

GUI Programming with Java

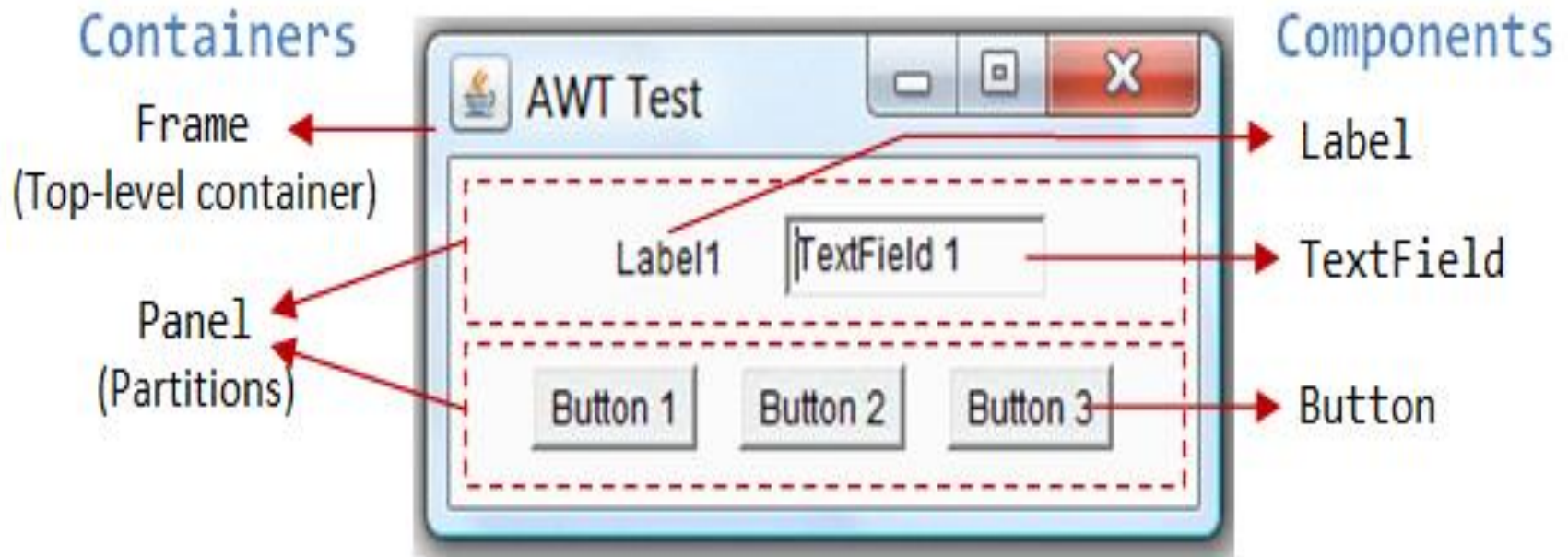
Java Provides the following Frameworks for building GUI-based applications. Those are

- AWT-(Abstract Window Toolkit)
- Swing

AWT-(Abstract Window Toolkit)

- **AWT** (Abstract Window Toolkit) is *an API to develop GUI or window-based applications* in java.
- **AWT** (Abstract Window Toolkit) API consists 12 packages along with 370 classes. Fortunately, only 2 packages - **java.awt** and **java.awt.event** - are commonly-used.
- The **java.awt** package provides various classes such as TextField, Label, TextArea, CheckBox, Choice, List etc.
- The **java.awt.event** package provides Event classes, such as ActionEvent, MouseEvent, KeyEvent and WindowEvent.

AWT Containers and Components



- Container:

The Container is a component in AWT that can contain another components like buttons, textfields, labels etc.

The Container class extends Frame and Panel.

- Frame:

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

- Panel:

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

- Component:

Component is an abstract class that contains various classes such as Button, Label,Checkbox,TextField,Menu and etc.

