19CS391 – Programming with Java (Embedded Project)

Unit IV EVENT DRIVEN PROGRAMMING 13

Introduction to Applet - Applet life cycle - Basics of event handling - The AWT event hierarchy - Mouse events – Adapter classes. Introducing to Swing - Creating and positioning a frame - Displaying information in a Component - Swing Components: Text Input - Choice components - Menus - Dialog boxes. JavaFX basic concepts – Using buttons and events – RadioButton – CheckBox – ComboBox - working with JDBC

java.awt

- Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.

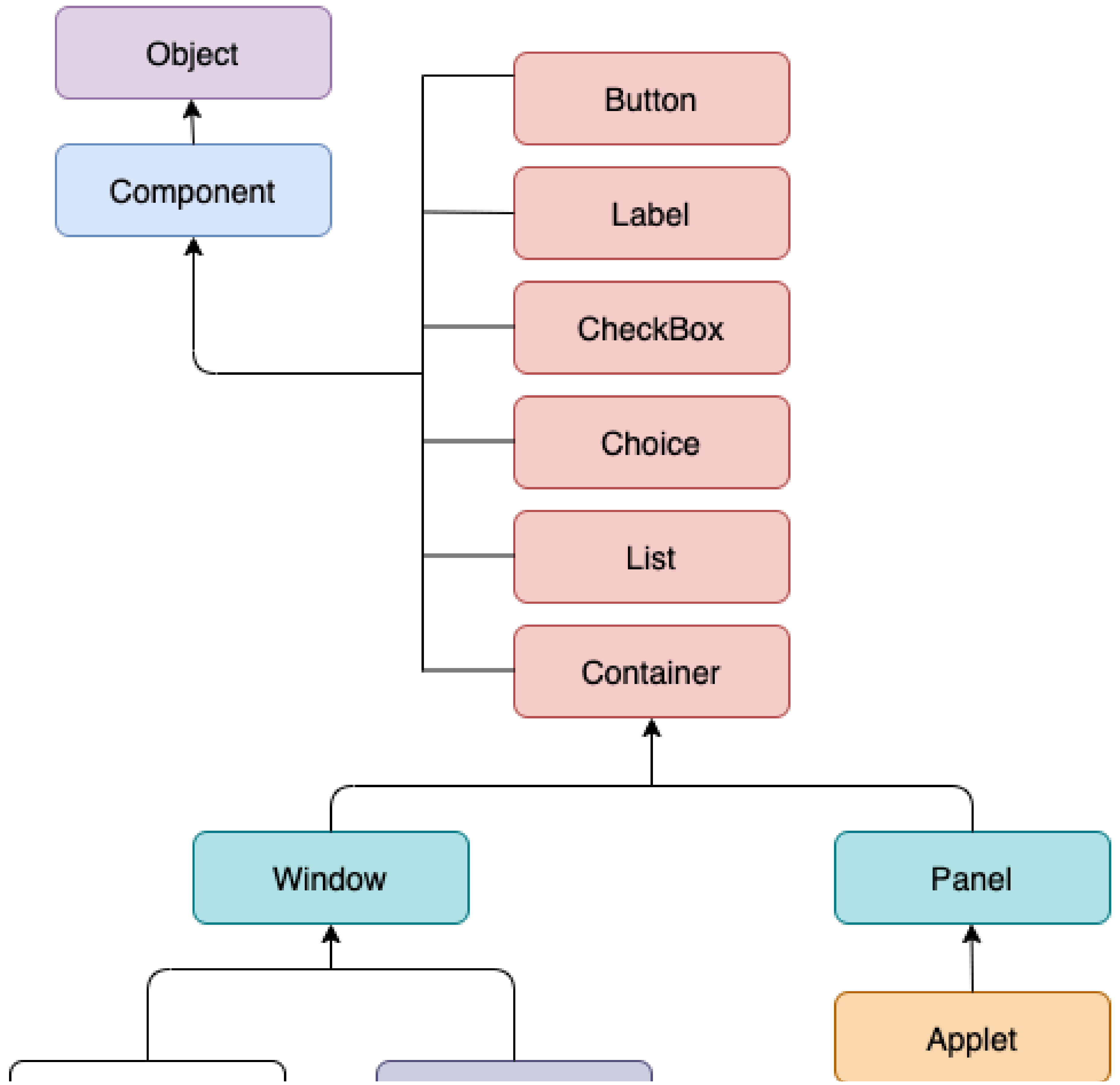
- Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system.

- AWT is heavyweight i.e. its components are using the resources of OS.

- The java.awt package
  - provides **classes for AWT API such as Button, TextField, Label, TextArea, RadioButton, CheckBox, Choice, List** etc.

Java AWT Hierarchy

The hierarchy of Java AWT classes are given below.



Java AWT Hierarchy

Component Class

- The component class stands **at the top of the AWT hierarchy, is an abstract class that contains all the properties of the component visible on the screen.**

Container

- The Container is a **component in AWT that can contain another components like buttons, textfields, labels etc.**
- The **classes that extend Container class are known as container such as Frame, Dialog and Panel.**

Window

  - The window is the container that has no borders and menu bars.
  - **Use frame, dialog or another window for creating a window.**

Panel

- The Panel is the **container that doesn't contain title bar and menu bars.**
- It can have other **components like button, textfield etc..**

Frame

- The Frame is the **container that contain title bar and can have menu bars.**
- **It can have other components like button, textfield etc..**

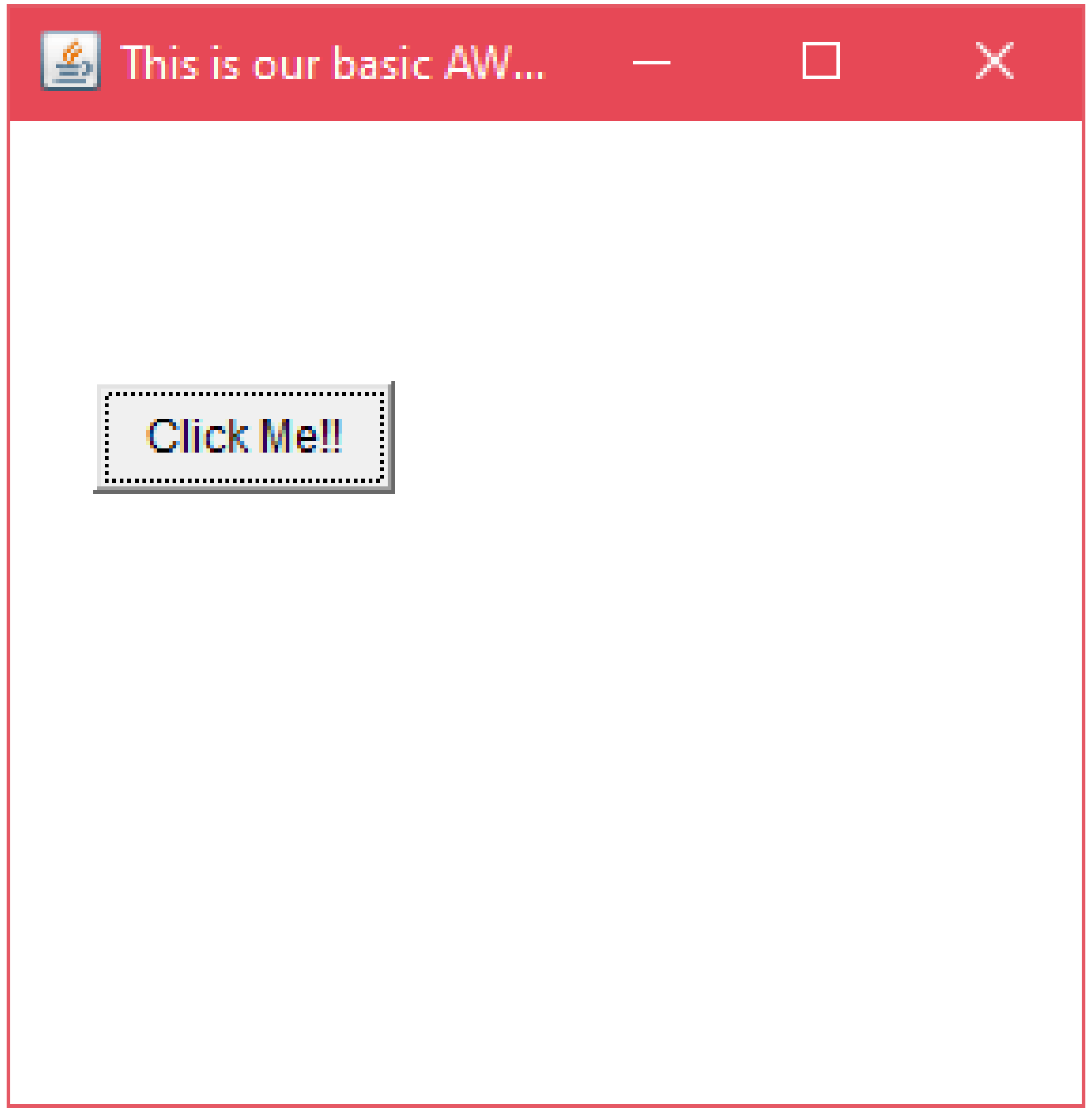
Methods of Component class

Method	Description
public void add(Component c)	inserts a component on this component.
public void setSize(int width,int height)	sets the size (width and height) of the component
public void setLayout(LayoutManager m)	defines the layout manager for the component.
public void setVisible(boolean status)	changes the visibility of the component, by default false.
public void setBounds(int x,int y, int width, int height)	x the the number of pixels from the left of the screen y is is the number from the top of the screen.

Java AWT Example

- **To create simple awt example, you need a frame.**
- There are two ways to create a frame in AWT.
  - **By extending Frame class (inheritance)**
  - **By creating the object of Frame class (association)**

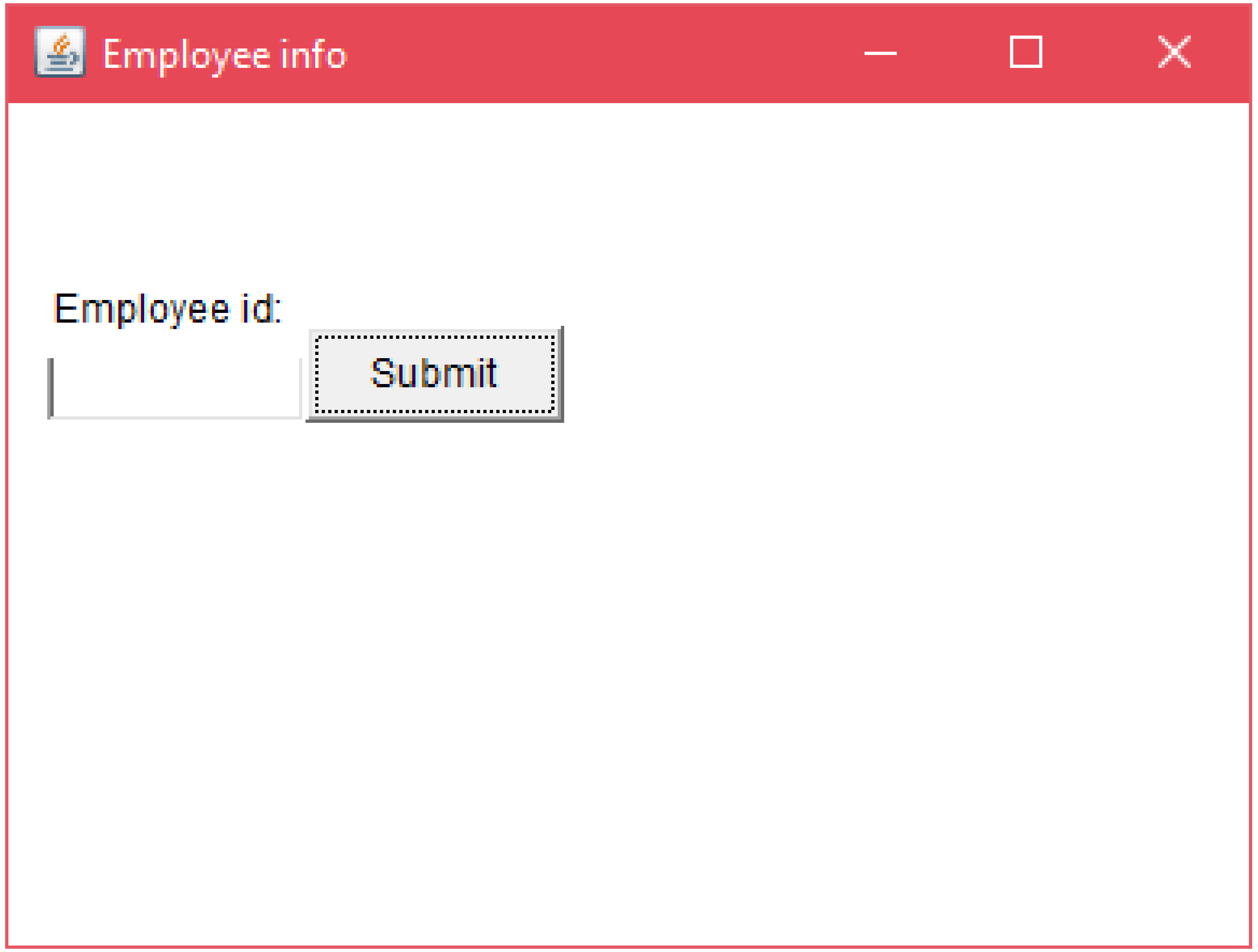
AWT Example by Inheritance



AWT Example by Inheritance

```
import java.awt.*;

class First extends Frame
{
  First()  // constructor
  {
    Button b=new Button("click me");

    b.setBounds(30,100,80,30); // setting button position

    add(b); //adding button into frame

                   setSize(300,300); //frame size 300 width and 300 height

    setTitle("This is our basic AWT Example");
    setLayout(null); //no layout manager

    setVisible(true); //now frame will be visible, by default not visible
    }
    public static void main(String args[])
    {
       First f=new First();
    }
}
```

AWT Example by Association



AWT Example by Association

```
import java.awt.*;

class AWTExample2
{

  AWTExample2() {

          Frame f = new Frame();    // creating a Frame

       Label l = new Label("Employee id:");    // creating a Label

          Button b = new Button("Submit");   // creating a Button

         TextField t = new TextField();    // creating a TextField
```

AWT Example by Association – Contd.

