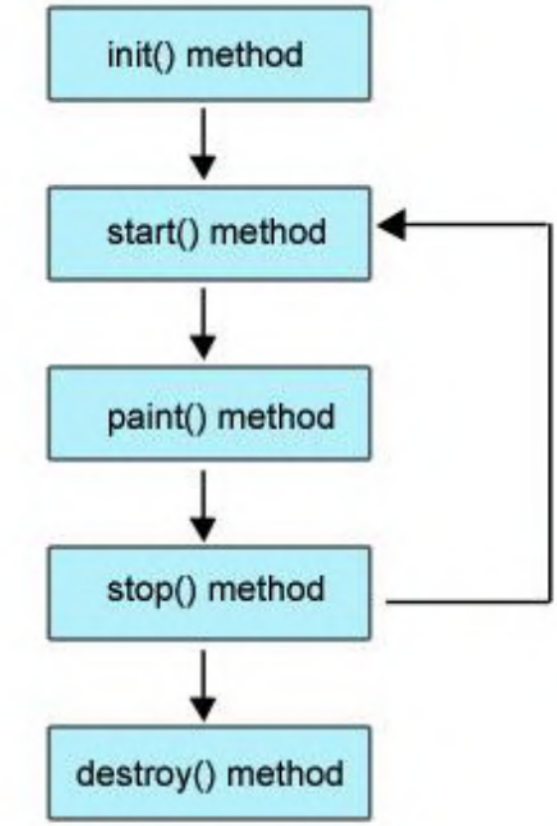


Figure: Life cycle of Applet

Remember:

java.applet.Applet class provides 4 methods (**init,start,stop & destroy**) and **java.awt.Graphics** class provides 1 method (**paint**) to create Applet.

Simple example of Applet:

- To execute an Applet, First Create an applet and compile it just like a simple java program.

First.java

```
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet
{
public void paint(Graphics g){
g.drawString("Welcome to Applet",50,150);
}
}
```

Compile:

D:\> javac First.java

After successful compilation, we get **First.class** file.

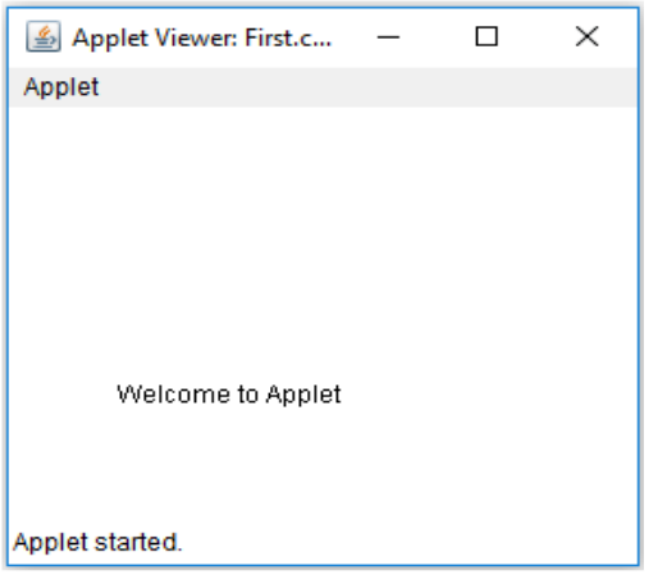
- After that create an html file and place the applet code in html file.

First.html

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

Execute:

D:\> appletviewer First.html



Displaying Graphics in Applet:

- **java.awt.Graphics** class provides many methods for graphics programming.

The Commonly used methods of Graphics class:

- **drawString(String str, int x, int y):** is used to draw the specified string.
- **drawRect(int x, int y, int width, int height):** draws a rectangle with the specified width and height.
- **fillRect(int x, int y, int width, int height):** is used to fill rectangle with the default color and specified width and height.
- **drawOval(int x, int y, int width, int height):** is used to draw oval with the specified width and height.
- **fillOval(int x, int y, int width, int height):** is used to fill oval with the default color and specified width and height.
- **drawLine(int x1, int y1, int x2, int y2):** is used to draw line between the points(x1, y1) and (x2, y2).
- **setColor(Color c):** is used to set the graphics current color to the specified color.
- **setFont(Font font):** is used to set the graphics current font to the specified font.

Example: GraphicsDemo.java

```
import java.applet.Applet;
import java.awt.*;
public class GraphicsDemo extends Applet
{
public void paint(Graphics g)
{
g.setColor(Color.red);
g.drawString("Welcome",50, 50);
g.drawLine(20,30,20,300);
g.drawRect(70,100,30,30);
g.fillRect(170,100,30,30);
g.drawOval(70,200,30,30);
g.setColor(Color.pink);
g.fillOval(170,200,30,30);
}
}
```