Useful 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

→ A Simple AWT Application:

- You need a **frame**. There are two ways to create a **frame** in AWT.
- **By extending Frame class** (inheritance)

Ex: class Example **extends** Frame
 {

 }

- **By creating the object of Frame class** (association)

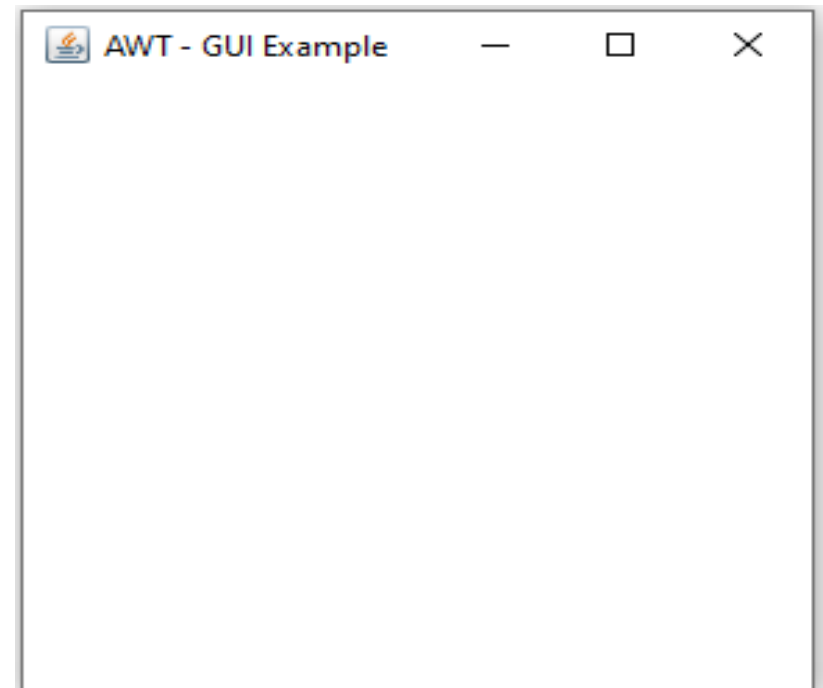
Ex: class Example
 {
 Frame obj=new Frame();

 }

Example:

SimpleGUIExample.java

```
import java.awt.*;  
public class SimpleGUIExample {  
    public static void main(String[] args) {  
        // To create frame  
        Frame f=new Frame("AWT - GUI Example");  
        f.setSize(400,400);  
        f.setLayout(null);  
        f.setVisible(true);  
    }  
}
```



AWT Components or Elements

- Button:

The **Button** class is used to create a labeled button. The application result in some action when the button is pressed.

Syntax:

```
Button b=new Button("Text");
```

(Or)

```
Button b1,b2;
```

```
b1=new Button("Text");
```

```
b.setBounds(50,100,80,30);
```

setBounds(int x,int y,int width,int height)

This method is used to declare location ,width & height of all components of AWT.

Example:

```
setBounds(50,100,80,30);
```

X Y
width Height

- Label:

The **Label** class is a component for placing text in a container. It is used to display a single line of read only text.

Syntax:

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

- TextField:

The **TextField** class is a text component that allows the user to enter a single line text.

Syntax:

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

- TextArea :

The **TextArea** class is a multi line region that allows the user to enter multiple line of text.

Syntax:

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

- Checkbox :

The **Checkbox** 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:

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

- Choice :

The **Choice** class is used to show popup menu of choices. Choice selected by user is shown on the top of a menu.

Syntax:

```
Choice c=new Choice();  
c.add("Item 1");  
c.add("Item 2");  
c.add("Item 3");
```

- Scrollbar :

The **Scrollbar** class is used to create a scrollbar.