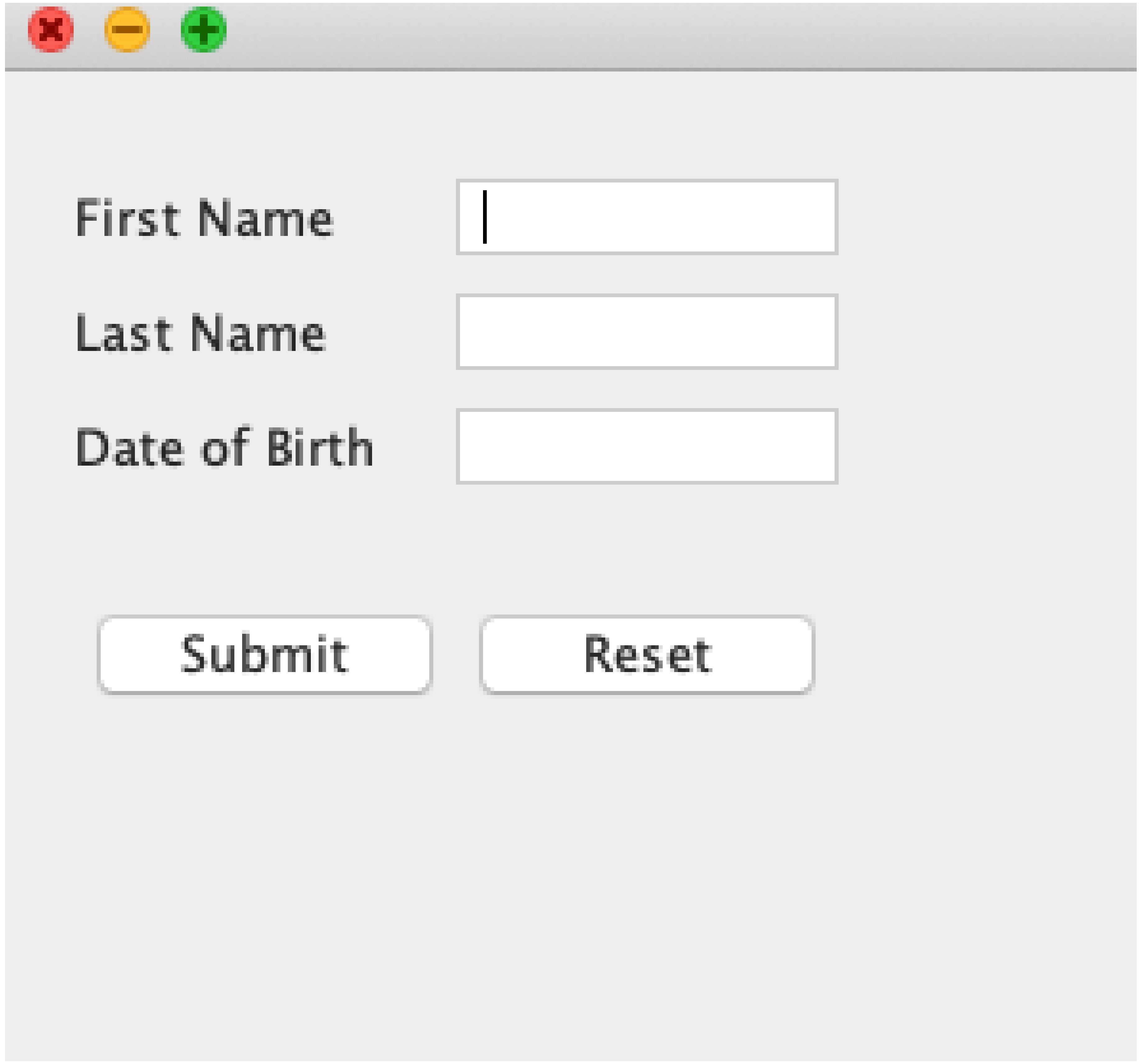
```
// setting position of above components in the frame
    l.setBounds(20, 80, 80, 30);
    t.setBounds(20, 100, 80, 30);
    b.setBounds(100, 100, 80, 30);

    // adding components into frame
    f.add(b);
    f.add(l);
    f.add(t);
```

AWT Example by Association – Contd.

```
// frame size 300 width and 300 height
    f.setSize(400,300);

    // setting the title of frame
    f.setTitle("Employee info");

    // no layout
    f.setLayout(null);

    // setting visibility of frame
    f.setVisible(true);
}
```

AWT Example by Association – Contd.

```
// main method
public static void main(String args[])
 {

// creating instance of Frame class
AWTExample2 awt_obj = new AWTExample2();

}

}
```


First Name

Last Name

Date of Birth

Event Handling

Any change in the state of any object is called event.

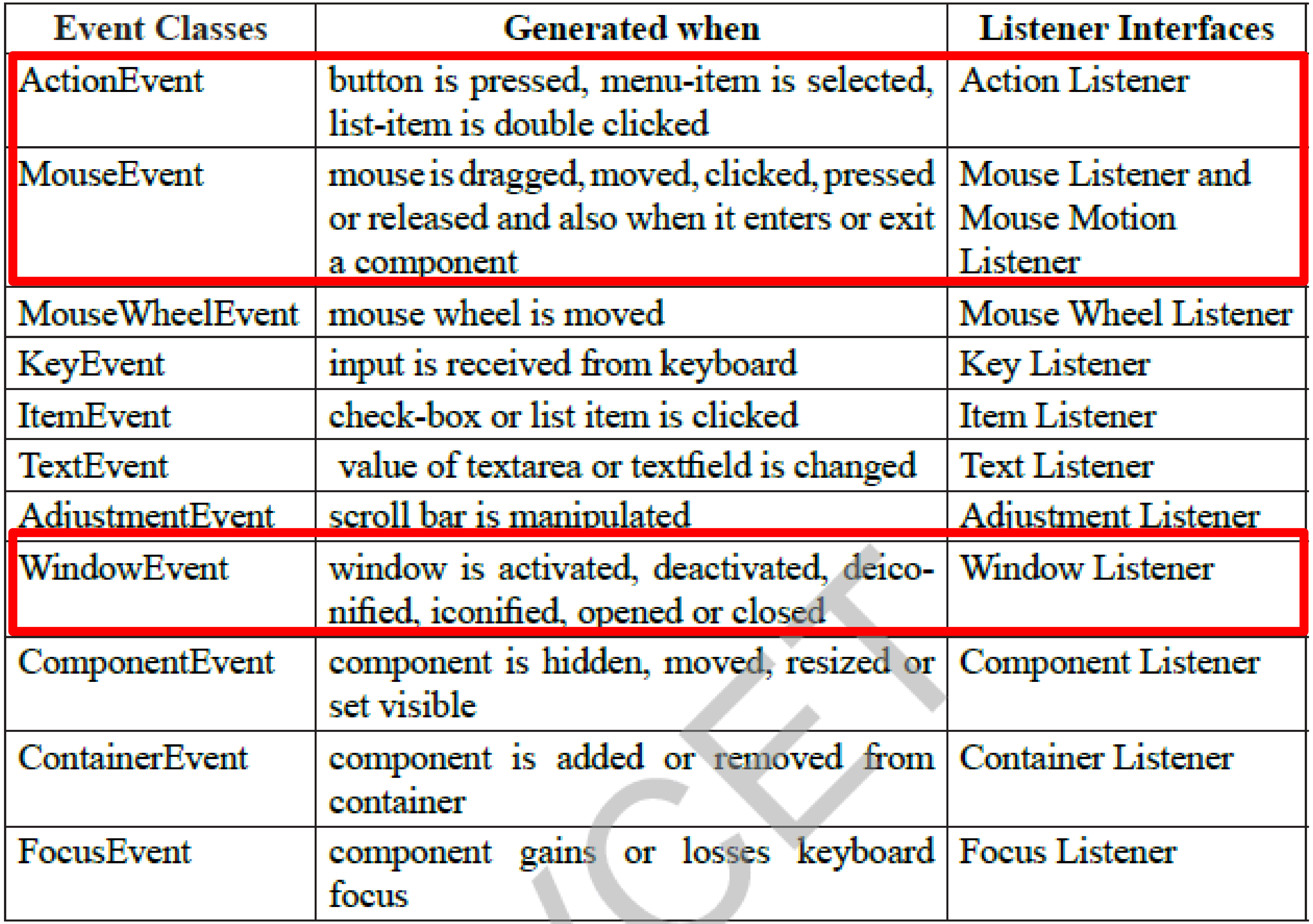
Example:

Pressing a button, entering a character in Textbox, Clicking or dragging a mouse, etc.

The three main components in event handling are:

- ***Events:***
    - **An event is a change in state of an object.**
    - Example, mouseClicked, mousePressed.
- ***Events Source:***
    - **Event source is an object that generates an event.**
    - Example: Button, Frame, TextField.
- ***Listeners:***
    - **A listener is an object that listens to the event.**
    - A listener gets notified when an event occurs.
    - When listener receives an event, it process it and then return.

Some of the event classes and Listener interfaces are listed below.



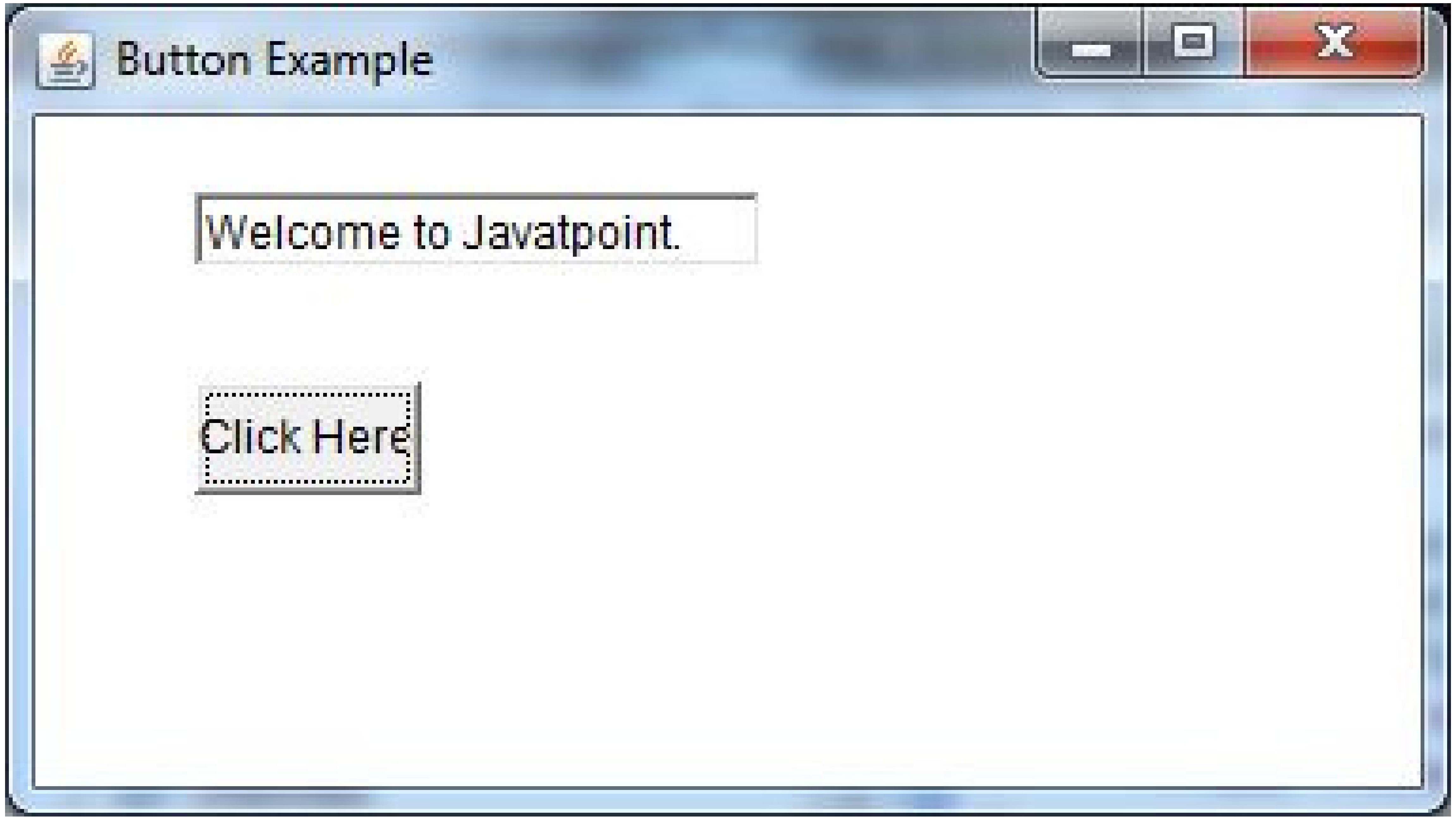
Event Classes	Generated when	Listener Interfaces
ActionEvent	button is pressed, menu-item is selected, list-item is double clicked	Action Listener
MouseEvent	mouse is dragged, moved, clicked, pressed or released and also when it enters or exit a component	Mouse Listener and Mouse Motion Listener
MouseWheelEvent	mouse wheel is moved	Mouse Wheel Listener
KeyEvent	input is received from keyboard	Key Listener
ItemEvent	check-box or list item is clicked	Item Listener
TextEvent	value of textarea or textfield is changed	Text Listener
AdjustmentEvent	scroll bar is manipulated	Adjustment Listener
WindowEvent	window is activated, deactivated, deico-nified, iconified, opened or closed	Window Listener
ComponentEvent	component is hidden, moved, resized or set visible	Component Listener
ContainerEvent	component is added or removed from container	Container Listener
FocusEvent	component gains or losses keyboard focus	Focus Listener

