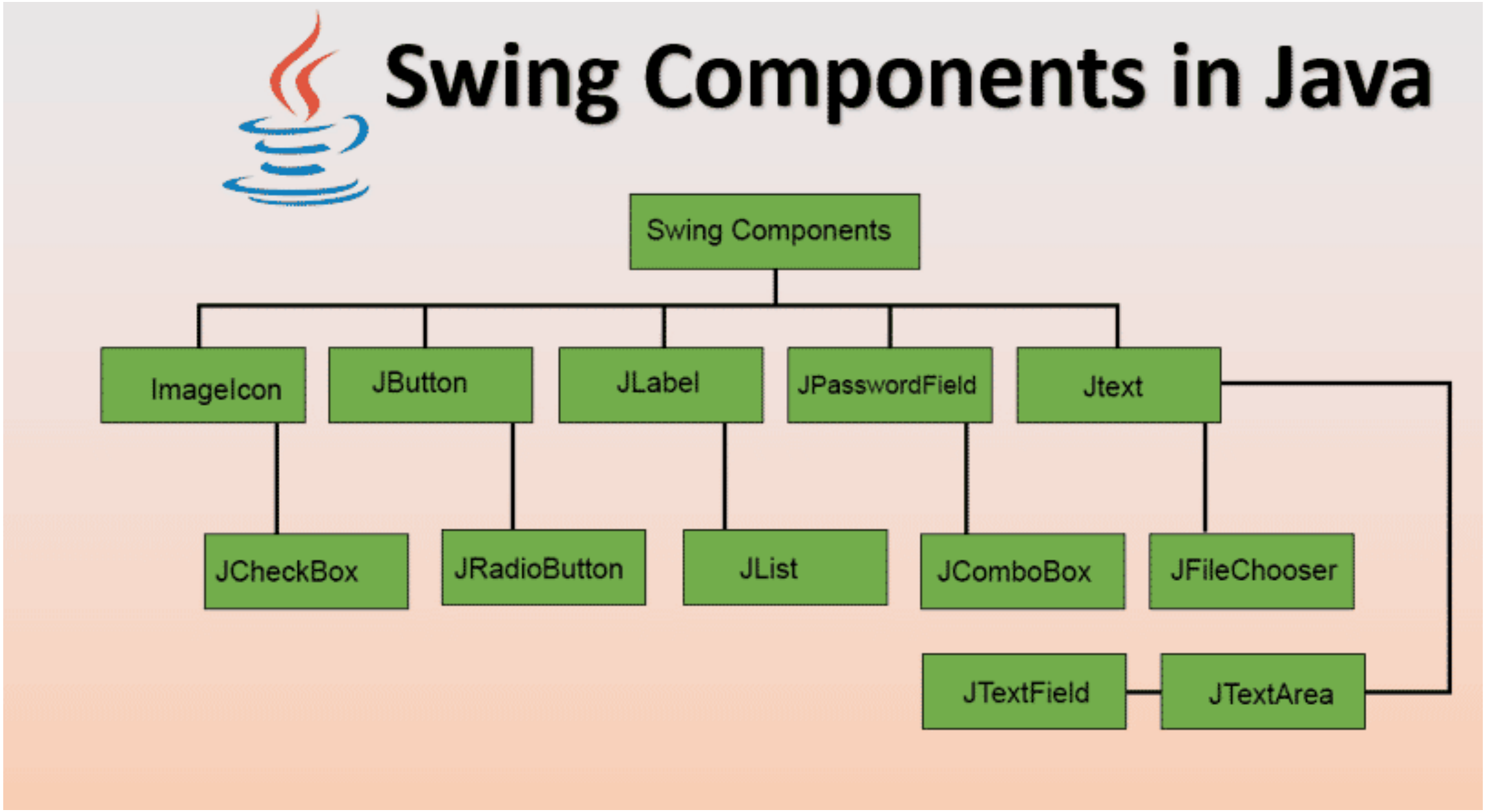
GUI using SWING Event Handling

- **Java Swing tutorial** 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.
- Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes – an API for providing a graphical user interface for Java programs. Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit.
- 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

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)	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.
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**.
6)	Java AWT has comparatively **less functionality as compared to Swing**.	Java Swing has **more functionality as compared to AWT**.
7)	The **execution time of AWT is more than Swing**.	The **execution time of Swing is less than AWT.**

Swing Components



1. JFrame

- The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.
- Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method.

Nested Classes

Modifier and Type	Class	Description
protected class	JFrame.AccessibleJFrame	This class implements accessibility support for the JFrame class.

Fields

Modifier and Type	Field	Description
protected AccessibleContext	accessibleContext	The accessible context property.
static int	EXIT_ON_CLOSE	The exit application default window close operation.
protected JRootPane	rootPane	The JRootPane instance that manages the contentPane and optional menuBar for this frame, as well as the glassPane.
protected boolean	rootPaneCheckingEnabled	If true then calls to add and setLayout will be forwarded to the contentPane.

Constructors

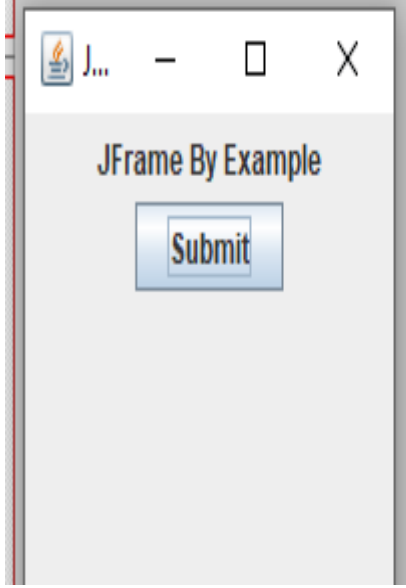
Constructor	Description
JFrame()	It constructs a new frame that is initially invisible.
JFrame(GraphicsConfiguration gc)	It creates a Frame in the specified GraphicsConfiguration of a screen device and a blank title.
JFrame(String title)	It creates a new, initially invisible Frame with the specified title.
JFrame(String title, GraphicsConfiguration gc)	It creates a JFrame with the specified title and the specified GraphicsConfiguration of a screen device.

```
import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
public class JFrameExample {
    public static void main(String args[]) {
        JFrame frame = new JFrame("JFrame Example");
        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout());
        JLabel label = new JLabel("JFrame By Example");
        JButton button = new JButton();
        button.setText("Submit");
        panel.add(label);
        panel.add(button);
        frame.add(panel);
        frame.setSize(200, 300);
       // frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