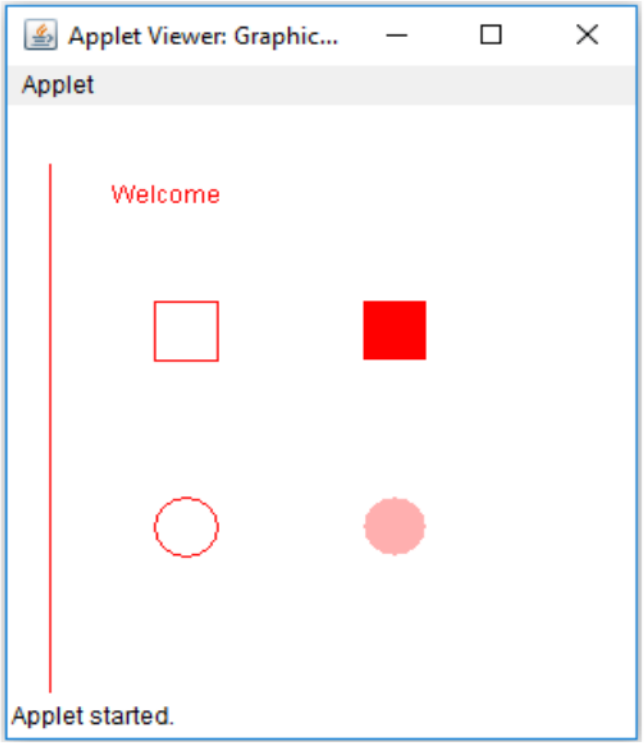
GraphicsDemo.html

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

Execution:

D:\> javac GraphicsDemo.java

D:\> appletviewer GraphicsDemo.html



Components of Applet:

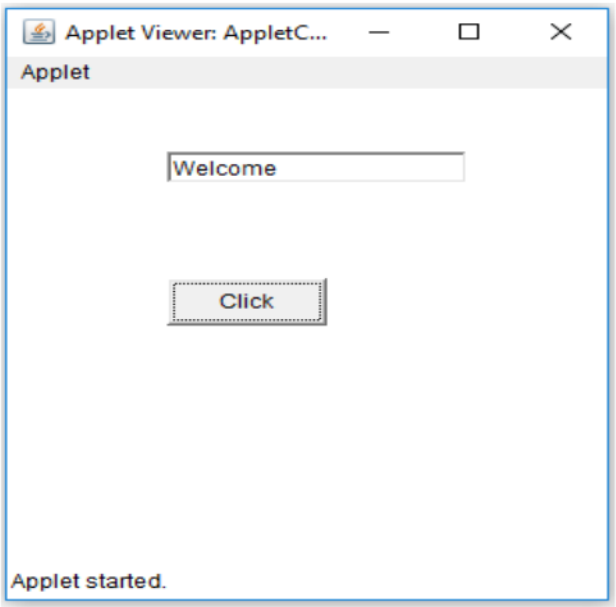
- The components of **AWT** are the components of **Applet**,i.e we can use AWT components (Button,TextField,Checkbox, TextArea,Choice & etc....) in applet.

- As we perform **event handling** in AWT or Swing, we can perform it in applet also.

Let's see the simple example of components and event handling in applet that prints a message by click on the button.

Example: AppletComponents.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class AppletComponents extends Applet implements ActionListener{
Button b;
TextField tf;
public void init(){
tf=new TextField();
tf.setBounds(80,40,150,20);
b=new Button("Click");
b.setBounds(80,120,80,30);
add(b);add(tf);
b.addActionListener(this);
setLayout(null);
}
public void actionPerformed(ActionEvent e)
{
 tf.setText("Welcome");
}
}
```



AppletComponents.html

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

Execution:

```
D:\>javac AppletComponents.java
D:\>appletviewer AppletComponents.html
```

JApplet Class:

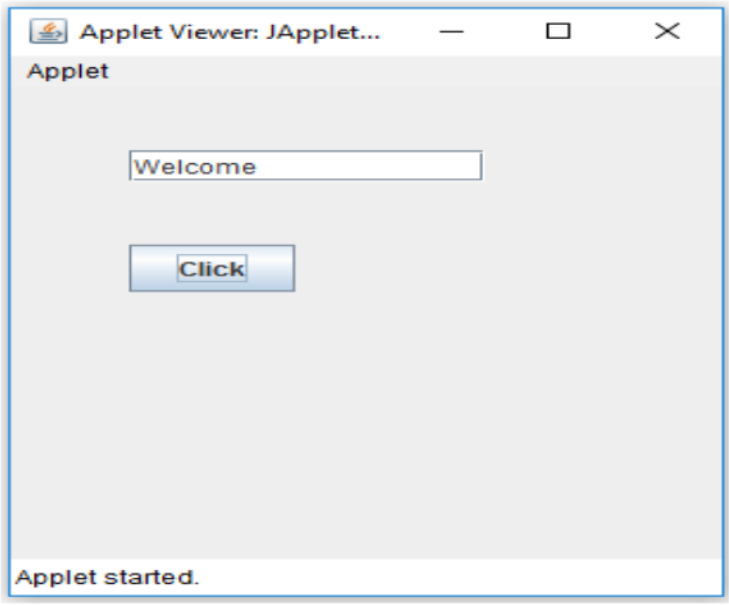
As we prefer Swing to AWT. Now we can use JApplet that can have all the controls of swing.

- The JApplet class extends the Applet class.

The components of **swing** are the components of **JApplet**,i.e we can use swing components (JButton,JTextField,JCheckBox, JTextArea,JList & etc....) in JApplet.

Example: JAppletComponents.java

```
import java.applet.*;
import javax.swing.*;
import java.awt.event.*;
public class JAppletComponents extends JApplet implements ActionListener
{
JButton b;
JTextField tf;
public void init(){
tf=new JTextField();
tf.setBounds(50,40,150,20);
b=new JButton("Click");
b.setBounds(50,100,70,30);
add(b);add(tf);
b.addActionListener(this);
setLayout(null);
}
public void actionPerformed(ActionEvent e){
tf.setText("Welcome");
}
}
```



JAppletComponents.html

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

Execution:

```
D:\>javac JAppletComponents.java
D:\>appletviewer JAppletComponents.html
```