Java ActionListener Interface :

java.awt.event. ActionEvent;
java.awt.event.ActionListener;

- The Java ActionListener is notified whenever you click on the button or menu item.
- It is notified against ActionEvent.
- It has only one method: actionPerformed().
- The actionPerformed() method is invoked automatically whenever you click on the registered component.

```
public abstract void actionPerformed(ActionEvent e) { }
```


Button Example
Welcome to Javatpoint.
Click Here

Java ActionListener Example: On Button click

```
import java.awt.*;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
public class ActionExample extends Frame implements ActionListener
{       TextField tf;
        Button b;
        ActionExample ()
        {
            setTitle("ActionListener Example");
            tf=new TextField(20);
            setBounds(50,50, 150,20);
            Button b=new Button("Click Here");
            b.setBounds(50,100,120,30);
            b.addActionListener(this);
            add(b);
            add(tf);
            setSize(400,400);
            setLayout(null);
            setVisible(true);
    }
public void actionPerformed(ActionEvent e)
   {
          tf.setText("Welcome to  ");
   }

public static void main(String args[])
{

     ActionExample a = new ActionExample ();

   }
}
```

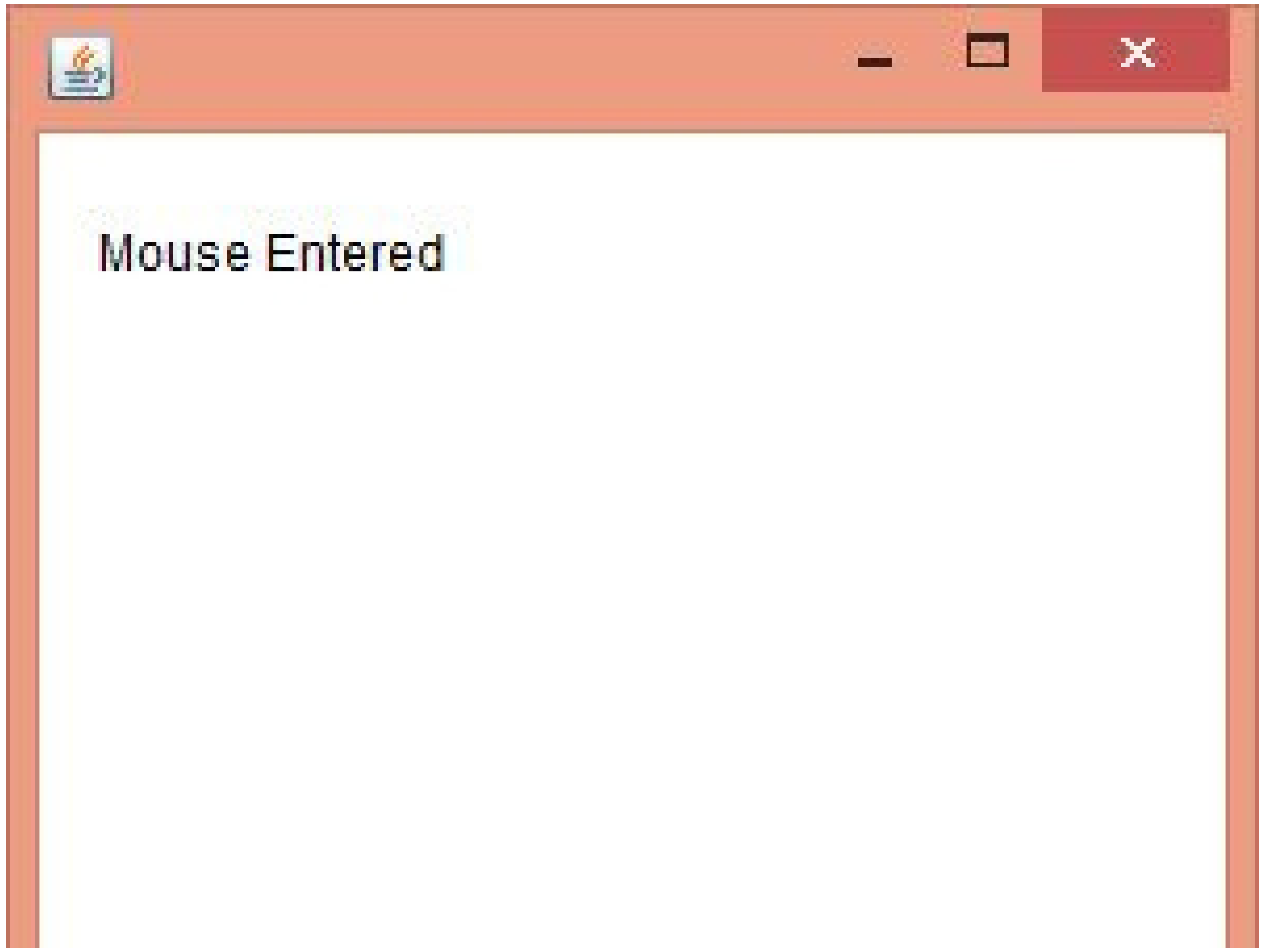
MouseListener Interface :

java.awt.event.MouseEvent & java.awt.event.MouseListener

- The Java MouseListener is notified whenever you change the state of mouse.
- It is notified against MouseEvent.
- It has five methods.

  - **public abstract void** mouseClicked(MouseEvent e){ }
  - **public abstract void** mouseEntered(MouseEvent e){ }
  - **public abstract void** mouseExited(MouseEvent e) { }
  - **public abstract void** mousePressed(MouseEvent e) { }
  - **public abstract void** mouseReleased(MouseEvent e) { }

Java MouseEvent and MouseListener Example



```
Mouse Entered
```

```
import java.awt.*;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;
```

```
public class MouseListenerExample  implements MouseListener
{
        Label l;

        MouseListenerExample()
         {
            Frame f = new Frame(“Mouse Listener”);

            f.addMouseListener(this);

                    l=new Label();

            l.setBounds(20,50,100,20);

            f.add(l);

            f.setSize(300,300);
            f.setLayout(null);
            f.setVisible(true);
        }
```

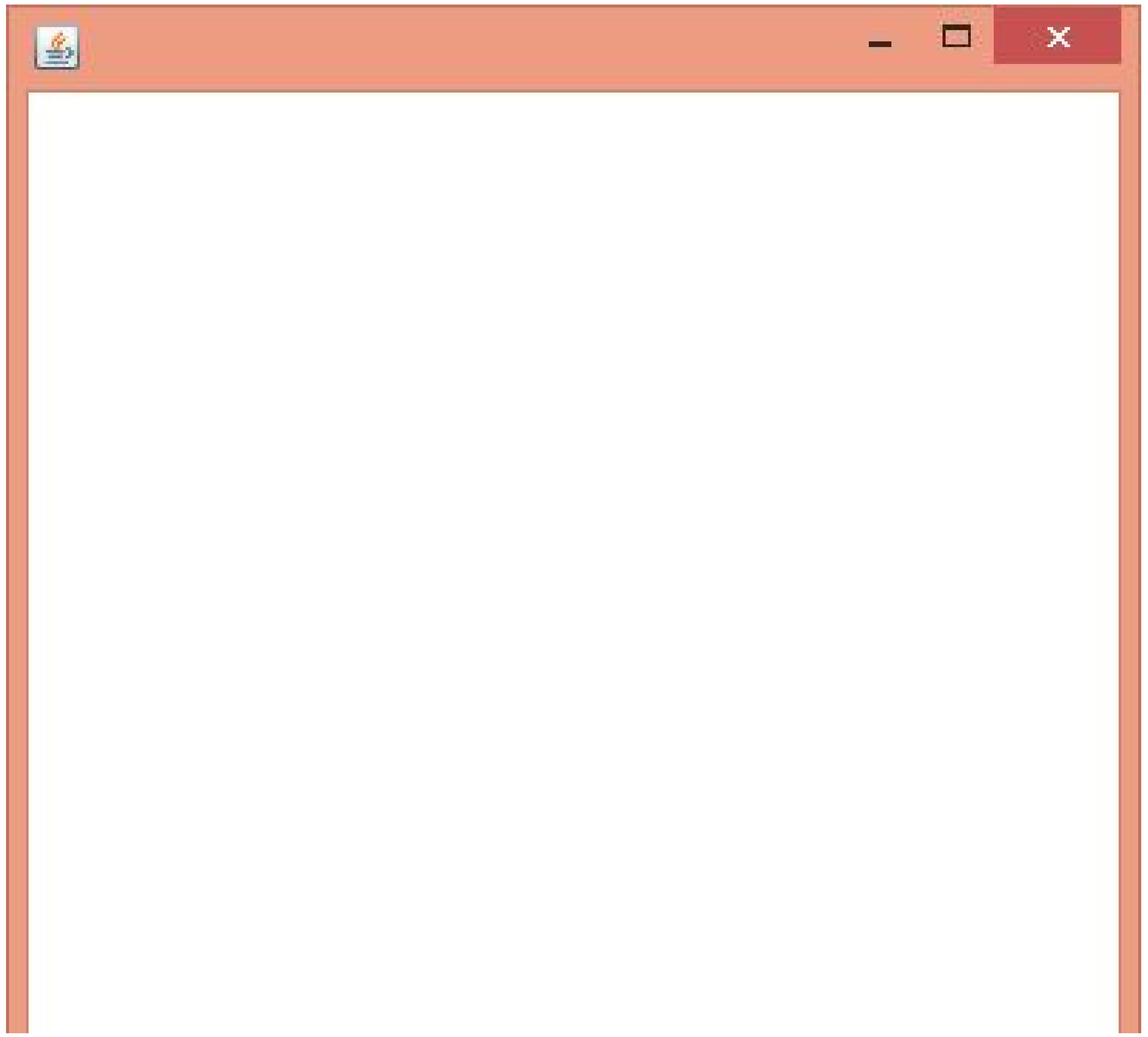
```
public void mouseClicked(MouseEvent e)
  {
     l.setText("Mouse Clicked");
   }
  public void mouseEntered(MouseEvent e)
  {
     l.setText("Mouse Entered");
   }
  public void mouseExited(MouseEvent e)
  {
     l.setText("Mouse Exited");
   }
```

```
public void mousePressed(MouseEvent e)
 {
       l.setText("Mouse Pressed");
   }
   public void mouseReleased(MouseEvent e)
  {
       l.setText("Mouse Released");
   }
public static void main(String[] args)
{
      MouseListernerExample  m = new MouseListenerExample();
 }
}    // main class
```

Java WindowListener Interface :

java.awt.event.WindowEvent;   java.awt.event.WindowListener;

- The Java WindowListener is notified whenever you change the state of window.
- It is notified against WindowEvent.
- It has three methods.
  - **public abstract void** windowActivated(WindowEvent e) { }
  - **public abstract void** windowClosed(WindowEvent e) { }
  - **public abstract void** windowOpened(WindowEvent e); { }



```
import java.awt.*;

import java.awt.event.WindowEvent;

import java.awt.event.WindowListener;
```

```
public class WindowExample  implements WindowListener
{
   WindowExample()
  {
     Frame f = new frame("Window Event");
     f.addWindowListener(this);
              f.setSize(400,400);
      f.setLayout(null);
              f.setVisible(true);
   }
```

```
public void windowActivated(WindowEvent  e)
{
   System.out.println("activated");
}
public void windowClosed(WindowEvent e)
{
   System.out.println("closed");
}
public void windowOpened(WindowEvent e)
{
   System.out.println("opened");
}

public static void main(String[] args)
 {
     WindowExample w= new WindowExample();
 }
```

Adapter Classes

- **An adapter class provides the default implementation of all methods in an event listener interface.**
- Adapter classes are very useful when you want to process only few of the events that are handled by a particular event listener interface.

Example:

- MouseAdapter provides empty implementation of MouseListener interface.

- It is useful because very often you do not really use all methods declared by interface,

  so implementing the interface directly is very lengthy.