```
C:\Users\ps\Desktop\java\unit-5\Swing in java>javac JFrameExample.java

C:\Users\ps\Desktop\java\unit-5\Swing in java>java JFrameExample

C:\Users\ps\Desktop\java\unit-5\Swing in java>java JFrameExample
```



JPanel

- The JPanel is a simplest container class. It provides space in which an application can attach any other component. It inherits the JComponents class.
- It doesn't have title bar.

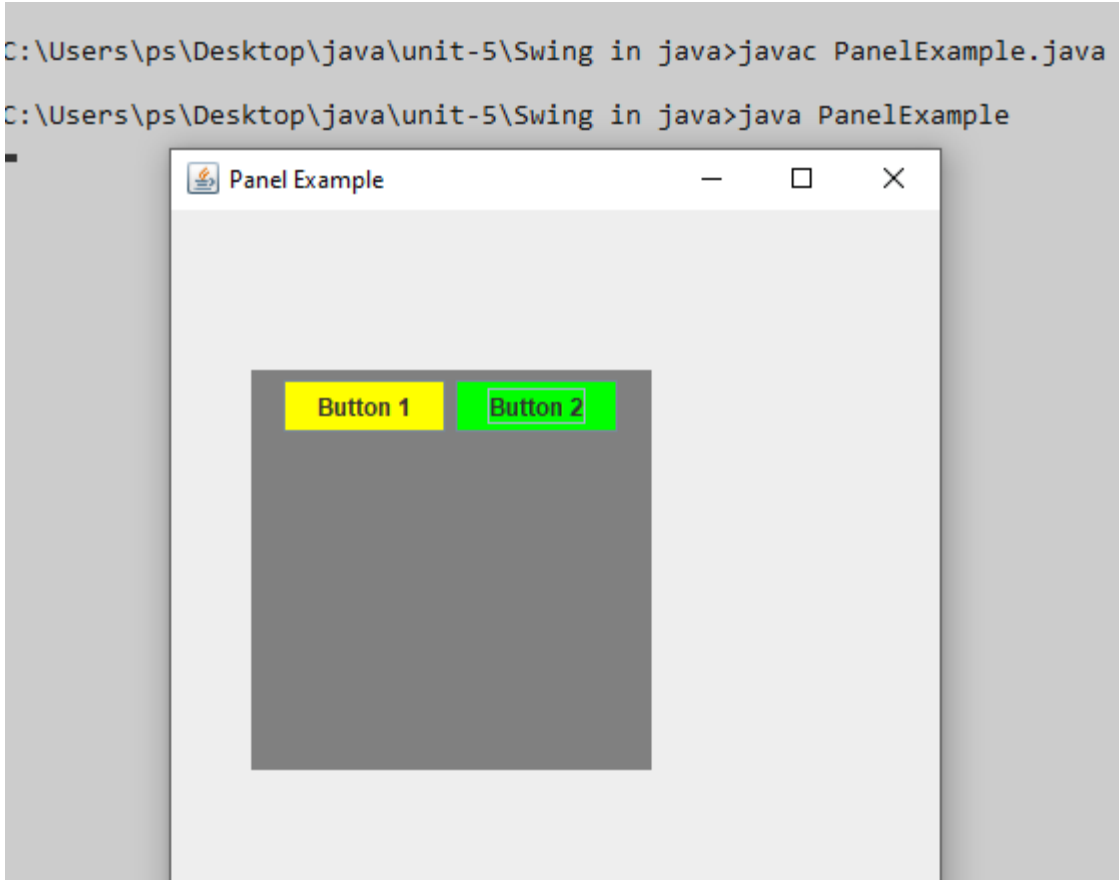
JPanel class declaration

public class JPanel **extends** JComponent **implements** Accessible

Commonly used Constructors

Constructor	Description
JPanel()	It is used to create a new JPanel with a double buffer and a flow layout.
JPanel(boolean isDoubleBuffered)	It is used to create a new JPanel with FlowLayout and the specified buffering strategy.
JPanel(LayoutManager layout)	It is used to create a new JPanel with the specified layout manager.

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



JLabel

- JLabel is a class of java Swing . JLabel is used to display a short string or an image icon. JLabel can display text, image or both .
- JLabel is only a display of text or image and it cannot get focus .
- JLabel is inactive to input events such a mouse focus or keyboard focus.
- By default labels are vertically centered but the user can change the alignment of label.

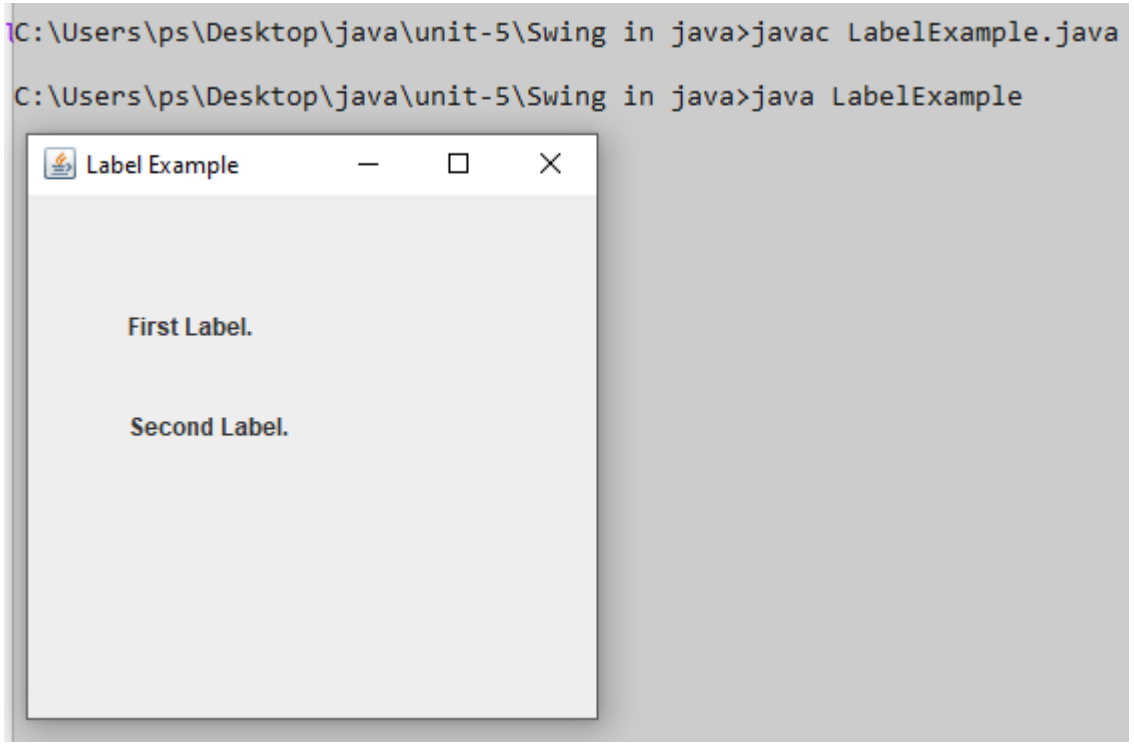
Constructor of the class are :

- **JLabel() :** creates a blank label with no text or image in it.
- **JLabel(String s) :** creates a new label with the string specified.
- **JLabel(Icon i) :** creates a new label with a image on it.
- **JLabel(String s, Icon i, int align) :** creates a new label with a string, an image and a specified horizontal alignment

Methods of JLabel

- **getIcon() :** returns the image that the label displays
- **setIcon(Icon i) :** sets the icon that the label will display to image i
- **getText() :** returns the text that the label will display
- **setText(String s) :** sets the text that the label will display to string s