Syntax:

```
Scrollbar s=new Scrollbar();
```

- AWT doesn't allow the user to close window directly...
- We need code to close the window

//Code for Close the window

```
addWindowListener(new WindowAdapter(){  
    public void windowClosing(WindowEvent we)  
    {  
        System.exit(0);  
    }  
});
```

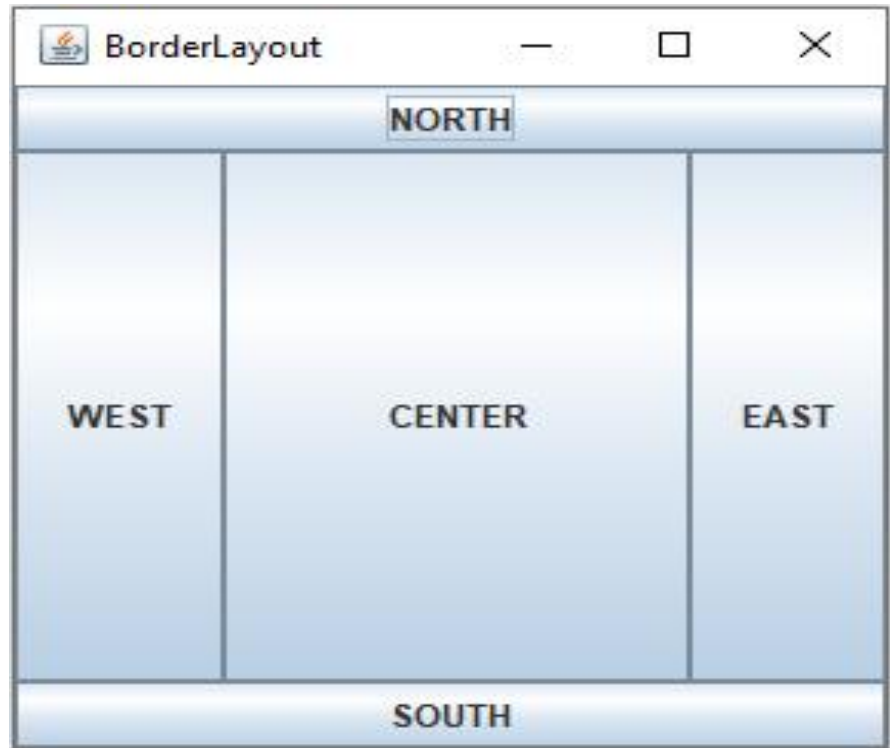
Note: We need to import one package "java.awt.event.*".

Layout Managers

- In Java, Layout Managers is used for arranging the components in order.
- Layout Manager is an interface that is implemented by all the classes of layout managers.
- There are following classes that represents the layout managers
 - `java.awt.BorderLayout`
 - `java.awt.FlowLayout`
 - `java.awt.GridLayout`
 - `java.awt.CardLayout`