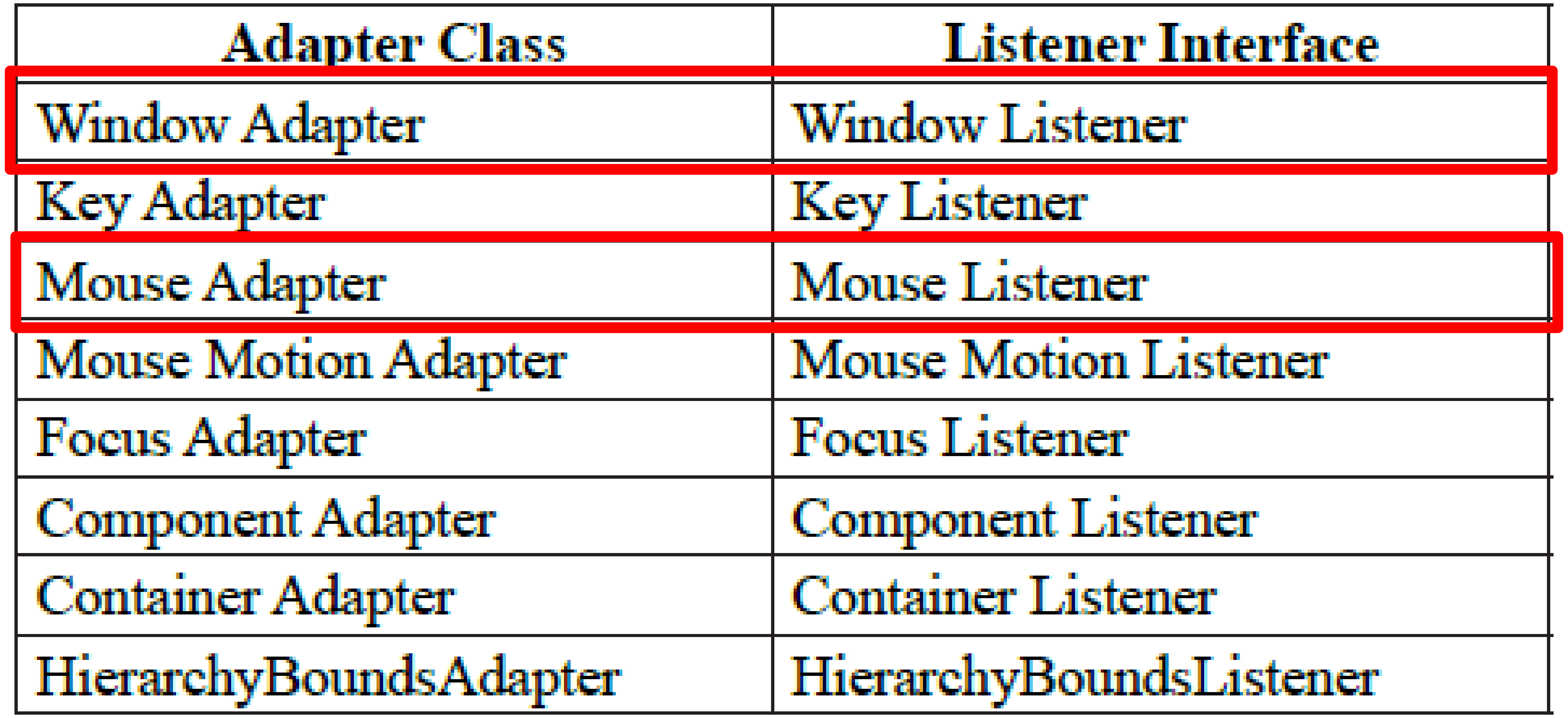
Adapter Classes

- Adapter class **is a simple java class that implements an interface with only EMPTY implementation.**

- **Instead of implementing interface if we extends Adapter class, we provide implementation only for require method**

Adapter Classes

- The adapter classes are found in **java.awt.event**
- The Adapter classes with their corresponding listener interfaces are as follows.



Adapter Class	Listener Interface
Window Adapter	Window Listener
Key Adapter	Key Listener
Mouse Adapter	Mouse Listener
Mouse Motion Adapter	Mouse Motion Listener
Focus Adapter	Focus Listener
Component Adapter	Component Listener
Container Adapter	Container Listener
HierarchyBoundsAdapter	HierarchyBoundsListener

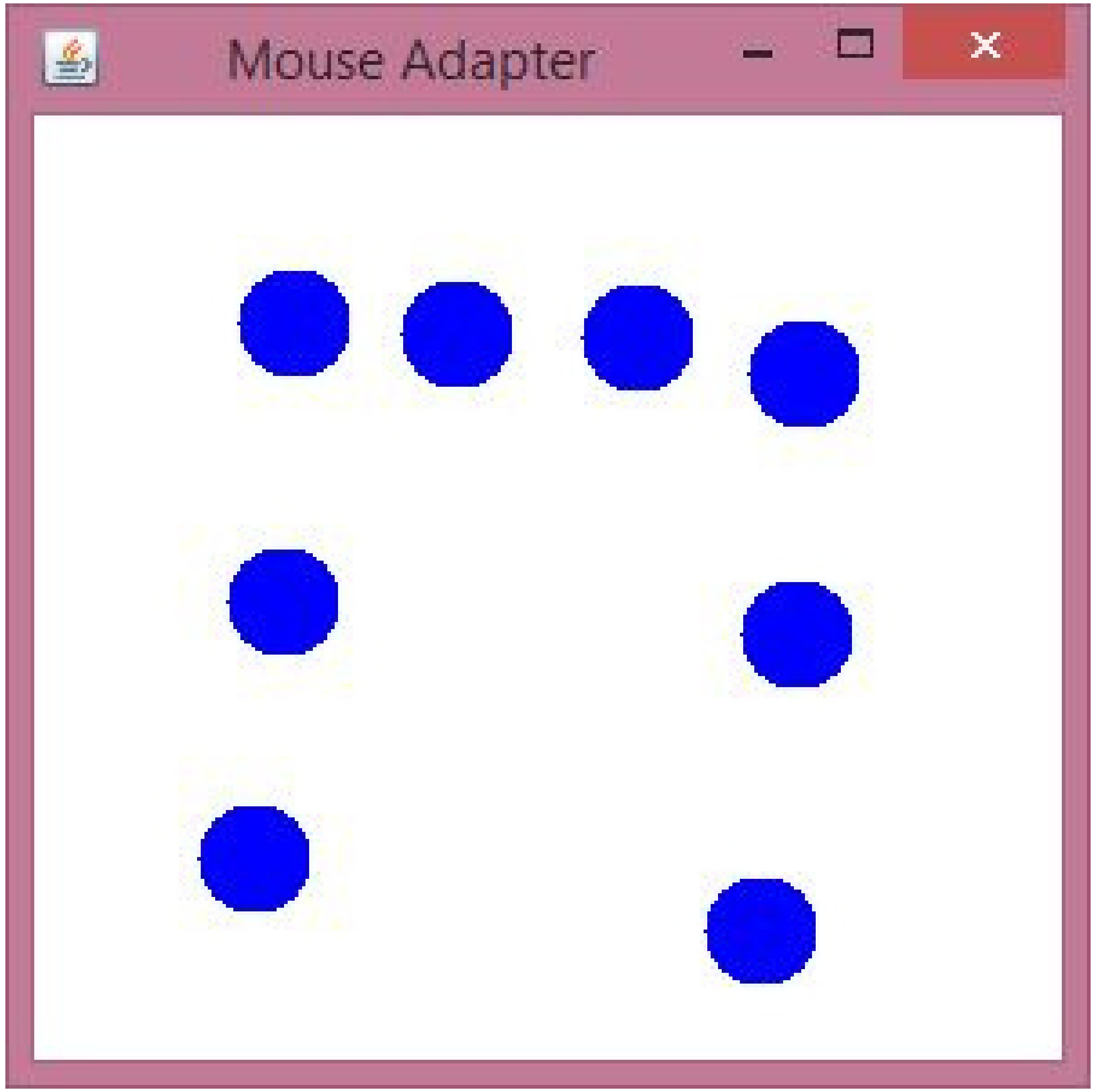
WindowAdapter Class : Example

MouseAdapter Class : Example

java WindowAdapter Example

```
import java.awt.*;

import java.awt.event.*;

import javax.swing.*;
```

```
public class WindowAdapterExample extends WindowAdapter
{
    JFrame f;
     WindowAdapterExample()
       {
          f=new JFrame("Window Adapter");
          f.addWindowListener(this);
          f.setSize(400,400);
          f.setLayout(null);
          f.setVisible(true);
    }
public void windowActivated(WindowEvent e)
    {
        System.out.println("activated");
    }
    public static void main(String[] args)
    {
        new WindowAdapterExample();
    }
}
```


Mouse Adapter

Java MouseAdapter Example

```
import java.awt.*;

import java.awt.event.*;

import javax.swing.*;
```

```
public class MouseAdapterExample extends MouseAdapter
{
    JFrame f;
    MouseAdapterExample()
  {
        f=new JFrame("Mouse Adapter");
        f.addMouseListener(this);

        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
public void mouseClicked(MouseEvent e)
 {
        Graphics g=f.getGraphics();
        g.setColor(Color.BLUE);
        g.fillOval(e.getX(),e.getY(),30,30);
    }

public static void main(String[] args)

{
   new MouseAdapterExample();
}
}
```

Swing:

- Swing is a lightweight Java graphical user interface (GUI) that is used to create various applications.
- Swing has components which are platform-independent.
- It enables the user to create buttons and scroll bars.
- Swing includes packages for creating desktop applications in Java.
- It is a part of **Java Foundation Classes(JFC).**

Difference between AWT and Swing:

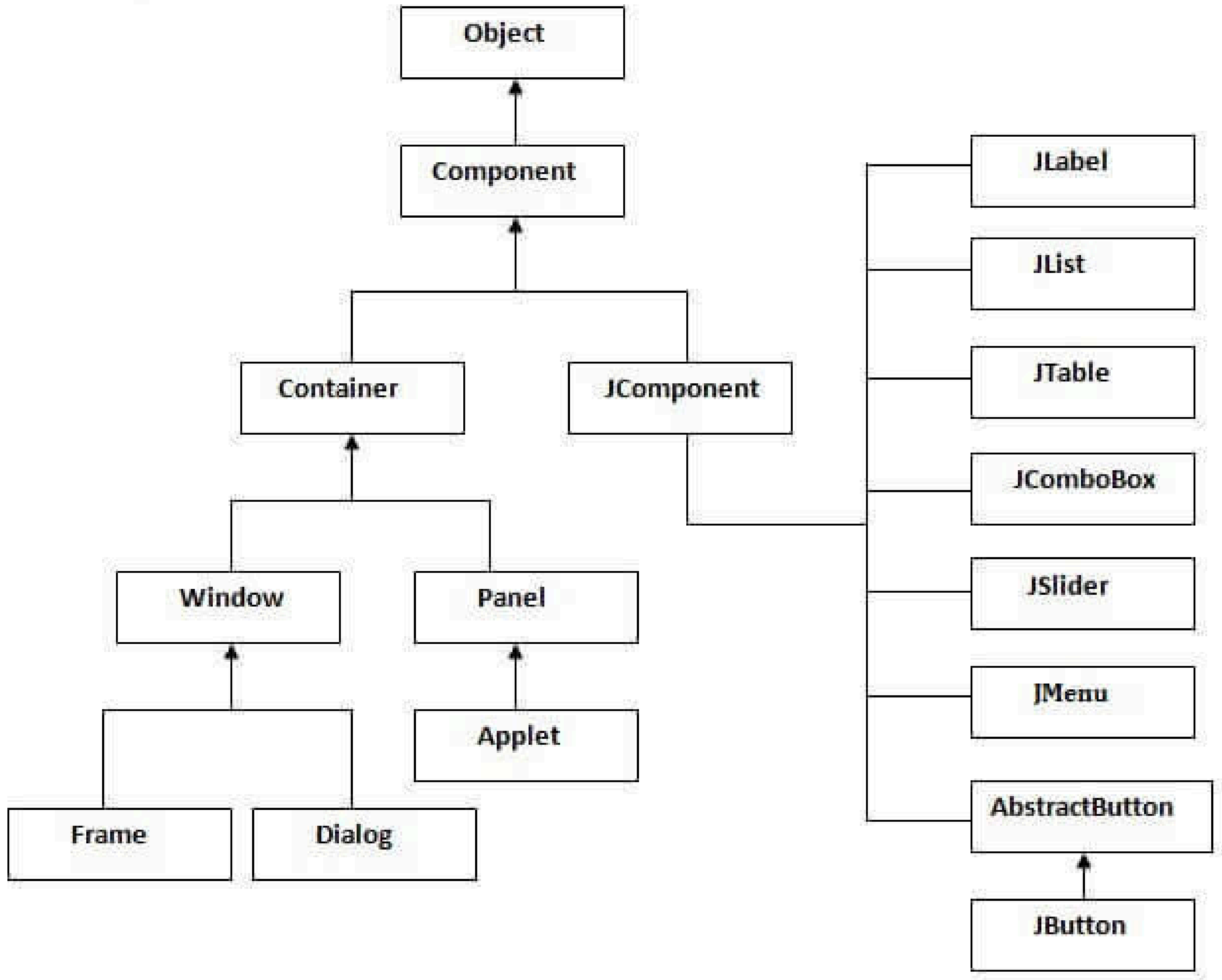
S.N O	AWT	Swing
1.	Java AWT is an API to develop GUI applications in Java	Swing is a part of Java Foundation Classes and is used to create various applications.
2.	The components of Java AWT are heavy weighted.	The components of Java Swing are light weighted.
3.	Java AWT has comparatively less functionality as compared to Swing.	Java Swing has more functionality as compared to AWT.