```
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);
    }
    }
```



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.
- AbstractButton is **an abstract base class for all button components** (JButton , JToggleButton , JCheckBox , JRadioButton , and JMenuItem and its subclasses). Since it provides functionality common to all types of buttons.

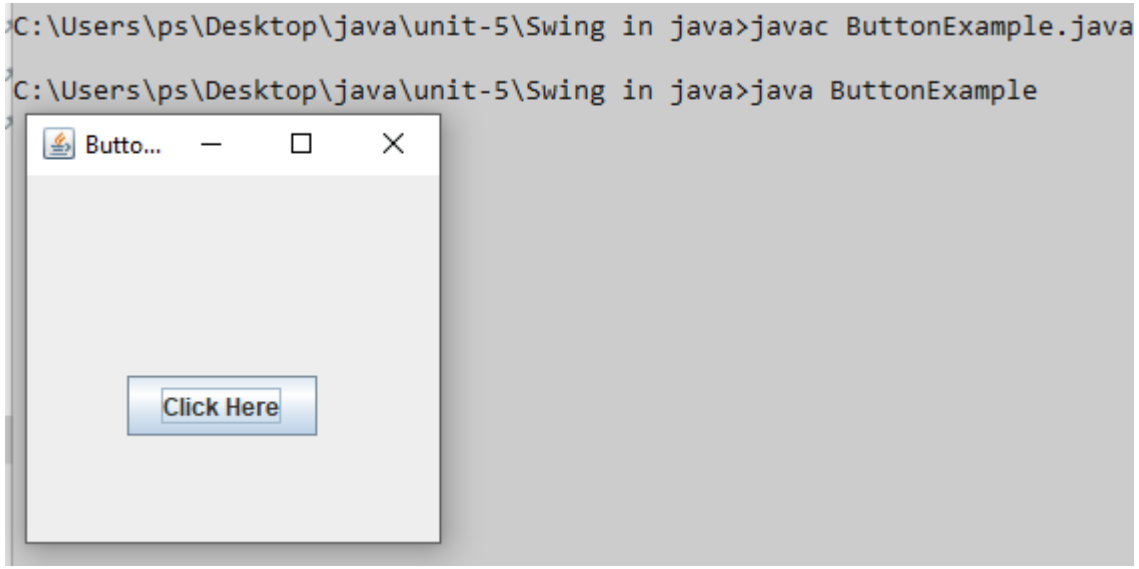
JButton class declaration :

public class JButton **extends** AbstractButton **implements** Accessible

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.

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 setIcon(Icon b)	It is used to set the specified Icon on the button.
Icon getIcon()	It is used to get the Icon of the button.
Void addActionListener(ActionListener a)	It is used to add the action listener to this object.

```
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);
}
}
```



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.

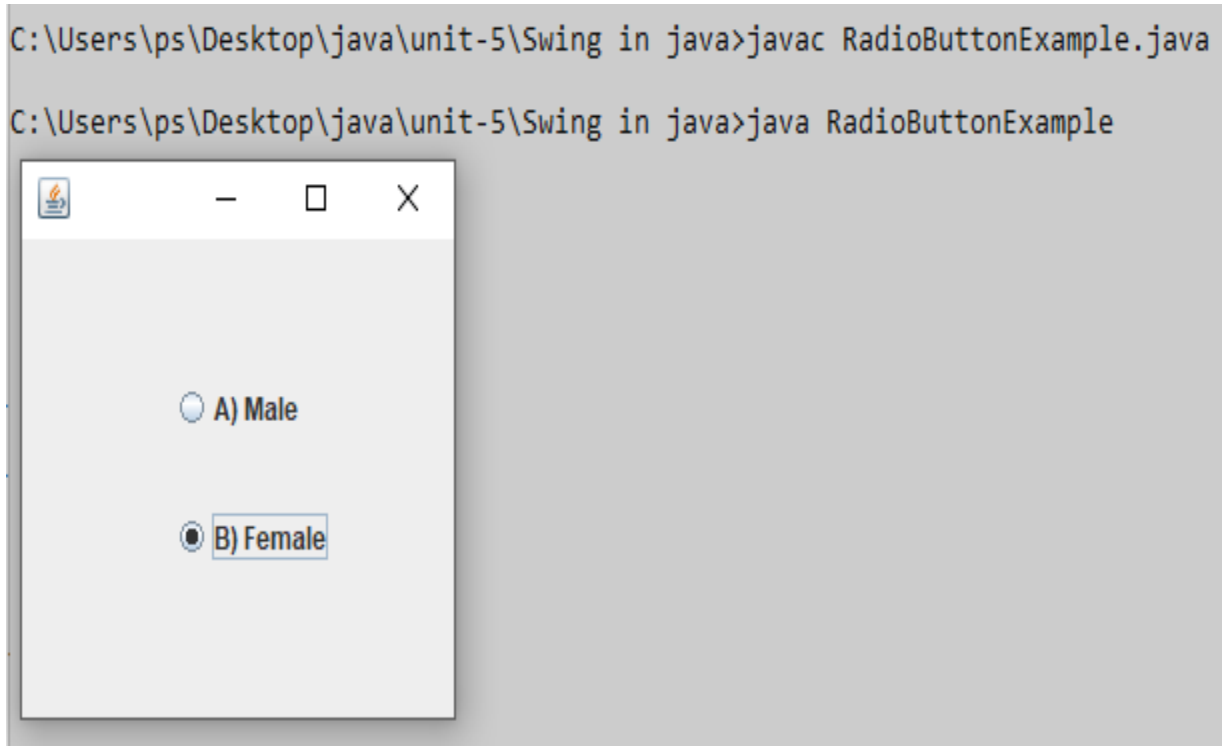
JRadioButton class declaration :

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

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.

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 setIcon(Icon b)	It is used to set the specified Icon on the button.
Icon getIcon()	It is used to get the Icon of the button.
void setMnemonic(int a)	It is used to set the mnemonic on the button.
void addActionListener(ActionListener a)	It is used to add the action listener to this object.

```
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();
}
}
```



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

public class JCheckBox **extends** JToggleButton **implements** Accessible

Constructor of the class are :

1.JCheckBox() : creates a new checkbox with no text or icon

2.JCheckBox(Icon i) : creates a new checkbox with the icon specified

3.JCheckBox(Icon icon, boolean s) : creates a new checkbox with the icon specified and the boolean value specifies whether it is selected or not.

4.JCheckBox(String t) :creates a new checkbox with the string specified

5.JCheckBox(String text, boolean selected) :creates a new checkbox with the string specified and the boolean value specifies whether it is selected or not.

6.JCheckBox(String text, Icon icon) :creates a new checkbox with the string and the icon specified.

7.JCheckBox(String text, Icon icon, boolean selected): creates a new checkbox with the string and the icon specified and the boolean value specifies whether it is selected or not.

Commonly used methods:

1.setIcon(Icon i) : sets the icon of the checkbox to the given icon

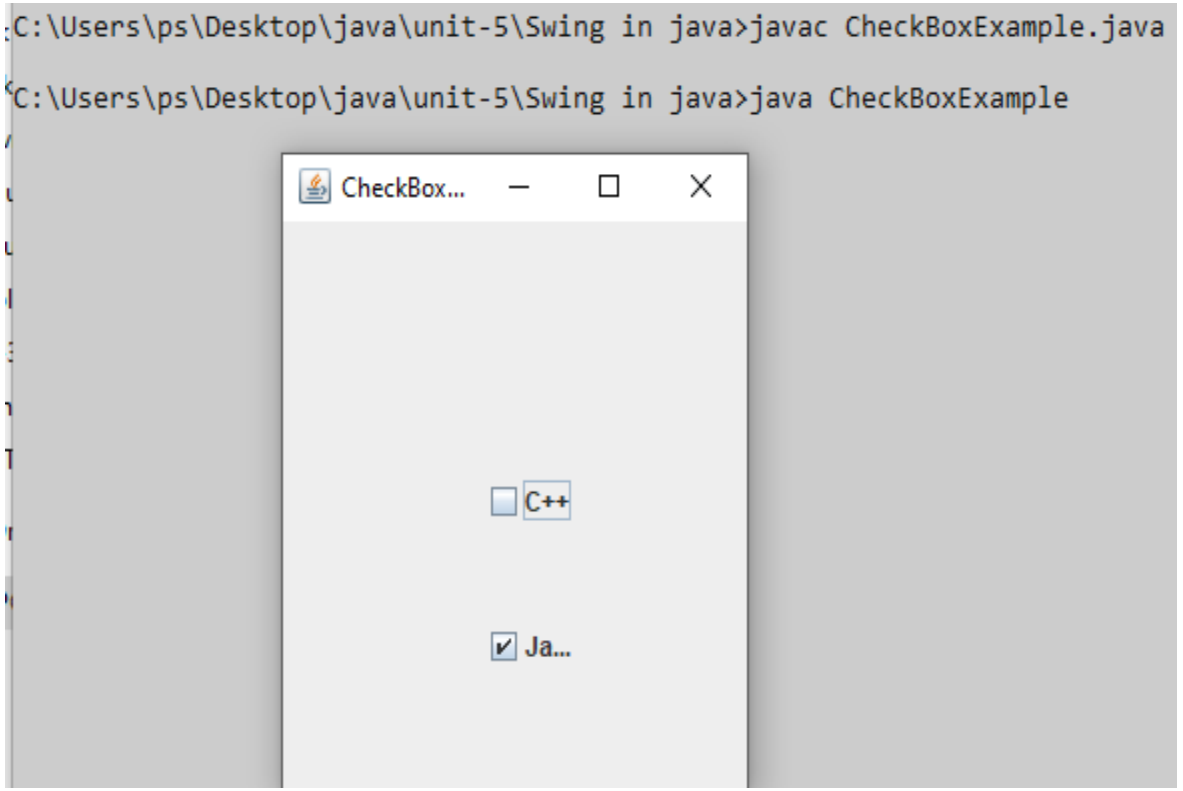
2.setText(String s) :sets the text of the checkbox to the given text

3.setSelected(boolean b) : sets the checkbox to selected if boolean value passed is true or vice versa

4.getIcon() : returns the image of the checkbox

5.getText() : returns the text of the checkbox