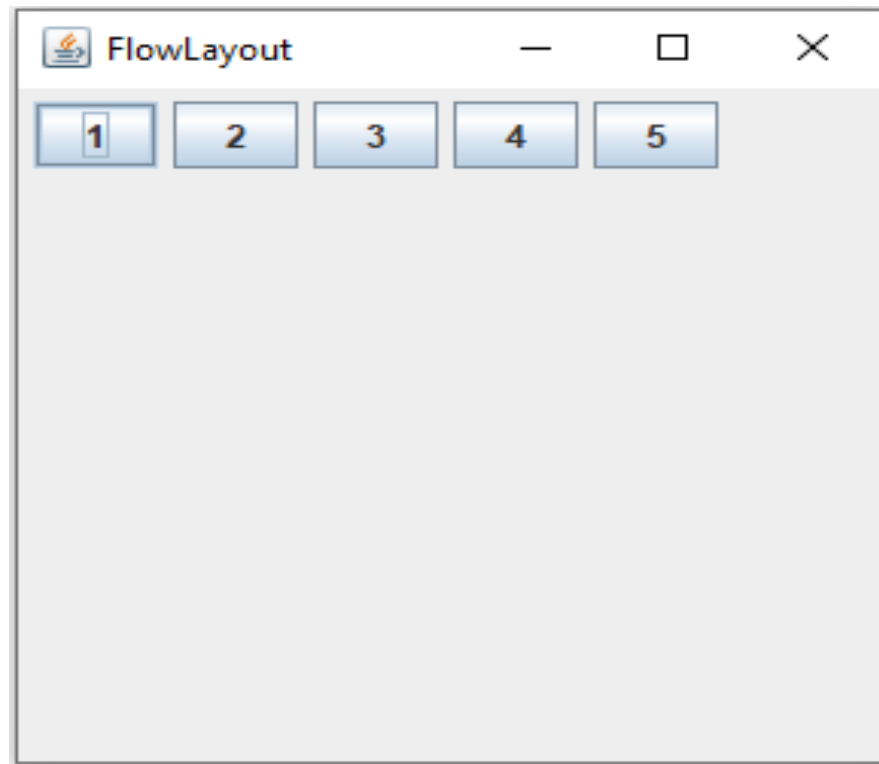
BorderLayout

- The **BorderLayout** is used to arrange the components in five regions: **NORTH**, **SOUTH**, **EAST**, **WEST** and **CENTER**.
- Each region (area) may contain one component only. It is the **default layout** of frame or window.



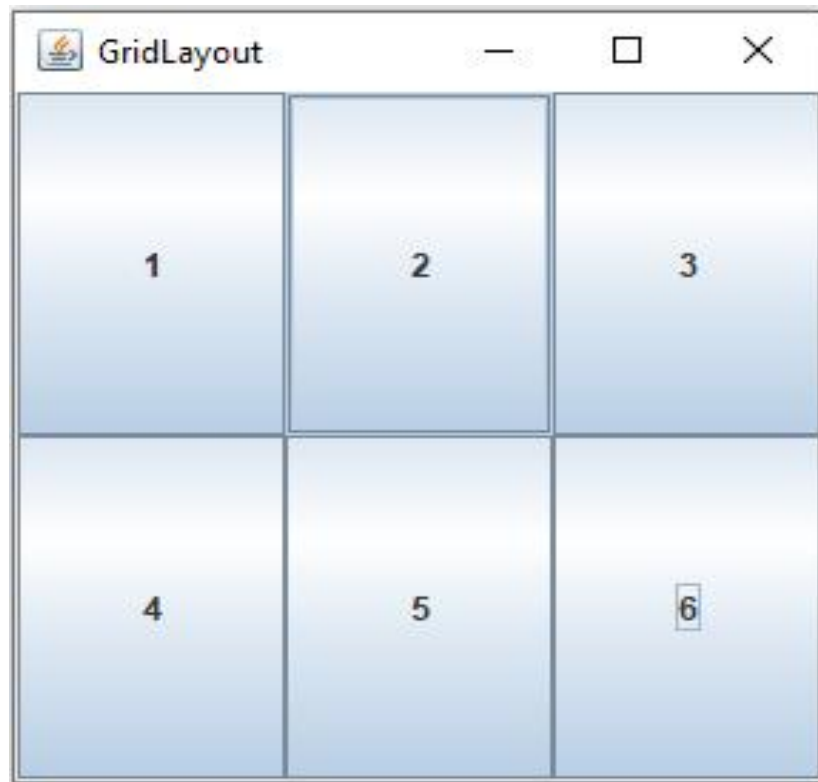
FlowLayout

- The **FlowLayout** is used to arrange the components **in a line**, one after another (in a flow).
- Fields of FlowLayout are **LEFT,RIGHT** and **CENTER**.