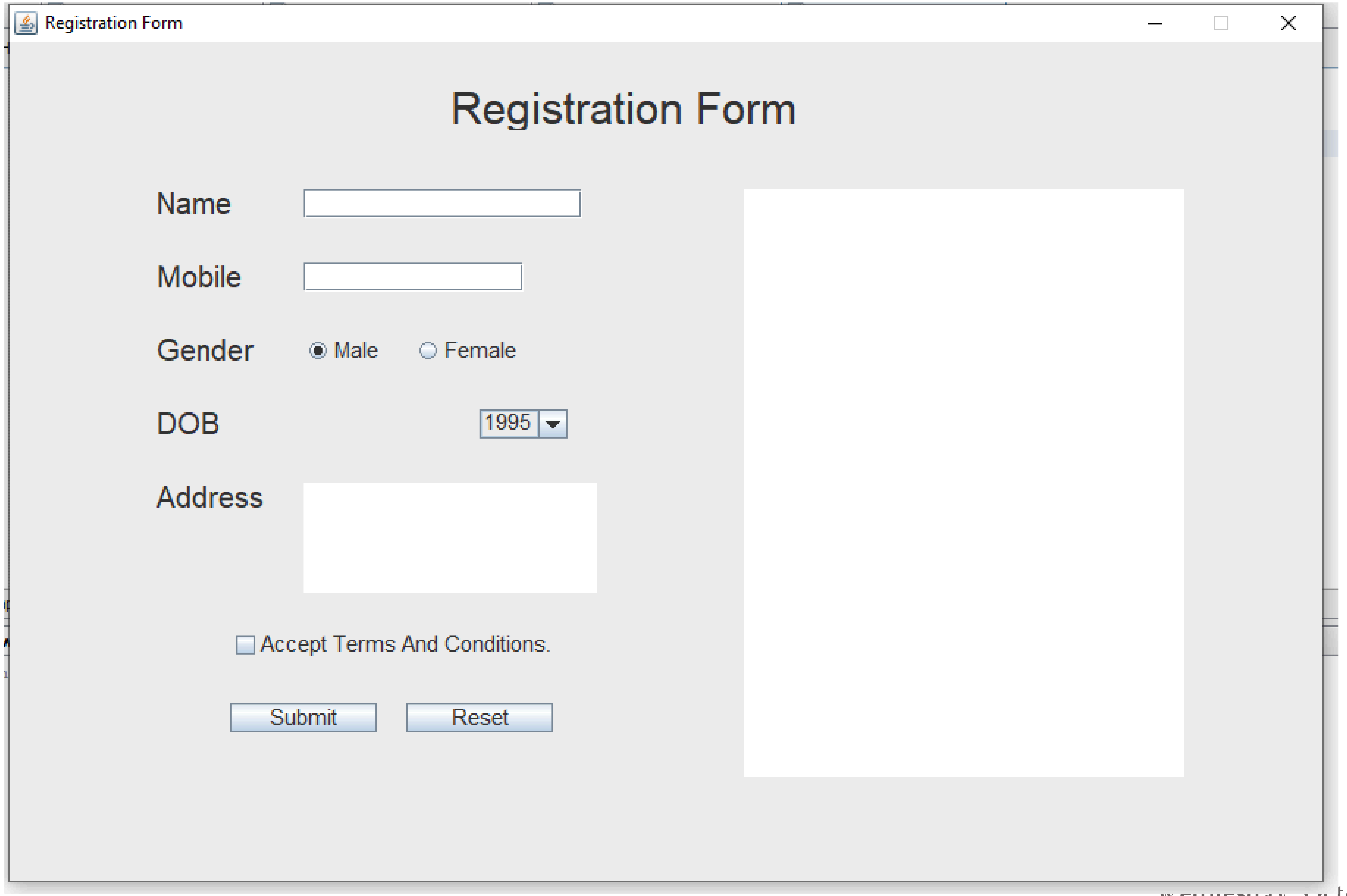
Difference between AWT and Swing:

3.	The execution time of AWT is more than Swing.	The execution time of Swing is less than AWT.
4.	The components of Java AWT are platform dependent.	The components of Java Swing are platform independent.
5.	MVC pattern is not supported by AWT.	MVC pattern is supported by Swing.
6.	AWT provides comparatively less powerful components.	Swing provides more powerful components.

Hierarchy of Java Swing classes

import javax.swing.*;





Registration Form

Registration Form

Name	
Mobile	
Gender	[✓ Male] Female
DOB	1995
Address	

☐ Accept Terms And Conditions.