```
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, 50,80);
       f.add(checkBox1);
       f.add(checkBox2);
       f.setSize(400,400);
       f.setLayout(null);
       f.setVisible(true);
    }
public static void main(String args[])
   {
   new CheckBoxExample();
   }}
```



JProgressBar

- JProgressBar is a part of Java Swing package.
- JProgressBar visually displays the progress of some specified task.
- JProgressBar shows the percentage of completion of specified task.
- The progress bar fills up as the task reaches it completion.
- In addition to show the percentage of completion of task, it can also display some text .

JProgressBar class declaration :

public class JProgressBar **extends** JComponent **implements** SwingConstants, Accessible

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.

1.int getMaximum() : returns the progress bar’s maximum value.

2.int getMinimum() : returns the progress bar’s minimum value.

3.String getString() : get the progress bar’s string representation of current value.

4.void setMaximum(int n) : sets the progress bar’s maximum value to the value n.

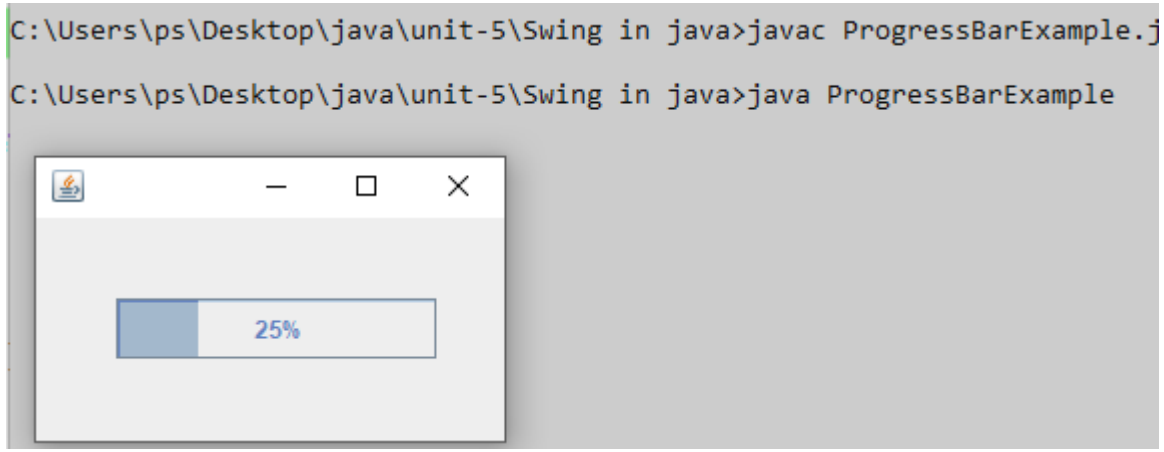
5.void setMinimum(int n) : sets the progress bar’s minimum value to the value n.

6.void setValue(int n) : set Progress bar’s current value to the value n.

7.void setString(String s) : set the value of the progress String to the String s.

```
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(500);
 }catch(Exception e)
  {

  }
}
}
public static void main(String[] args) {
    ProgressBarExample m=new
ProgressBarExample();
    m.setVisible(true);
    m.iterate();
}
}
```



JFileChooser

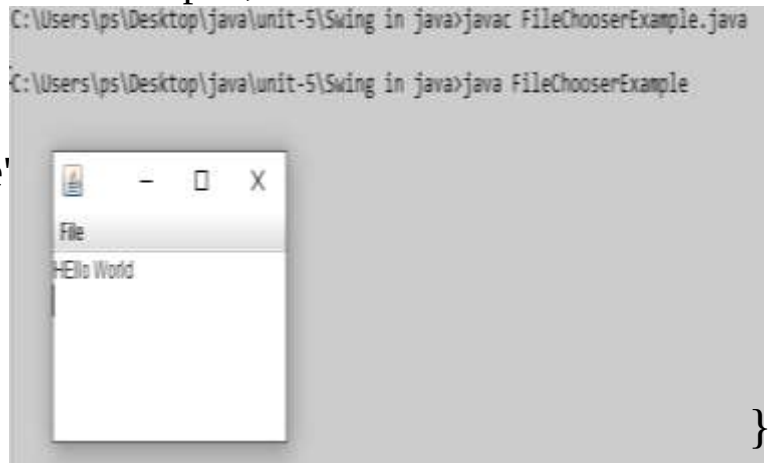
- The object of JFileChooser class represents a dialog window from which the user can select file. It inherits JComponent class.

JFileChooser class declaration

public class JFileChooser **extends** JComponent **implements** Accessible

Constructor	Description
JFileChooser()	Constructs a JFileChooser pointing to the user's default directory.
JFileChooser(File currentDirectory)	Constructs a JFileChooser using the given File as the path.
JFileChooser(String currentDirectoryPath)	Constructs a JFileChooser using the given path.

```
import javax.swing.*;
import java.awt.event.*;
import java.io.*;
public class FileChooserExample extends JFrame
   implements ActionListener{
JMenuBar mb;    JMenu file;    JMenuItem open;
   JTextArea ta;
FileChooserExample(){
open=new JMenuItem("Open File"
open.addActionListener(this);
file=new JMenu("File");
file.add(open);
mb=new JMenuBar();
mb.setBounds(0,0,800,20);
mb.add(file);
ta=new JTextArea(800,800);
ta.setBounds(0,20,800,800);
add(mb);
add(ta);            }
public void actionPerformed(ActionEvent e) {
if(e.getSource()==open){
```