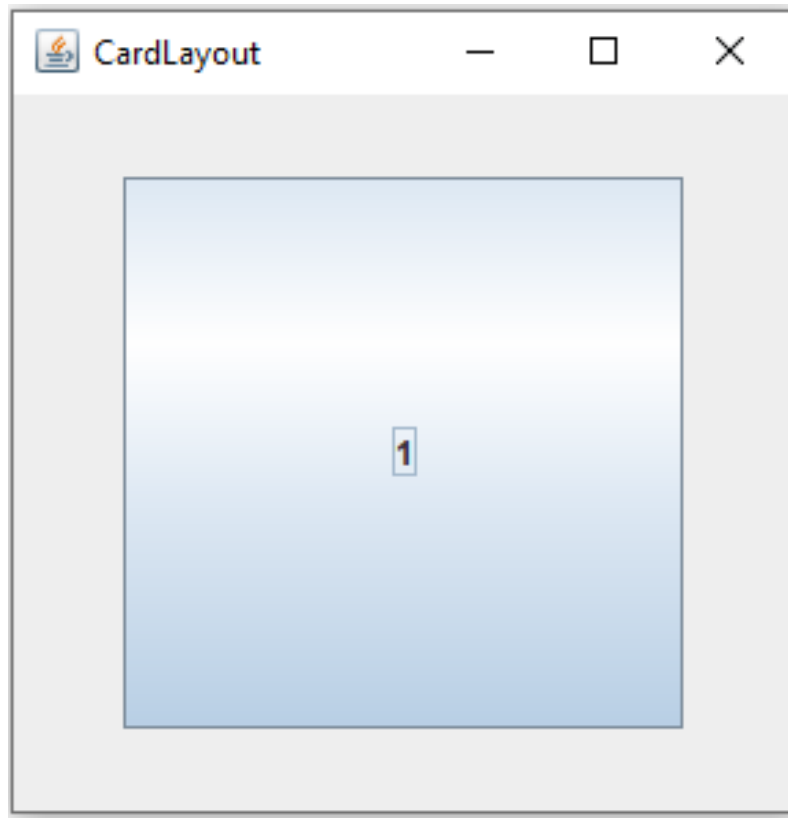
GridLayout

- The **GridLayout** is used to arrange the components in rectangular grid.
- One component is displayed in each rectangle.



CardLayout

- The **CardLayout** class manages the components in such a manner that only one component is visible at a time.
- It treats each component as a card that is why it is known as CardLayout.



Event Handling

- **Event**: Changing the state of an object(component) is known as an event. For example, click on button, moving mouse etc.
- **Events** are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard and selecting an item from list.
- **Def: Event Handling** is the mechanism that controls the event and decides what should happen if an event occurs.
- This mechanism have the code which is known as event handler that is executed when an event occurs.
- The **java.awt.event** package provides many event classes and Listener interfaces for event handling.

Event Classes	Description	Listener Interface
ActionEvent	generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released and also when it enters or exit a component	MouseListener
KeyEvent	generated when input is received from keyboard	KeyListener
ItemEvent	generated when check-box or list item is clicked	ItemListener
TextEvent	generated when value of textarea or textfield is changed	TextListener
MouseWheelEvent	generated when mouse wheel is moved	MouseWheelListener



Steps to perform Event Handling

Following steps are required to perform event handling:

- Register the **component** with the **Listener**.

By using **addActionListener(ActionListener a)**

Example:

```
Button b=new Button("Submit");  
b.setBounds(100,50,80,30);  
b.addActionListener(this);
```

- Provide **event handling code**.

By using **actionPerformed(ActionEvent e)** method of ActionListener Interface.

Example:

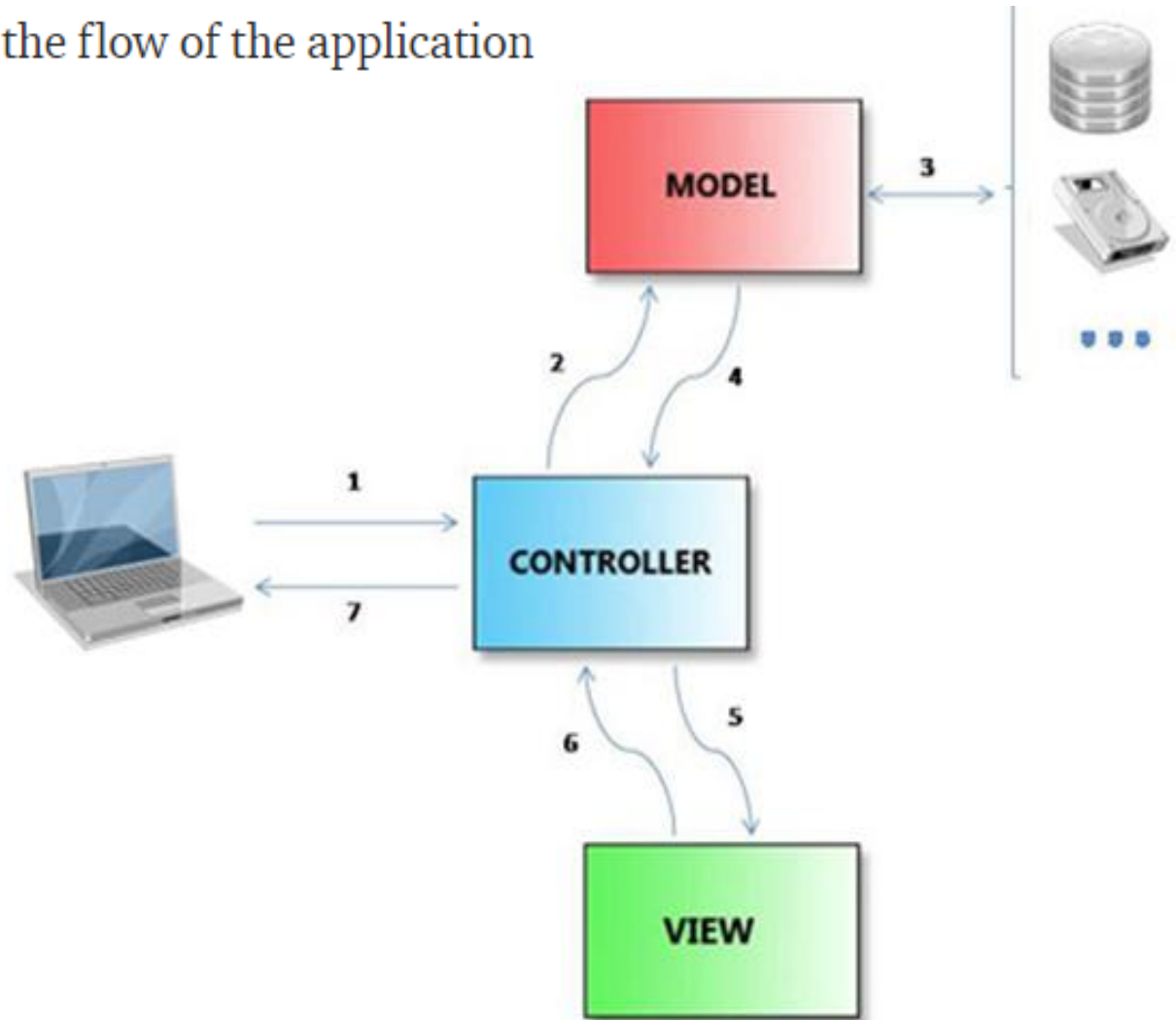
```
Public void actionPerformed(ActionEvent e)  
{  
tf.setText("welcome");  
}
```

Simple Example:

```
import java.awt.*;
import java.awt.event.*;
class EventExample extends Frame implements ActionListener{
    TextField tf;
    EventExample(){
        tf=new TextField(); //create components
        tf.setBounds(60,50,170,20);
        Button b=new Button("click me");
        b.setBounds(100,120,80,30);
        b.addActionListener(this); //register listener & passing current instance
        add(b);add(tf); //add components and set size, layout and visibility
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e){
        tf.setText("Welcome");
    }
    public static void main(String args[]){
        new EventExample();
    }
}
```

MVC architecture:

- *Model* — Represents the business layer of the application
- *View* — Defines the presentation of the application
- *Controller* — Manages the flow of the application



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, JPasswordField etc.