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
```

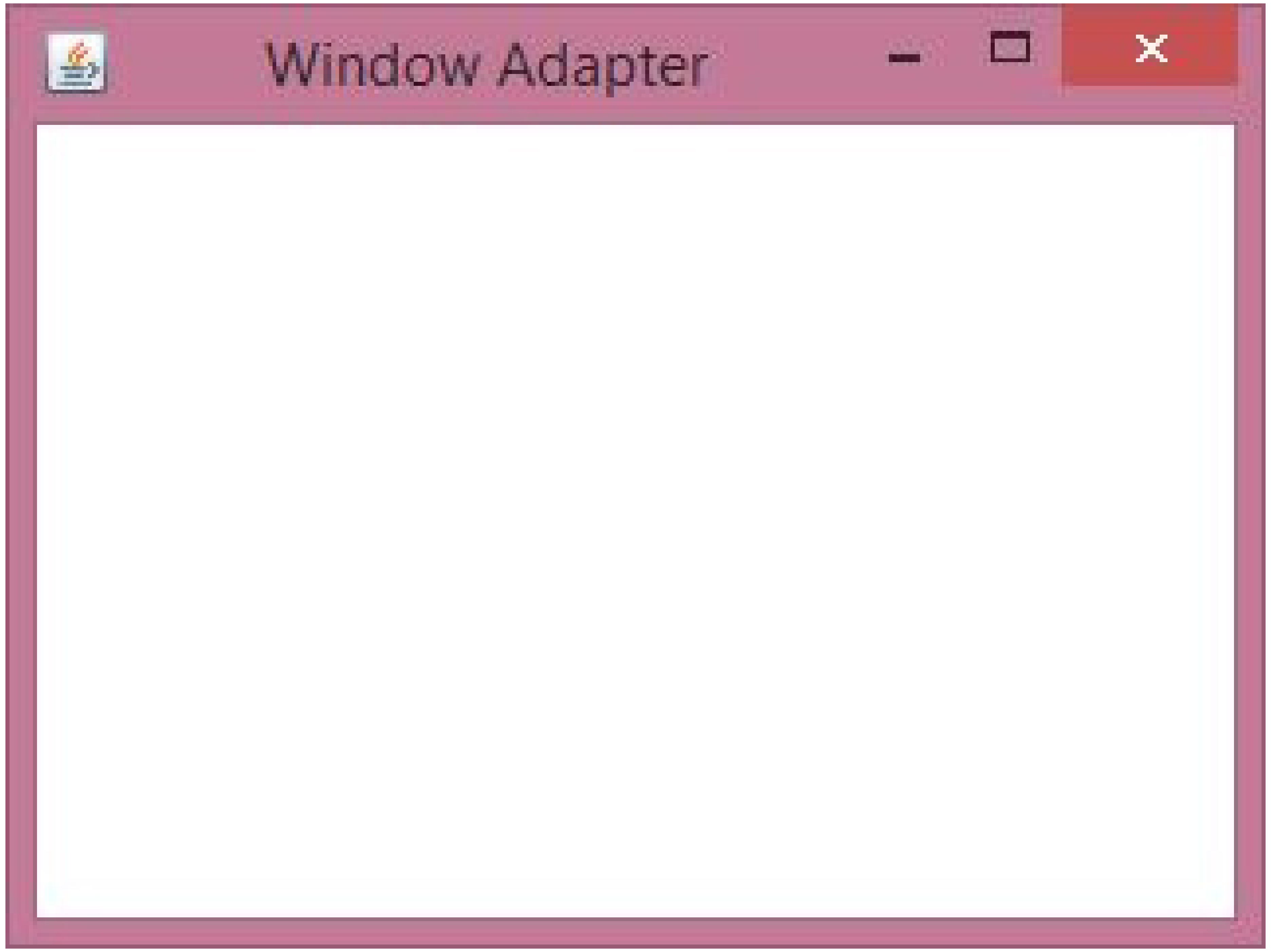
Commonly used Methods of AbstractButton class:

Methods	Description
void setText(String s)	It is used to set specified text on button
String getText()	It is used to return the text of the button.
void setEnabled(boolean b)	It is used to enable or disable the button.
void addActionListener(ActionListener a)	It is used to add the action listener to this object.

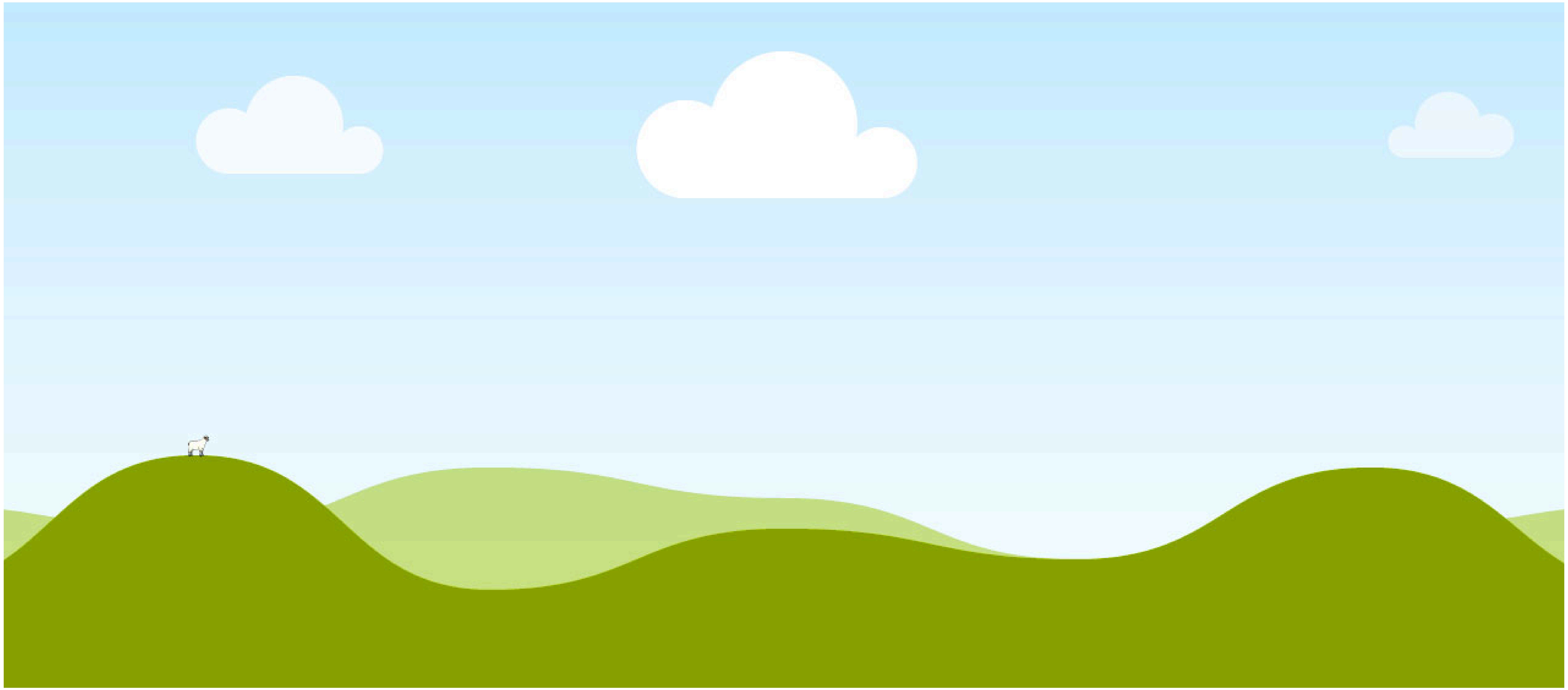
The adapter classes are found in **java.awt.event** and **javax.swing.event** packages.

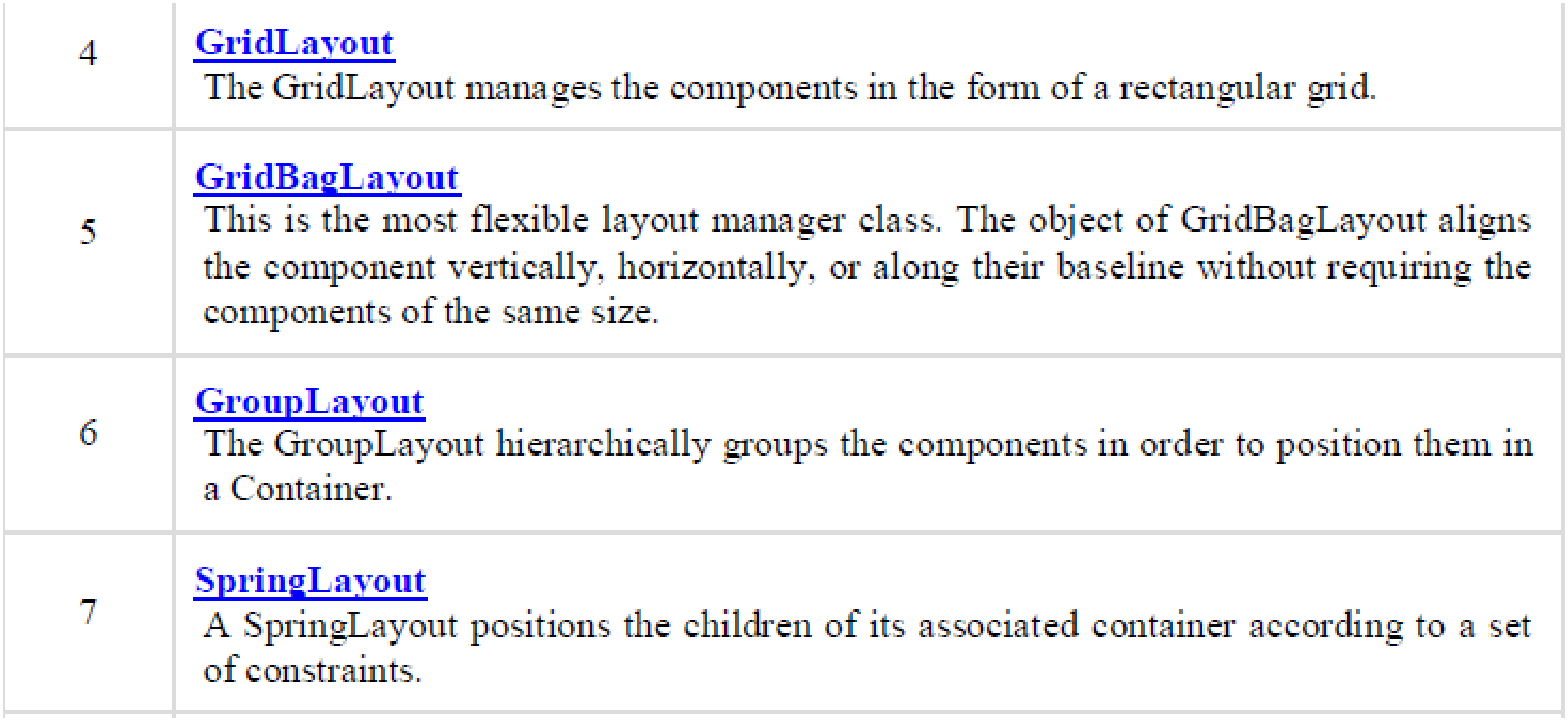
The Adapter classes with their corresponding listener interfaces are as follows.

Adapter Class	**Listener Interface**
Window Adapter	Window Listener
Key Adapter	Key Listener
Mouse Adapter	Mouse Listener
Mouse Motion Adapter	Mouse Motion Listener
Focus Adapter	Focus Listener
Component Adapter	Component Listener
Container Adapter	Container Listener
HierarchyBoundsAdapter	HierarchyBoundsListener

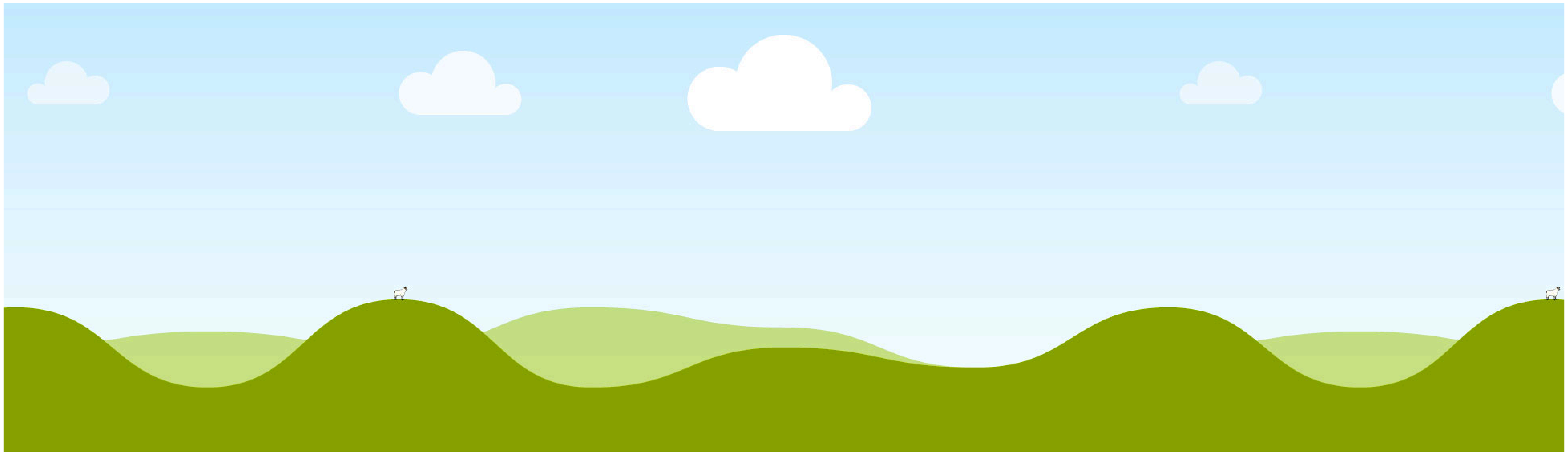

Window Adapter

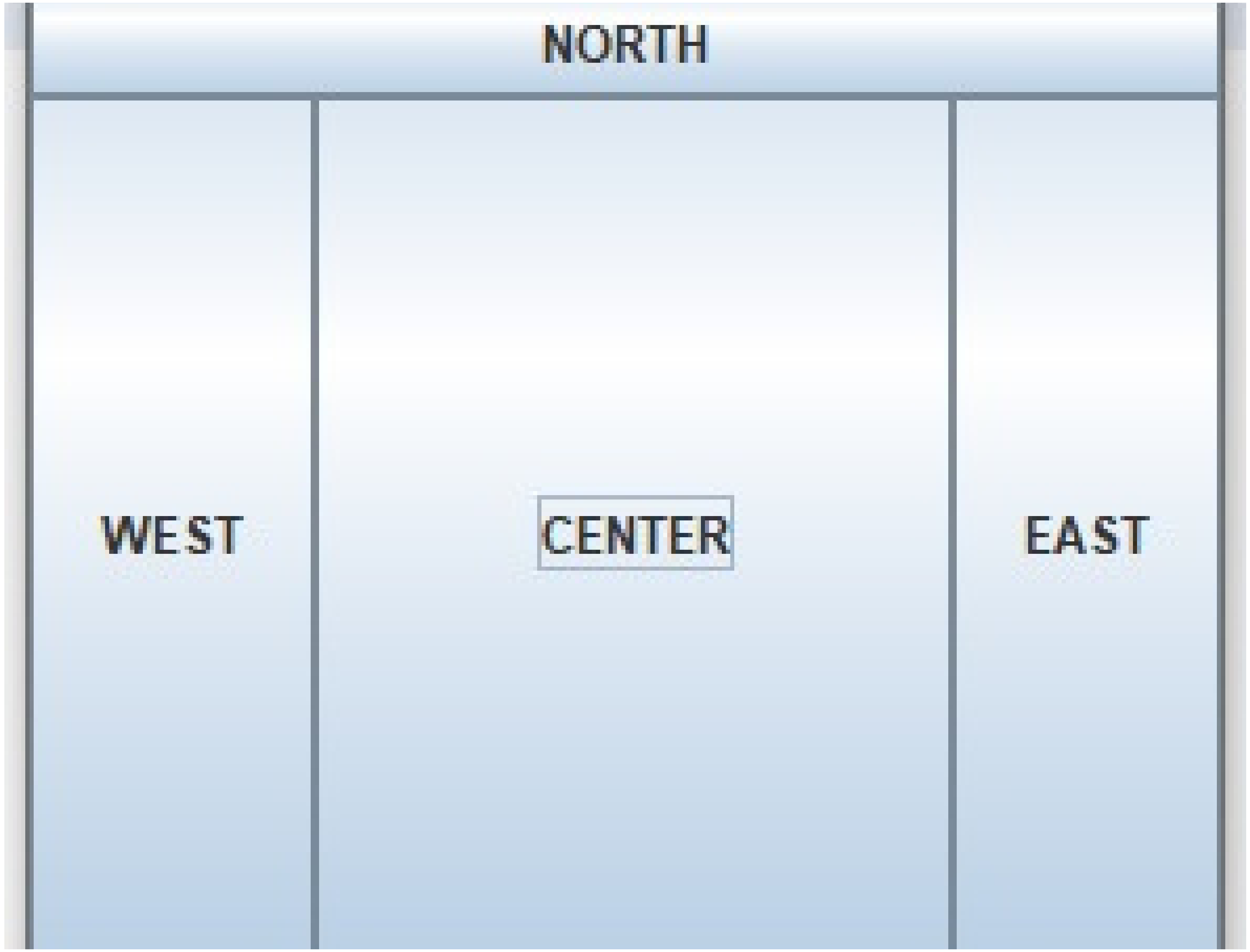
- **Layout management - Java LayoutManagers**
- The LayoutManagers are used to arrange components in a particular manner.
- LayoutManager is an interface that is implemented by all the classes of layout managers.





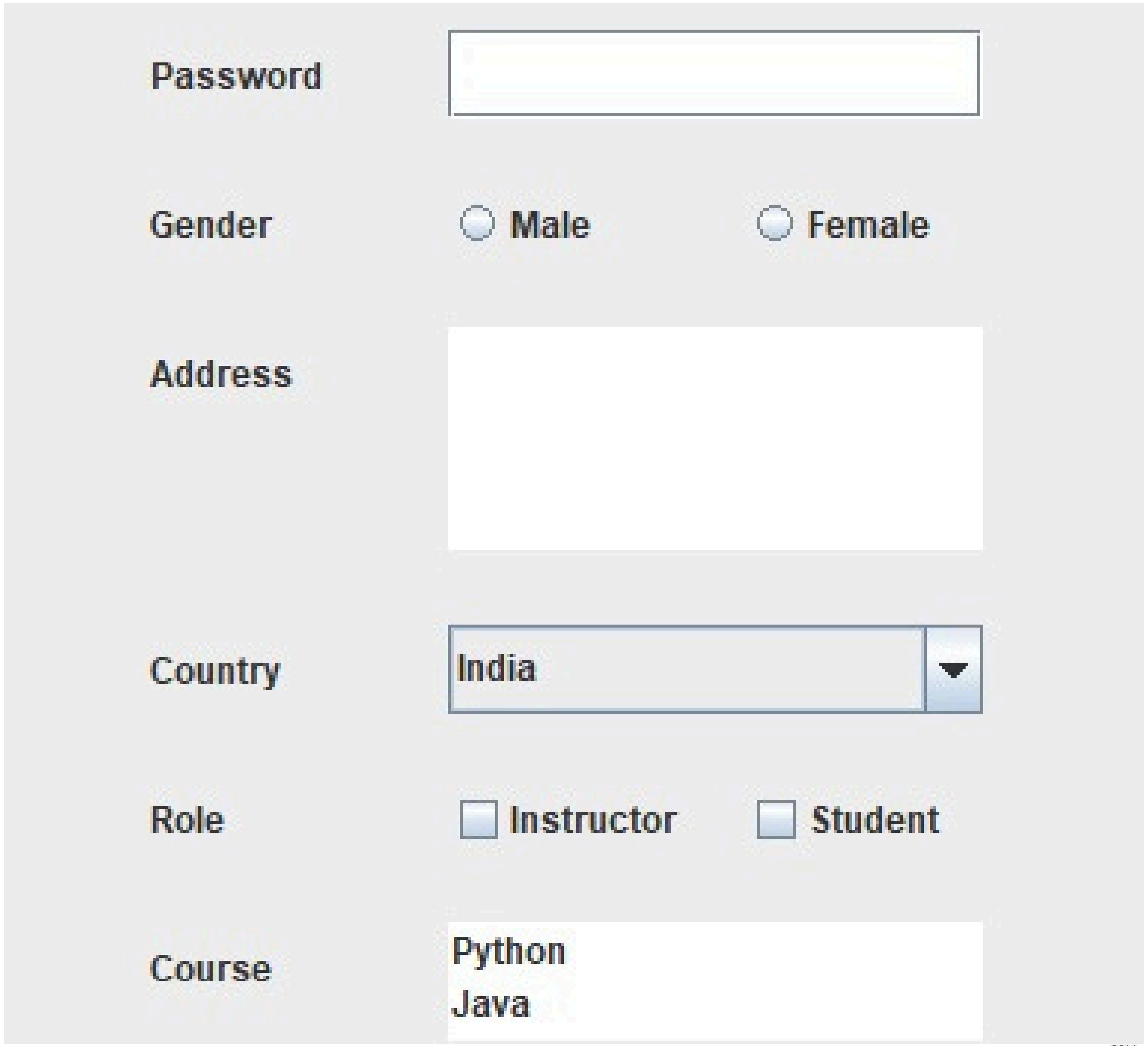
4	**GridLayout** The GridLayout manages the components in the form of a rectangular grid.
5	**GridBagLayout** This is the most flexible layout manager class. The object of GridBagLayout aligns the component vertically, horizontally, or along their baseline without requiring the components of the same size.
6	**GroupLayout** The GroupLayout hierarchically groups the components in order to position them in a Container.
7	**SpringLayout** A SpringLayout positions the children of its associated container according to a set of constraints.




NORTH
WEST
CENTER
EAST

Swing Components

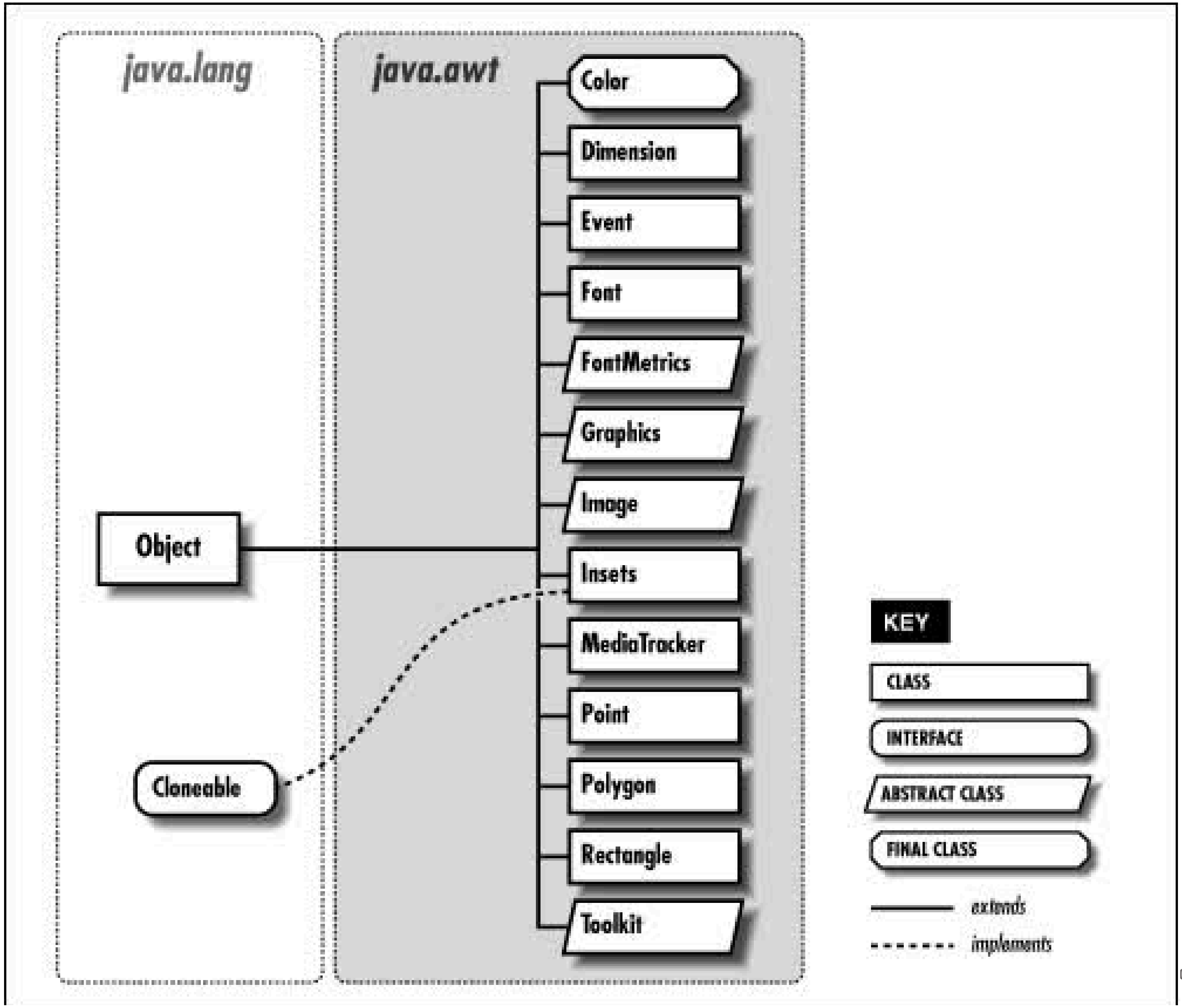
- Text Fields
- Label
- PasswordFiled
- Text Areas
- Buttons
- Check Boxes
- Radio Buttons
- Lists
- Choices
- Scrollbars
- Windows
- Menus – MenuBar, Menu, MenuItem
- Dialog Boxes.



Password		
Gender	○ Male	○ Female
Address		
Country	India ▾	
Role	☐ Instructor	☐ Student
Course	Python Java	

AWT

Graphics programming - Frame – Components
Working with 2D shapes - Using color, fonts, and images


java.lang
java.awt
Object
Cloneable
Color
Dimension
Event
Font
FontMetrics
Graphics
Image
Insets
MediaTracker
Point
Polygon
Rectangle
Toolkit
KEY
CLASS
INTERFACE
ABSTRACT CLASS
FINAL CLASS
extends
implements

- **Graphics :**
  - Encapsulates the graphics context. This context is used by the various output methods to display output in a window.
- **Color :** Manages colors in a portable, platform-independent fashion.
- **Component :** An abstract superclass for various AWT components.
- **Container :** A subclass of Component that can hold other components
- **Font :** Encapsulates a type font.
- **Image :** Encapsulates graphical images.

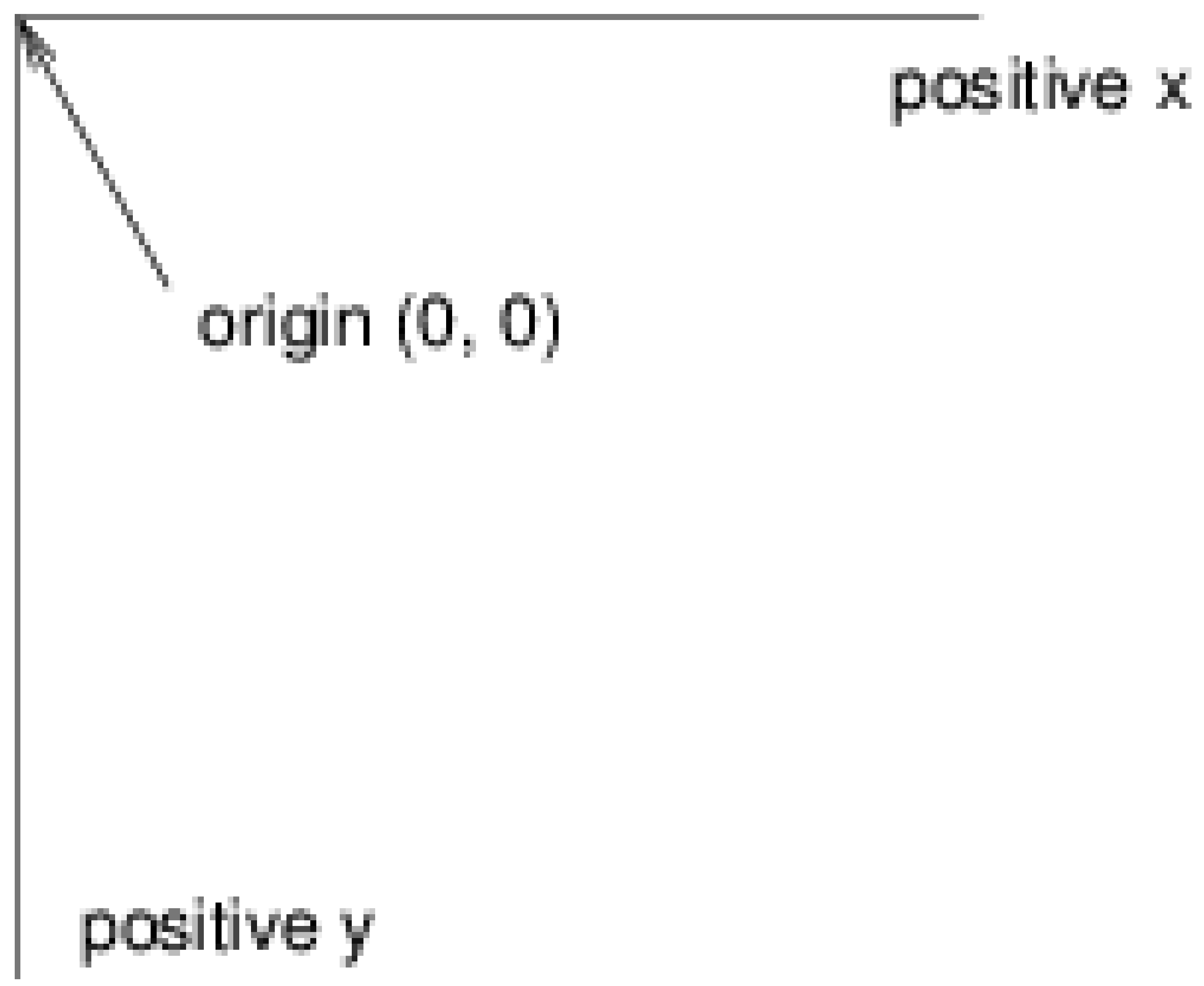
- **Frame**
  - Frame encapsulates a “window.” It is a subclass of Window and has a title bar, menu bar,   borders, and resizing corners.

- **Canvas**
  - It is not considered as a part of the hierarchy for applet or frame windows. Canvas encapsulates a blank window upon which you can draw.

  Graphics class

  - The Graphics class is an abstract class that provides the means to access different graphics devices.

  - It lets us draw images on the screen, display images,


positive x
origin (0, 0)
positive y

Displaying Graphics in Applet

Commonly used methods of Graphics class

public abstract void drawString(String str, int x, int y)

is used to draw the specified string.

public abstract void drawLine(int x1, int y1, int x2, int y2)

is used to draw line between the points(x1, y1) and (x2, y2).

public abstract void setColor(Color c) **: Color**

is used to set the graphics current color to the specified color.

public abstract void setFont(Font font) **: Font**

is used to set the graphics current font to the specified font.

- **public void drawRect(int x, int y, int width, int height)**

  draws a rectangle with the specified width and height.

- **public abstract void fillRect(int x, int y, int width, int height)**

  is used to fill rectangle with the default color and specified width and height.

public abstract void drawOval(int x, int y, int width, int height)

is used to draw oval with the specified width and height.

public abstract void fillOval(int x, int y, int width, int height)

is used to fill oval with the default color and specified width and height.

public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)

is used draw a circular or elliptical arc.

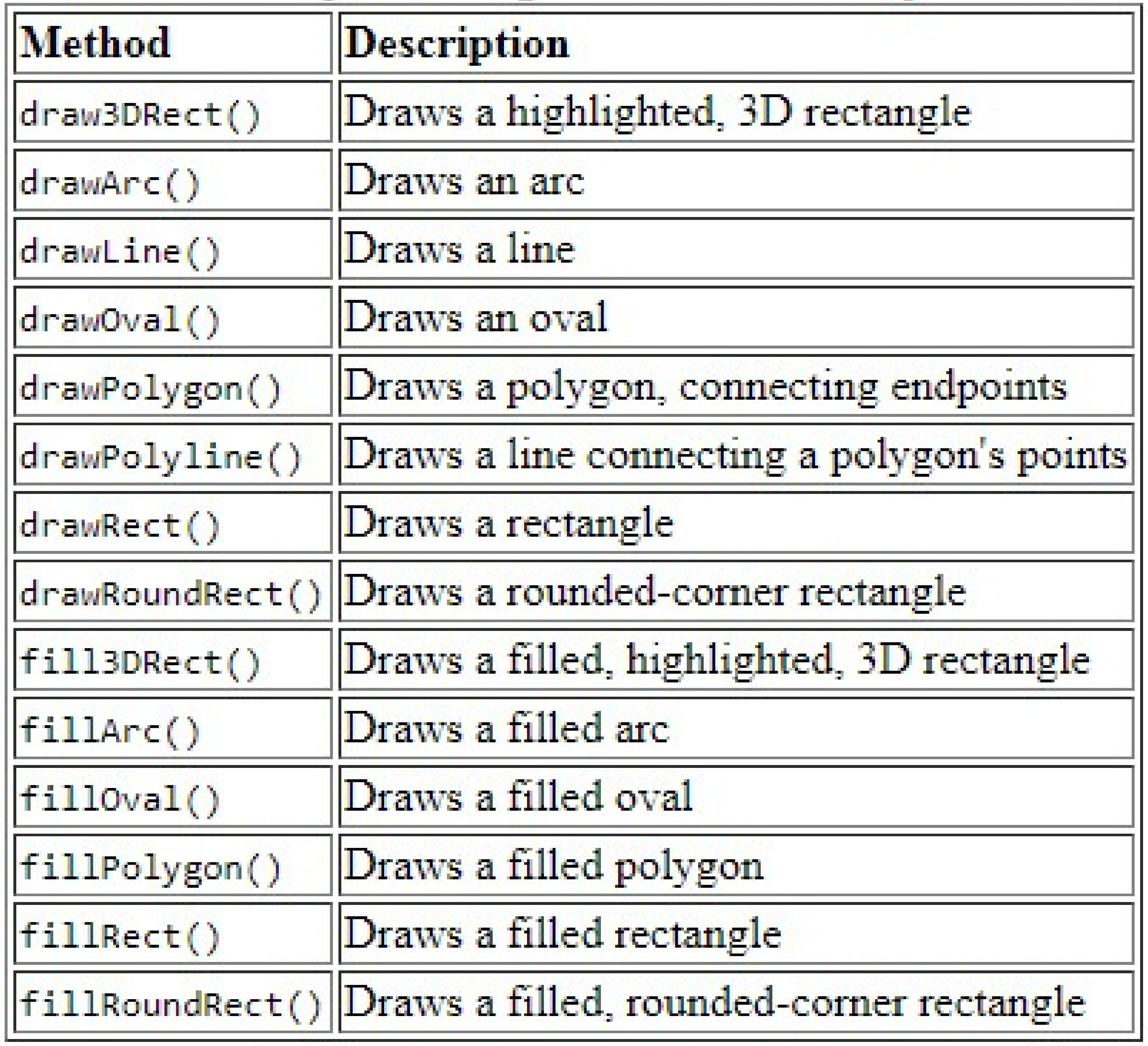
public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)

is used to fill a circular or elliptical arc.

public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer)

is used draw the specified image.

Table 13.1: Shape Drawing Methods in the Graphics Class



Method	Description
`draw3DRect()`	Draws a highlighted, 3D rectangle
`drawArc()`	Draws an arc
`drawLine()`	Draws a line
`drawOval()`	Draws an oval
`drawPolygon()`	Draws a polygon, connecting endpoints
`drawPolyline()`	Draws a line connecting a polygon's points
`drawRect()`	Draws a rectangle
`drawRoundRect()`	Draws a rounded-corner rectangle
`fill3DRect()`	Draws a filled, highlighted, 3D rectangle
`fillArc()`	Draws a filled arc
`fillOval()`	Draws a filled oval
`fillPolygon()`	Draws a filled polygon
`fillRect()`	Draws a filled rectangle
`fillRoundRect()`	Draws a filled, rounded-corner rectangle

Colors in Java

- Java package comes with the Color class.
- Default sRGB color space

Color class static color variables available are:

Color.black Color.lightGray Color.blue Color.magenta Color.cyan

Color.orange Color.darkGray Color.pink Color.gray Color.red

Color.green Color.white Color.yellow

The current graphics color can be changed using setColor() method defined in Graphics class.

```
void setColor(Color newColor) // newColor indicates new drawing color
```

Fonts in Java

The Font class states fonts, which are used to render text in a visible way.

Font class constructor

- Font(Font font) //Creates a new Font from the specified font.
- Font(String name, int style, int size) //Creates a new Font from the specified name, style and point size.

Font variables available in Font class are:

Font.BOLD	Font. SANS_SERIF
Font.ITALIC	Font. PLAIN
Font. MONOSPACED	Font. SERIF
int size	int style
float pointSize	String name

Images in Java

Image control is superclass for all image classes representing graphical images.

Image class constructor

Image() // create an Image object

getImage() method that returns the object of Image.

Its syntax is as follows.

public Image getImage(URL u, String image) ;

drawImage(img, 60,80, this);

```
import java.awt.Graphics;

import java.awt.Canvas;

import java.awt.Color;

import java.awt.Font;

import java.awt.Image;

import java.awt.Toolkit;

import javax.swing.JFrame;
```

```
public class JavaApplication4 extends Canvas
{
    public static void main(String[] args)
{
        // TODO code application logic here

         JFrame frame = new JFrame("My Drawing");

         Canvas canvas = new JavaApplication4();

         canvas.setSize(400, 400);

         frame.add(canvas);

         frame.pack();

         frame.setVisible(true);

         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
```

```
public void paint(Graphics g)
{
    g.setColor(Color.red);

    g.drawString("Welcome to Kamaraj",4,20);

     g.setFont(new Font("Monospaces",Font.PLAIN,12));

    g.drawString("Font in PLAIN", 50, 70);

    g.setFont(new Font("Serif", Font.ITALIC, 24));

    g.drawString("Font in ITALIC", 50, 120);

    g.setFont(new Font("Serif", Font.BOLD, 24));

    g.drawString(("Font in Bold", 50, 150);
}
```

```
public void paint(Graphics g)
   {

      g.setColor(Color.red);

        g.drawLine(30, 40, 100, 200);

      g.setColor(Color.blue);

   g.drawRect(200, 30, 50, 70);
            g.fillRect(200, 110, 50, 70);

      g.setColor(Color.magenta);

        g.drawRoundRect (100, 30, 50, 75, 60, 50);
    g.fillRoundRect (100, 110, 50, 75, 60, 50);
```

```
        g.setColor(Color.green);

     g.drawOval(100, 200, 30, 50);
          g.fillOval(200, 200, 30, 50);

        g.setColor (Color.gray);

          g.draw3DRect (300, 30, 50, 75, true);
        g.fill3DRect (300, 30, 50, 75, true);

        g.setColor (Color.cyan);

        g.drawArc(10, 240, 70, 70, 0, 75);
        g.fillArc(10, 240, 70, 70, 0, 75) ;

          int xpoints[] = {50, 25, 25, 75, 75};
      int ypoints[] = {10, 35, 85, 85, 35, 10};
      int num = 5;
      g.drawPolygon(xpoints, ypoints, num);
  }
```

Applet Programming

Introduction

- **An application is a standalone program that can be invoked from the command line.**
- **An applet is a program that runs in the context of a browser session.**
- **A servlet is a program that is invoked on demand on a server program and that runs in the context of a web server process.**

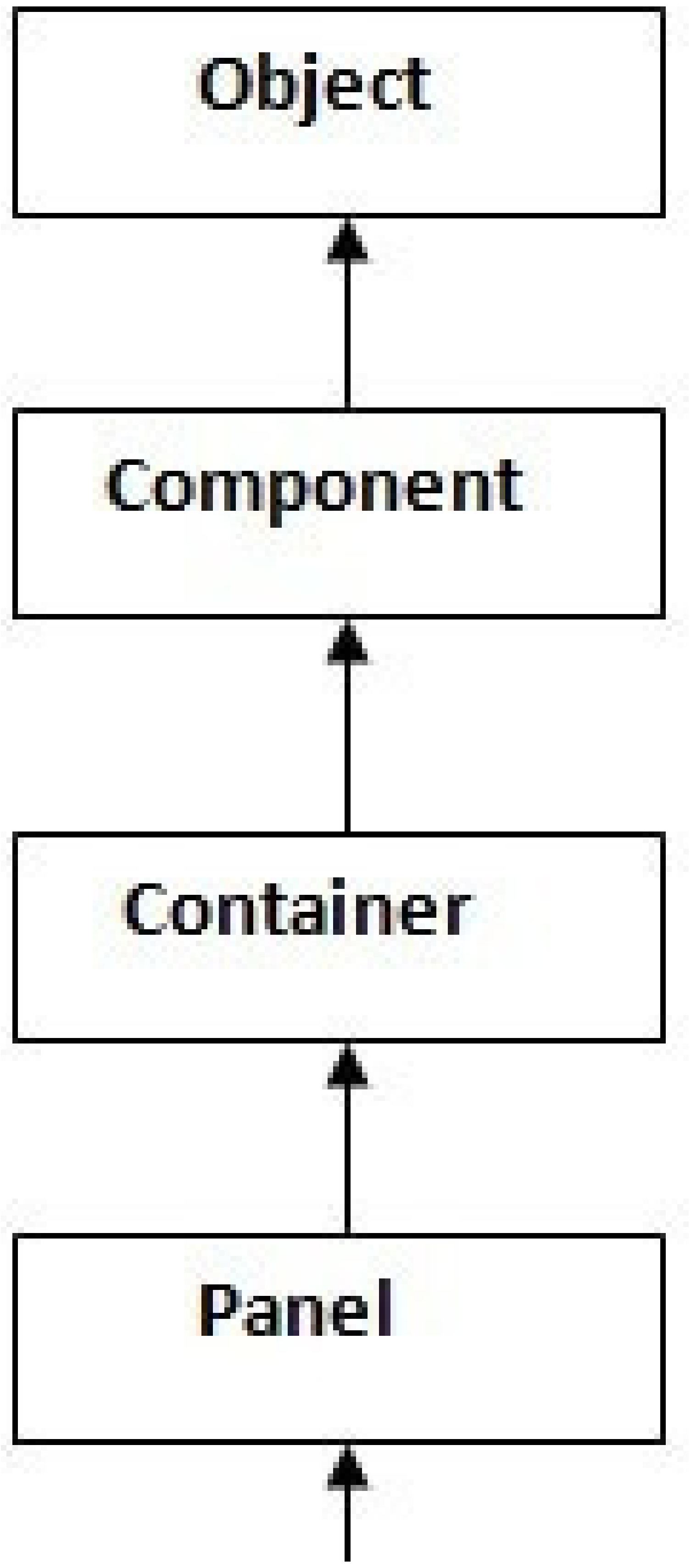
Java Applet

- 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.

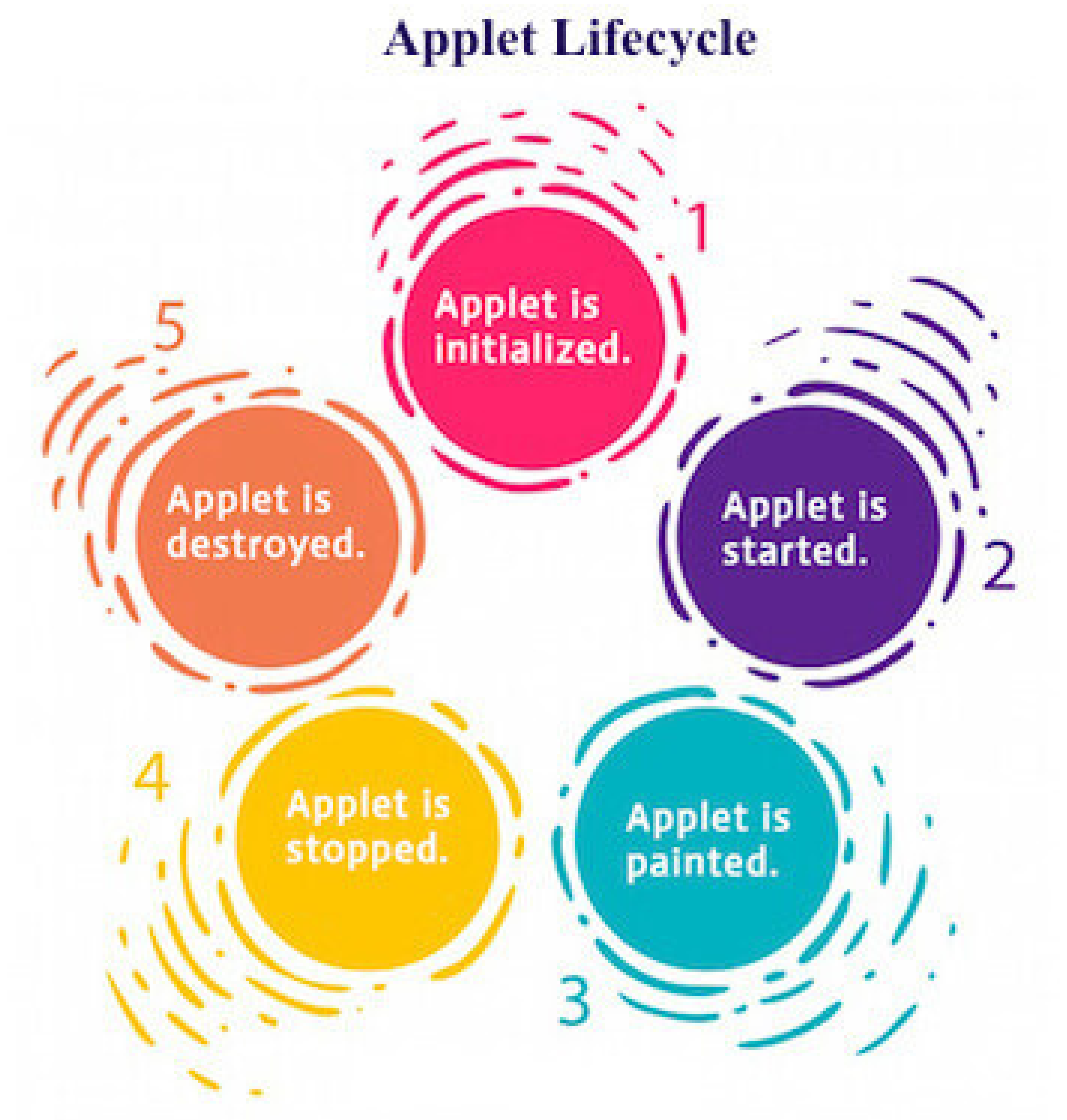
Advantage of Applet

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

Hierarchy of Applet



Lifecycle of Java Applet



Lifecycle methods for Applet:

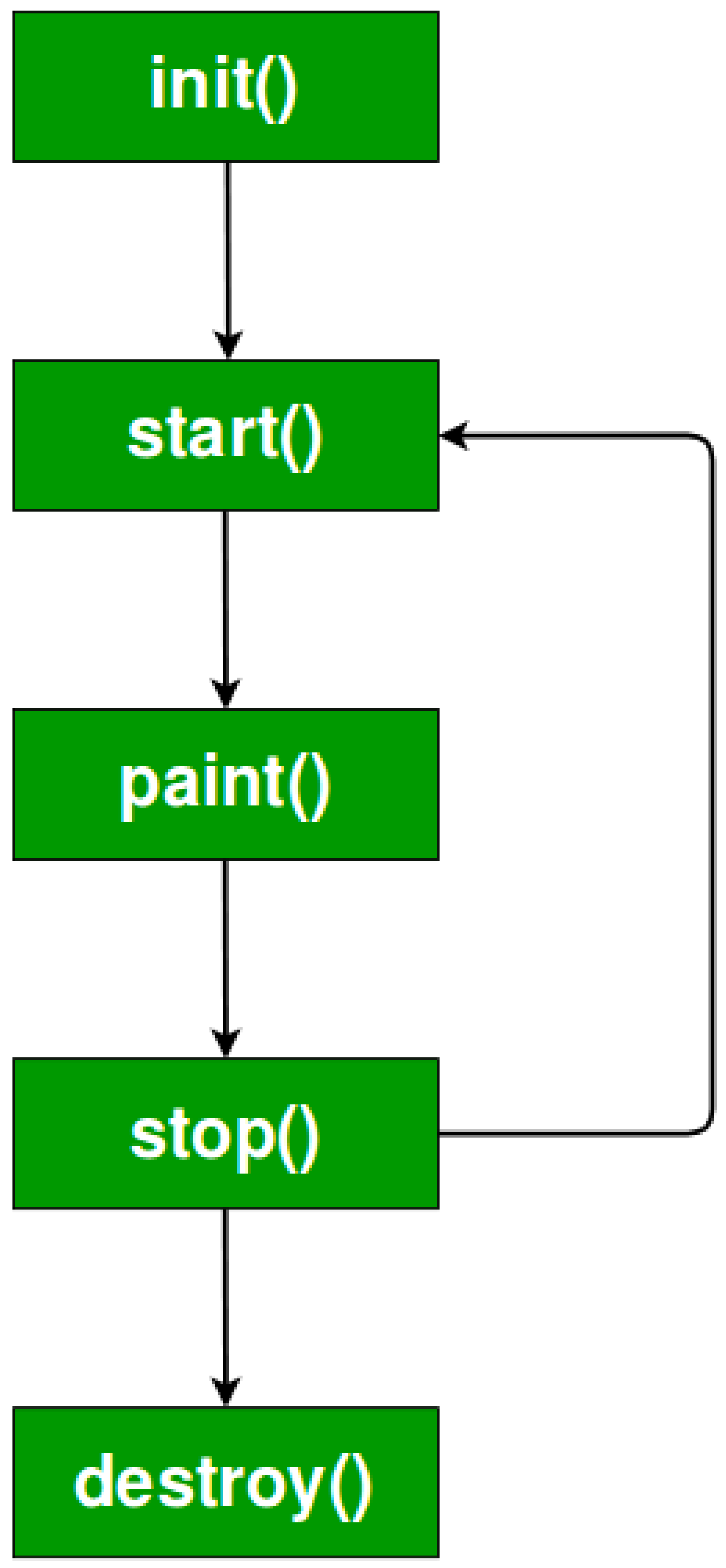
- java.applet.Applet class 4 life cycle methods
- java.awt.Component class provides 1 life cycle methods for an applet.

java.applet.Applet class

- For creating any applet java.applet.Applet class must be inherited.
- It provides 4 life cycle methods of applet.

Lifecycle of Java Applet

- init() method
- start() method
- paint() method
- stop() method
- destroy() method



java.applet.Applet class

1. **public void init():**
   - is used to initialized the Applet. It is invoked only once.
2. **public void start():**
   - is invoked after the init() method or browser is maximized. It is used to start the Applet.
3. **public void stop():**
   - is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.
4. **public void destroy():**
   - is used to destroy the Applet. It is invoked only once.

java.awt.Component class

The Component class provides 1 life cycle method of applet.

public void paint(Graphics g):

- is used to paint the Applet.
- It provides Graphics class object that can be used for drawing oval, rectangle, arc etc.

How to run an Applet?

There are two ways to run an applet

- By html file
- By appletViewer tool (for testing purpose)

Simple example of Applet by appletviewer tool:

- To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it.
- After that run it by: appletviewer First.java.
- Now Html file is not required but it is for testing purpose only.

Simple example of Applet by appletviewer tool:

```
//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",150,150);
}

}
/*
<applet code="First.class" width="300" height="300">
</applet>
*/
```

Simple example of Applet by html file:

1. To execute the applet by html file, create an applet and compile it.
2. After that create an html file and place the applet code in html file.
3. Now click the html file.

first.java

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

myapplet.html

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

Graphics programming

- Java contains support for **graphics that enable programmers** to visually enhance applications

- Java contains many more sophisticated **drawing capabilities** as part of the Java 2D API