```
if(i==JFileChooser.APPROVE_OPTION){
       File f=fc.getSelectedFile();
       String filepath=f.getPath();
try{
       BufferedReader br=new BufferedReader(new
FileReader(filepath));
       String s1="",s2="";
       while((s1=br.readLine())!=null){
       s2+=s1+"\n";
       }
       ta.setText(s2);
       br.close();
       }catch (Exception ex) {ex.printStackTrace();  }
   }    }    }
public static void main(String[] args) {
   FileChooserExample om=new
FileChooserExample();
           om.setSize(500,500);
           om.setLayout(null);
           om.setVisible(true);

om.setDefaultCloseOperation(EXIT_ON_CLOSE);
}    }
```

JTextField

- JTextField is a part of javax.swing package.
- The class JTextField is a component that allows editing of a single line of text.

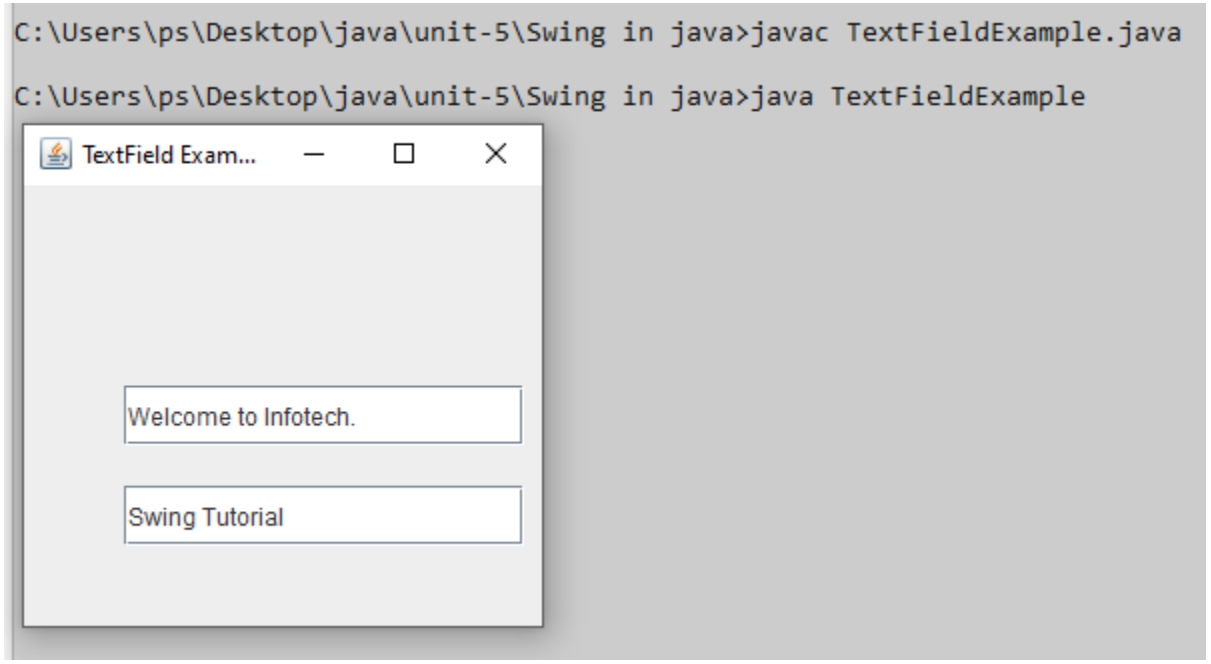
JTextField class declaration :

public class JTextField **extends** JTextComponent **implements** SwingConstants

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

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

```
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 Infotech.");
    t1.setBounds(50,100, 200,30);
    t2=new JTextField("Swing Tutorial");
    t2.setBounds(50,150, 200,30);
    f.add(t1); f.add(t2);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
    }
    }
```

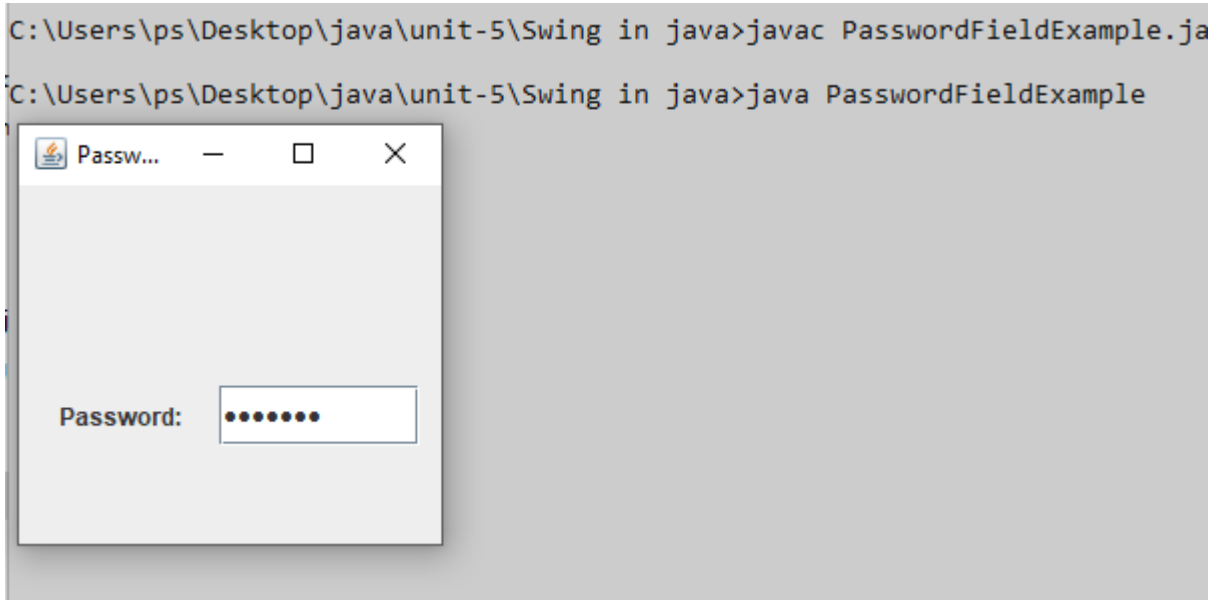


JPasswordField

- PasswordField is a part of javax.swing package .
- The class JPasswordField is a component that allows editing of a single line of text where the view indicates that something was typed by does not show the actual characters.
- JPasswordField inherits the JTextField class in javax.swing package.

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