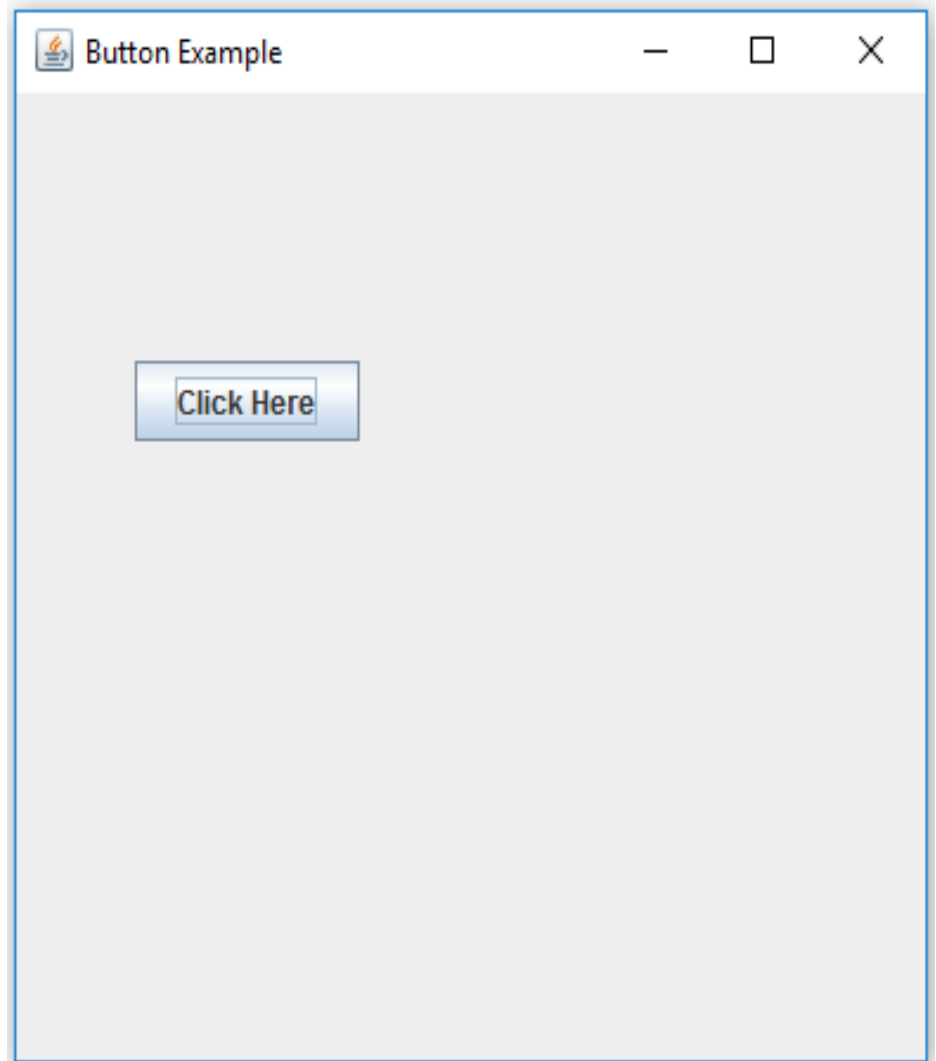
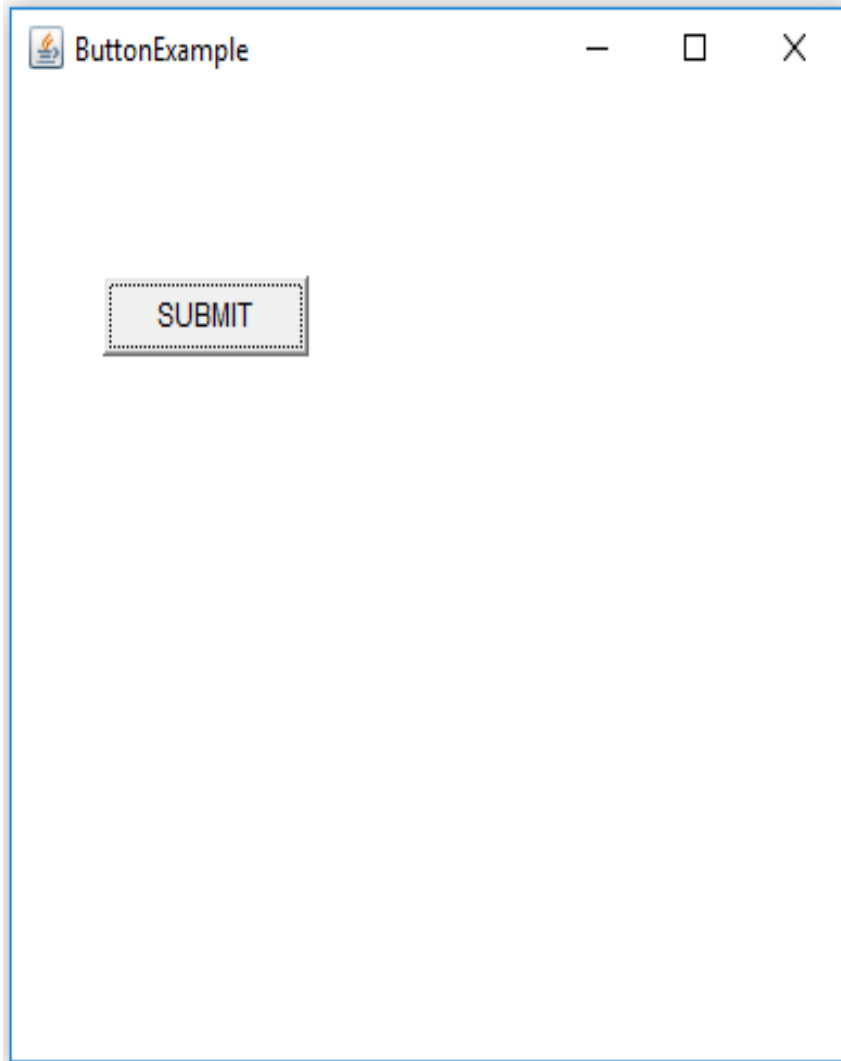
Difference between AWT and Swing

AWT	Swing
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.
The components of AWT are platform dependent.	The components of Swing are platform independent.
AWT components requires java.awt package.	Swing components requires javax.swing package.
AWT components are heavyweight.	Swing components are lightweight.
AWT provides less components than Swing.	Swing provides more components than AWT.
MVC pattern is not supported by AWT.	MVC pattern is supported by Swing.

AWT

vs

Swing



→ 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) {
// creating a frame
JFrame f=new JFrame("Simple Swing Example");
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
}
```

In Constructor()

```
import javax.swing.*;
class SampleExample extends JFrame {
//constructor
SampleExample() {
setSize(300,300);
setLayout(null);
setVisible(true);
setTitle("SimpleExample");
}
public static void main(String args[]){
SampleExample f=new SampleExample();
}
}
```

Components of Swing

JButton:

The JButton class is used to create a labeled button .The application result in some action when the button is pushed.

Example:

```
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.

Example :

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

JTextField:

The JTextField class is a text component that allows the user to enter a single line of text.

Example :

```
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.

Example :

```
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".

Example :

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

JPasswordField:

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

Example :

```
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.

Example :

```
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)

Example :

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

JPanel:

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

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);  
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.

Example :

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

JTabbed Pane:

The JTabbedPane class is used to switch between a group of components by clicking on a tab with a given title or icon.

Example :

```
JPanel p1=new JPanel();  
JPanel p2=new JPanel();  
JTabbedPane tp=new JTabbedPane();  
tp.add("Title",p1);  
tp.add("Title",p2);
```

JMenuBar, JMenu & JMenuItem:

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

Example :

```
JMenuBar mb=new JMenuBar();  
JMenu menu = new JMenu("Menu");  
JMenuItem i1 = new JMenuItem("item1");  
JMenuItem i2 = new JMenuItem("item2");  
JMenuItem i3 = new JMenuItem("item3");
```