```
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);
}
}
```



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.
- It is used to edit the text .
- The text in JTextArea can be set to different available fonts and can be appended to new text .
- A text area can be customized to the need of user .

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

1.append(String s) : appends the given string to the text of the text area.

2.getLineCount() : get number of lines in the text of text area.

3.setFont(Font f) : sets the font of text area to the given font.

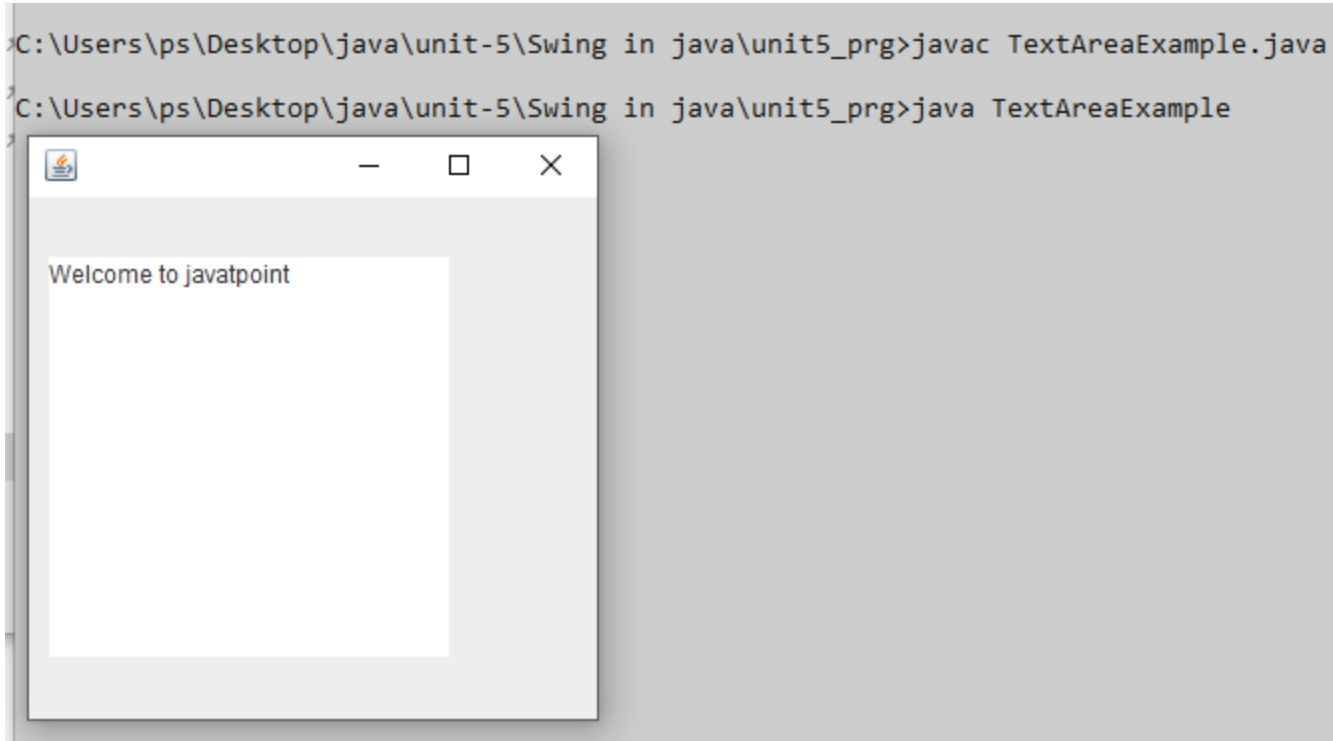
4.setColumns(int c) : sets the number of columns of the text area to given integer.

5.setRows(int r) : sets the number of rows of the text area to given integer.

6.getColumns() : get the number of columns of text area.

7.getRows() : get the number of rows of text area.

```
import javax.swing.*;
public class TextAreaExample
{
    TextAreaExample(){
       JFrame f= new JFrame();
       JTextArea area=new JTextArea("Welcome to javatpoint");
       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 TextAreaExample();
   }}
```



JScrollBar

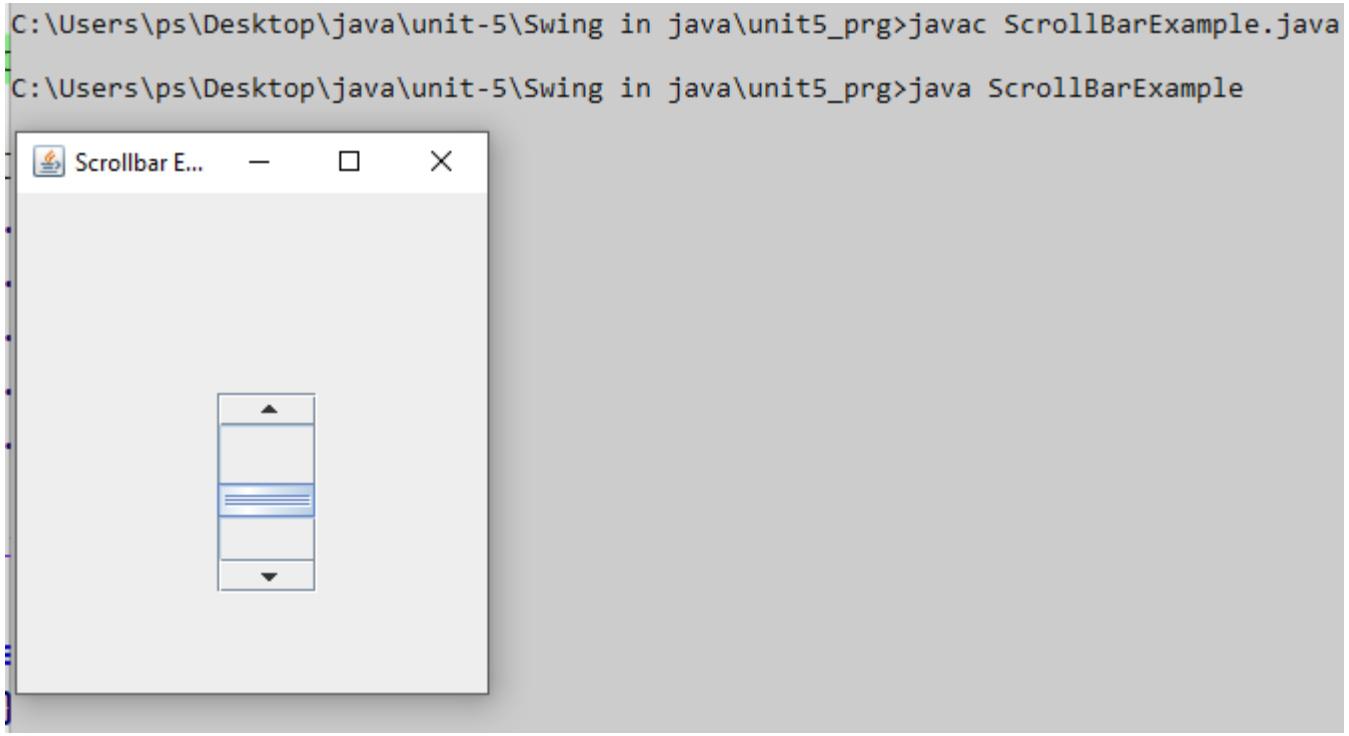
- The object of JScrollbar class is used to add horizontal and vertical scrollbar. It is an implementation of a scrollbar. It inherits JComponent class.

JScrollBar class declaration

public class JScrollBar **extends** JComponent **implements** Adjustable, Ac cessible

Constructor	Description
JScrollBar()	Creates a vertical scrollbar with the initial values.
JScrollBar(int orientation)	Creates a scrollbar with the specified orientation and the initial values.
JScrollBar(int orientation, int value, int extent, int min, int max)	Creates a scrollbar with the specified orientation, value, extent, minimum, and maximum.

```
import javax.swing.*;
class ScrollBarExample
{
ScrollBarExample(){
    JFrame f= new JFrame("Scrollbar Example");
 JScrollBar s=new JScrollBar();
s.setBounds(100,100, 50,100);
f.add(s);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
public static void main(String args[])
{
new ScrollBarExample();
}}
```



JComboBox

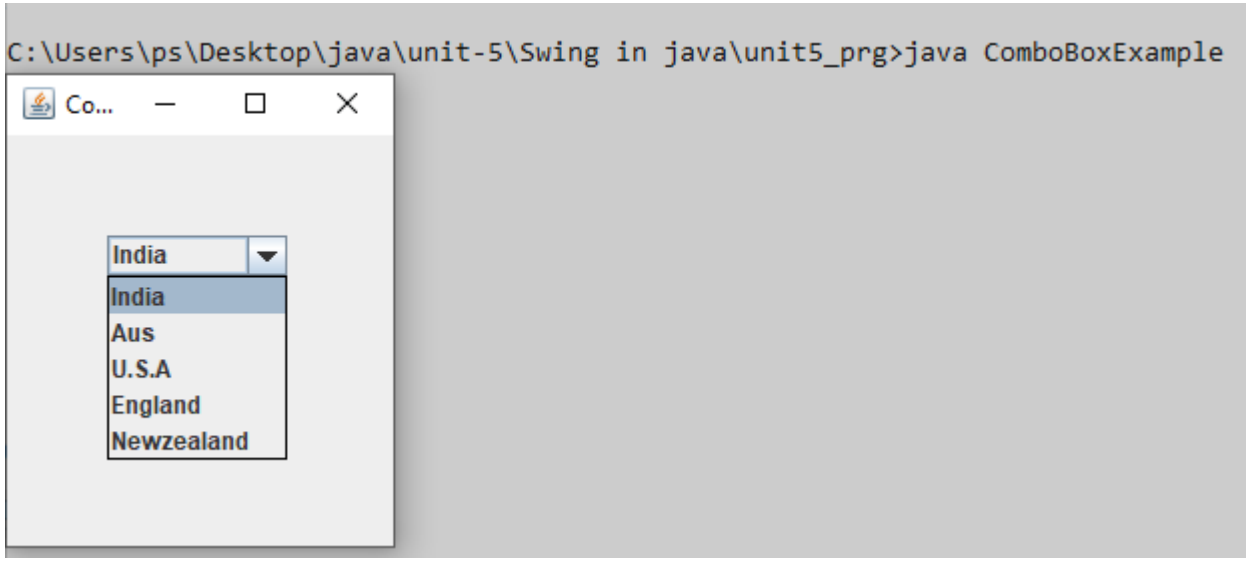
- JComboBox is a part of Java Swing package.
- JComboBox inherits JComponent class .
- JComboBox shows a popup menu that shows a list and the user can select a option from that specified list .
- JComboBox can be editable or read- only depending on the choice of the programmer .

Constructor of the JComboBox are:

1.JComboBox() : creates a new empty JComboBox .
2.JComboBox(ComboBoxModel M) : creates a new JComboBox with items from specified ComboBoxModel
3.JComboBox(E [ ] i) : creates a new JComboBox with items from specified array.
4.JComboBox(Vector items) : creates a new JComboBox with items from the specified vector

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

```
import javax.swing.*;
public class ComboBoxExample {
JFrame f;
ComboBoxExample(){
    f=new JFrame("ComboBox Example");
    String
country[]={"India","Aus","U.S.A","England","Newzeala
nd"};
    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();
}
}
```



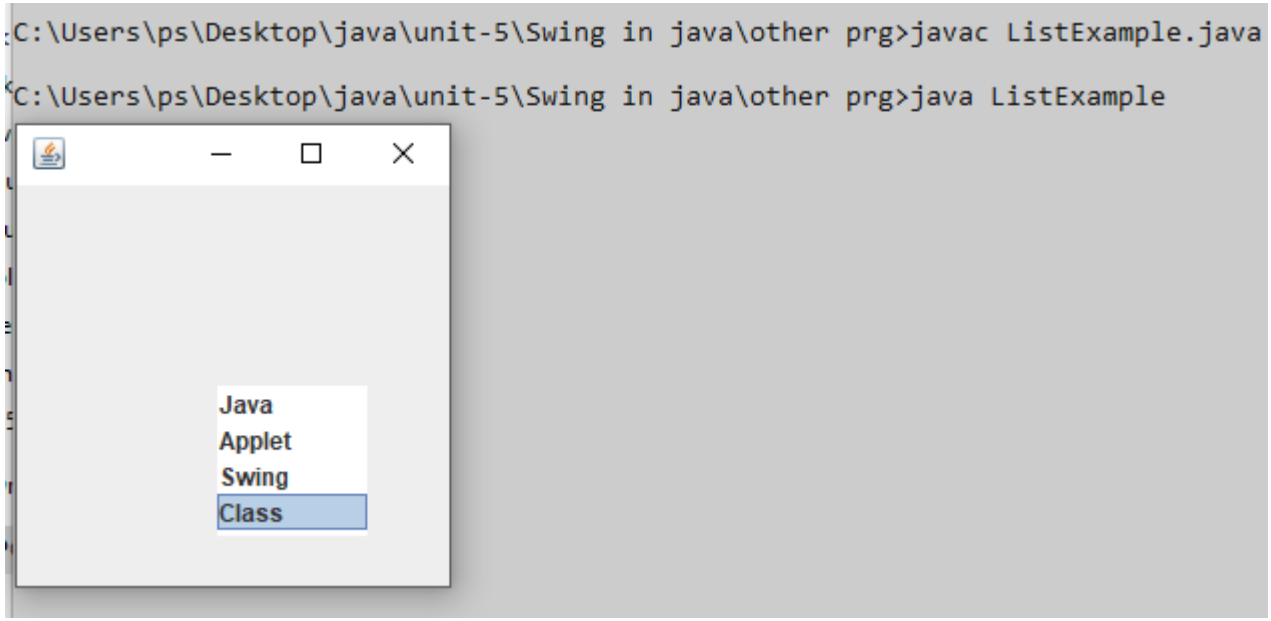
JList

- JList is a component that displays a set of Objects and allows the user to select one or more items .
- JList inherits JComponent class. JList is a easy way to display an array of Vectors .

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

Methods	Description
Void addListSelectionListener(List SelectionListener 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.

```
import javax.swing.*;
public class ListExample
{
    ListExample(){
       JFrame f= new JFrame();
       DefaultListModel l1 = new DefaultListModel();
        l1.addElement("Java");
        l1.addElement("Applet");
        l1.addElement("Swing");
        l1.addElement("Class");
        JList 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();
   }}
```



Menus

- JMenuBar, JMenu and JMenuItems are a part of Java Swing package. JMenuBar is an implementation of menu bar .
- The JMenuBar contains one or more JMenu objects, when the JMenu objects are selected they display a popup showing one or more JMenuItems .
  JMenu basically represents a menu . It contains several JMenuItem Object . It may also contain JMenu Objects (or submenu).

Constructor

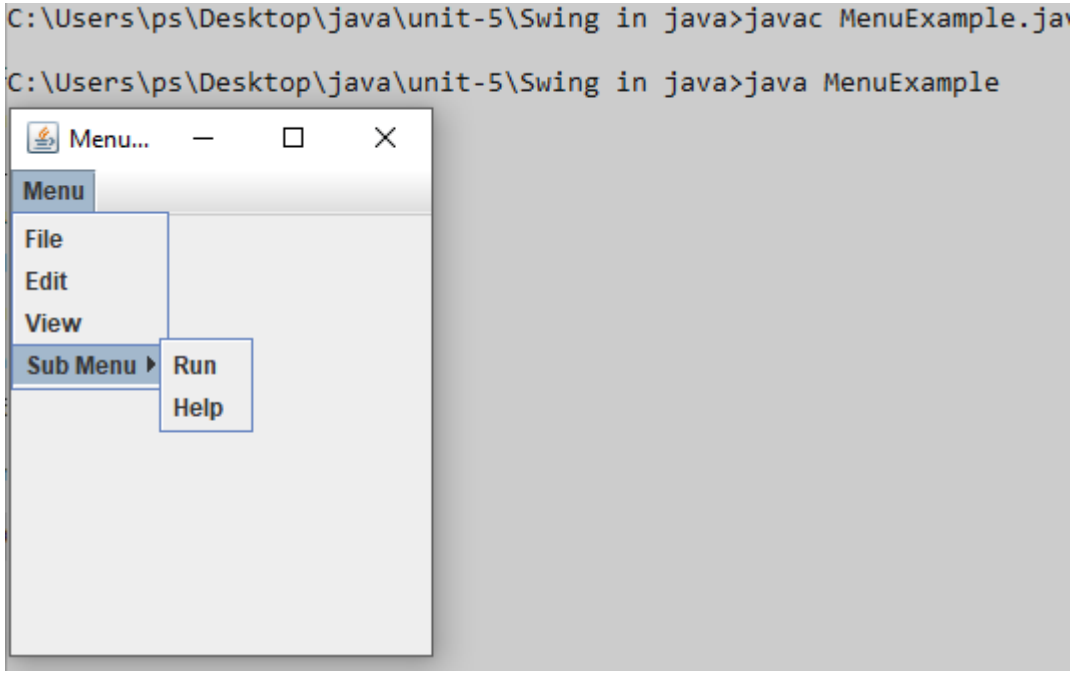
1.JMenuBar() : Creates a new MenuBar.
2.JMenu() : Creates a new Menu with no text.
3.JMenu(String name) : Creates a new Menu with a specified name.
4.JMenu(String name, boolean b) : Creates a new Menu with a specified name and boolean

Commonly used methods:

1.add(JMenu c) : Adds menu to the menu bar. Adds JMenu object to the Menu bar.
2.add(Component c) : Add component to the end of JMenu
3.add(Component c, int index) : Add component to the specified index of JMenu
4.add(JMenuItem menuItem) : Adds menu item to the end of the menu.
5.add(String s) : Creates a menu item with specified string and appends it to the end of menu.
6.getItem(int index) : Returns the specified menuitem at the given index

```
import javax.swing.*;
class MenuExample
{
        JMenu menu, submenu;
        JMenuItem i1, i2, i3, i4, i5;
        MenuExample(){
        JFrame f= new JFrame("Menu and MenuItem Example");
        JMenuBar mb=new JMenuBar();
        menu=new JMenu("Menu");
        submenu=new JMenu("Sub Menu");
        i1=new JMenuItem("File");
        i2=new JMenuItem("Edit");
        i3=new JMenuItem("View");
        i4=new JMenuItem("Run");
        i5=new JMenuItem("Help");
        menu.add(i1); menu.add(i2); menu.add(i3);
        submenu.add(i4); submenu.add(i5);
        menu.add(submenu);
        mb.add(menu);
        f.setJMenuBar(mb);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
}
```

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



Introduction to Event Handling

- Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling.

Classification of Events

- Foreground Events
- Background Events

1. Foreground Events

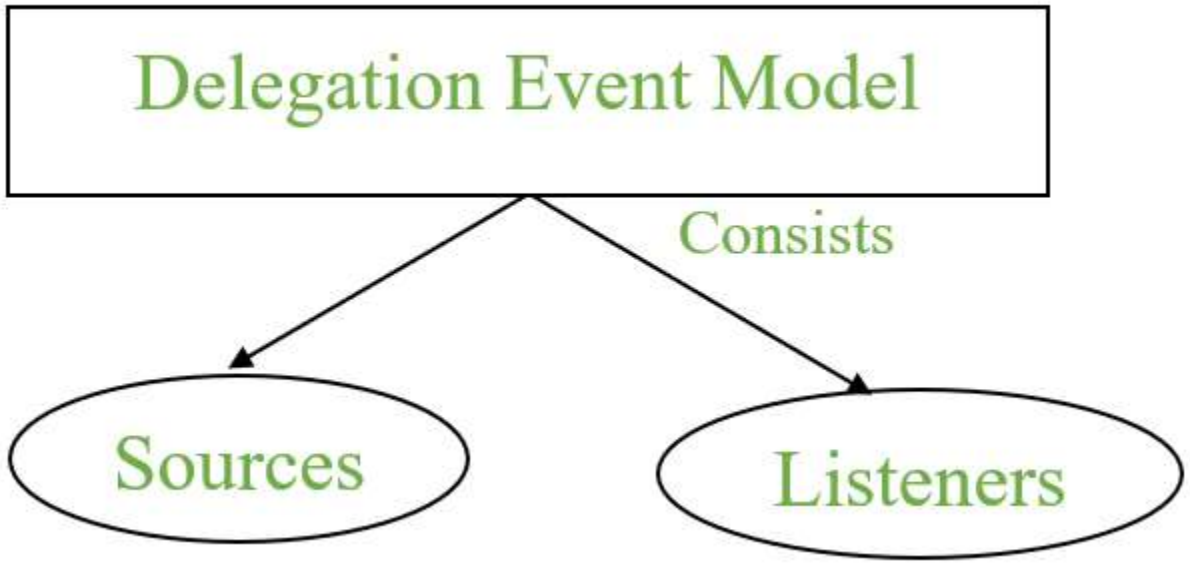
Foreground events are the events that require user interaction to generate, i.e., foreground events are generated due to interaction by the user on components in Graphic User Interface (**GUI**). Interactions are nothing but clicking on a button, scrolling the scroll bar, cursor moments, etc.

2. Background Events

Events that don’t require interactions of users to generate are known as background events. Examples of these events are operating system failures/interrupts, operation completion, etc.

Delegation Event model

- It has Sources and Listeners.



•**Source:** Events are generated from the source. There are various sources like buttons, checkboxes, list, menu-item, choice, scrollbar, text components, windows, etc., to generate events.
•**Listeners:** Listeners are used for handling the events generated from the source. Each of these listeners represents interfaces that are responsible for handling events.
To perform Event Handling, we need to register the source with the listener.
Registering the Source With Listener
Different Classes provide different registration methods.
Syntax: `addTypeListener()`

where Type represents the type of event.
Example 1: For **KeyEvent** we use *addKeyListener()* to register.
Example 2:that For **ActionEvent** we use *addActionListener()* to register.

Note: As Interfaces contains abstract methods which need to implemented by the registered class to handle events.

Event Class	Listener Interface	Description
ActionEvent	ActionListener	An event that indicates that a component-defined action occurred like a button click or selecting an item from the menu-item list.
AdjustmentEvent	AdjustmentListener	The adjustment event is emitted by an Adjustable object like Scrollbar.
ComponentEvent	ComponentListener	An event that indicates that a component moved, the size changed or changed its visibility.
ContainerEvent	ContainerListener	When a component is added to a container (or) removed from it, then this event is generated by a container object.
TextEvent	TextListener	An event that occurs when an object's text changes.
WindowEvent	WindowListener	An event which indicates whether a window has changed its status or not.

FocusEvent	FocusListener	These are focus-related events, which include focus, focusin, focusout, and blur.
ItemEvent	ItemListener	An event that indicates whether an item was selected or not.
KeyEvent	KeyListener	An event that occurs due to a sequence of keypresses on the keyboard.
MouseEvent	MouseListener & MouseMotionListener	The events that occur due to the user interaction with the mouse (Pointing Device).
MouseWheelEvent	MouseWheelListener	An event that specifies that the mouse wheel was rotated in a component.

Different interfaces consists of different methods which are specified below.

Listener Interface	Methods
ActionListener	•actionPerformed()
AdjustmentListener	•adjustmentValueChanged()
ComponentListener	•componentResized() •componentShown() •componentMoved() •componentHidden()
ContainerListener	•componentAdded() •componentRemoved()
FocusListener	•focusGained() •focusLost()

ItemListener	•itemStateChanged()
KeyListener	•keyTyped() •keyPressed() •keyReleased()
MouseListener	•mousePressed() •mouseClicked() •mouseEntered() •mouseExited() •mouseReleased()
MouseMotionListener	•mouseMoved() •mouseDragged()
MouseWheelListener	•mouseWheelMoved()
TextListener	•textChanged()
WindowListener	•windowActivated() •windowDeactivated() •windowOpened() •windowClosed() •windowClosing() •windowIconified() •windowDeiconified()

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener