Introduction to Java Swing

- Java Swing is used to build GUI for Java applications.
- There are two types of swing controls :
  - Containers (Eg: **JFrame, JPanel**)
  - Components (individual components that are placed inside containers)
    - Eg:- **JLabel, JTextField, JButton, JCheckBox, JradioButton, JComboBox, JList, JPasswordField, JTextArea, JTable** etc.
- There are separate classes for each swing control. Those classes are defined in the package **javax.swing**
- When Users of Java applications directly interact with GUI using mouse or keyboard, Events will be generated.
- Corresponding to each generated event , there is an **Event Class** and an **Event Listener** interface. Those classes and interfaces are defined in the package **java.awt.event** or in **javax.swing.event**

How to build a GUI using **JButton & JLabel?**

- **Import the packages** –
  - **javax.swing** - it contains the classes for the component controls and the container **JFrame**
  - **java.awt.event** - it contains the event classes and event listeners for event handling
- Create a **GUI class** that implements **ActionListener** interface
- Inside the class, declare a **JButton object** and **JLabel object**
- Define **Constructor of the class** that does the following:
  - **create a JFrame,** set its size, set the layout to null
  - create and initialize **JButton & JLabel objects** (set the boundaries x, y, width and height within the frame)
  - **Add button and label into frame** in a proper layout
  - **Call addActionListener()** method of Button object t**o register for receiving the eventobject generated** when the user click on the button control
  - make the V**isible property of JFrame to true** (show the frame)
- Define the **public void actionPerformed(ActionEvent e)** method of the **ActionListener** interface to handle the events generated from the Button object
- Create **mian() function** and instantiate an object of GUI class (i. e. **create an object of the GUI Class**)

Demonstrating the above steps with an exercise that displays a message in the label box while clicking on the Button

```java
// Introducing JButton and JLabel
import  javax.swing.*;
import java.awt.event.*;
class GUISample1 implements ActionListener
{
     JButton  btnMsg;
     JLabel lblMsg;
     GUISample1()
     {
        JFrame  frm = new JFrame("Button & Label Interface");
        btnMsg = new JButton("Click Me");
        lblMsg = new JLabel("...........");
        frm.setSize(300,300);
        frm.setLayout(null);
        frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        btnMsg.setBounds(50, 100, 100, 50);
        lblMsg.setBounds(50, 160, 100, 50);
        frm.add(btnMsg);
        frm.add(lblMsg);
        btnMsg.addActionListener(this);
        frm.setVisible(true);
     }
     public void actionPerformed(ActionEvent ae)
     {
         if(btnMsg.getText().equals("Click Me"))
         {
                lblMsg.setText("Clicked !!!");
                btnMsg.setText("Clear Me");
         }
         else
         {
                lblMsg.setText("...........");
                btnMsg.setText("Click Me");
         }
     }
     public static void main(String args[])
     {
        new GUISample1();
     }
}
```

Methods of JButton class

1	**JButton(String str)**	**Constructor to create JButton object**
2	**setBounds(x, y, width, height)**	**To set the position and size of the button in the frame**
3	**setText(string)**	**To change the text message on the button**
4	**getText()**	**To get the text message on the button**
5	**addActionListener(listener_object)**	**To register for the action event**

Methods of JLabel class

1	**JLabel(String str)**	Constructor ot create label object
2	**getText()**	To read the string message on the label object
3	**setText(string)**	to set the string message on a label object
4	set**Bounds(x, y, width, height)**	

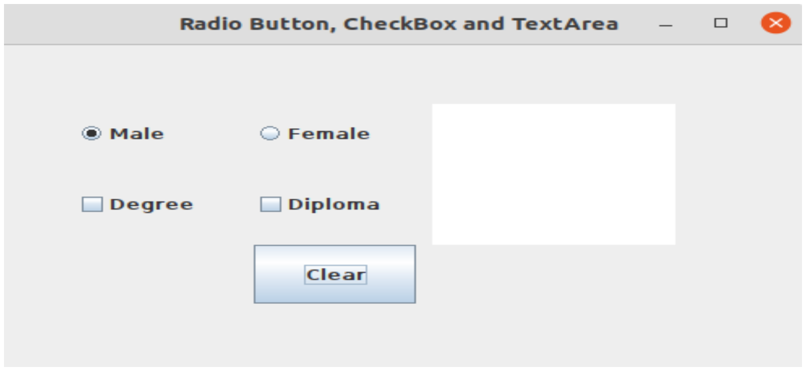
Methods of JFrame class

1	**JFrame(String)**	constructor to create a JFrame object with title
2	**setSize(width, height)**	set the size of the frame window
3	**setLayout( )**	decide which Layout Manager to use or manual layout
4	**add(component_object)**	to add component object into a frame
5	**setVisible(true)**	to show the frame on screen
6	**setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)**	To close the application when we close the frame

How to build a GUI using **JRadioButton, JCheckBox , JTextArea**

- **Import the packages** –
  - **javax.swing** - it contains the classes for the component controls and the container **JFrame**
  - **java.awt.event** - it contains the event classes and event listeners for event handling
- Create a **GUI class** that implements **ActionListener** and **ItemListener** interfaces
- Inside the class, declare necessary **JRadioButton** objects, **JCheckBox** objects and **JTextArea** objects
- Define **Constructor of the class** that does the following:
  - **create a JFrame,** set its size, set the layout to null
  - create and i**nitialise JRadioButton** objects, **JCheckBox** objects and **JTextArea** objects(set the boundaries **x, y, width** and **height** within the frame)
  - **Add the created componet objects into the frame** in a proper layout
  - Call **addActionListener()** method of Button object t**o register for receiving the eventobject generated** when the user click on the buttoncontrol (Only if you are added a **JButton** in your frame)
  - Call **addItemListener()** method of RadioButton and CheckBox to receive their change events
  - make the V**isible property of Jframe to true** (show the frame)
- Define the **public void actionPerformed(ActionEvent e)** of the **ActionListener** interface to handle the events generated from the Button object
- Define the **public void itemStateChanged(ItemEvent e)** of the **ItemListener** interface to handle the events generated from the **RadioButton** and **CheckBox** object
- Create **mian()** function and instantiate an object of GUI class (i. e. **create an object of the GUI Class**)

Sample Program:- Demonstrating the above steps with an exercise that displays a the state of a RadioButton group for choosing gender and two CheckBoxes to select the qualifications. A Button for clearing the data is also included . User Interface will be as shown below:



```java
// Introducing JRadioButton and JCheckBox and JTextArea
import  javax.swing.*;
import java.awt.event.*;
class GUISample2 implements ActionListener, ItemListener
{
     JButton  btnClear;
     JRadioButton rdMale, rdFemale;
     JCheckBox chkDegree, chkDiploma;
     JTextArea txtAreaData;
     ButtonGroup bg;

     GUISample2()
     {
        JFrame  frm = new JFrame("Radio Button, CheckBox and TextArea");
        frm.setSize(500,500);
        frm.setLayout(null);
        frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        btnClear = new JButton("Clear");
        btnClear.setBounds(160, 170, 100, 50);

        rdMale = new JRadioButton("Male");
        rdMale.setBounds(50,50,100,50);

        rdFemale = new JRadioButton("Female");
        rdFemale.setBounds(160,50,100,50);

        bg=new ButtonGroup();
        bg.add(rdMale);
        bg.add(rdFemale);

        chkDegree = new JCheckBox("Degree");
        chkDegree.setBounds(50,110, 100,50);

        chkDiploma = new JCheckBox("Diploma");
        chkDiploma.setBounds(160,110, 100,50);

        txtAreaData = new JTextArea("");
        txtAreaData.setBounds(270, 50, 150, 120);

```

```
            frm.add(rdMale);
            frm.add(rdFemale);
            frm.add(chkDegree);
            frm.add(chkDiploma);
            frm.add(btnClear);
            frm.add(txtAreaData);
            btnClear.addActionListener(this);
            rdMale.addItemListener(this);
            rdFemale.addItemListener(this);
            chkDegree.addItemListener(this);
            chkDiploma.addItemListener(this);
            frm.setVisible(true);
        }
```

```
    public void actionPerformed(ActionEvent ae)
    {
       if (ae.getSource() == btnClear)
       {
               chkDegree.setSelected(false);
               chkDiploma.setSelected(false);
               rdMale.setSelected(false);
               rdFemale.setSelected(false);
               txtAreaData.setText("");
       }
    }
    public void itemStateChanged(ItemEvent ie)
    {
       String data ="";
       if(rdMale.isSelected())
               data = data + "Gender = Male\n";
       else if (rdFemale.isSelected())
               data = data + "Gender = Female\n";

       if (chkDegree.isSelected())
               data = data + "Diploma\n";
       if (chkDiploma.isSelected())
               data = data + "Degree\n";
       txtAreaData.setText(data);
    }

    public static void main(String args[])
    {
       new GUISample2();
    }
}
```

Methods of JRadioButton class

1	**JRadioButton(String str)**	**Constructor ot create RadioButton object**
2	**getText()**	**To read the text message on RadioButton**
3	**setText(string)**	**to set the string message on RadioButton object**
4	**setBounds(x, y, width, height)**	
5	**isSelected()**	**Returns true if the radiobutton is selected**
6	**addItemListener()**	**To register for listening the ItemChange Event generated when the user interacts with JRadiButton**
7	**setSelected(boolean)**	**To select and deselect the button through program**

Methods of JCheckBox class

1	**JCheckBox(String str)**	**Constructor to create CheckBox object**
2	**getText()**	**To read the string message on the object**
3	**setText(string)**	**to set the string message on the object**
4	**setBounds(x, y, width, height)**	
5	**isSelected()**	**Returns true if the checkBox is ticked**
6	**addItemListener()**	**To register for listening the ItemChange Event generated when the user interacts with JCheckBox**
7	**setSelected(boolean)**	**To select and deselect the CheckBox through program**

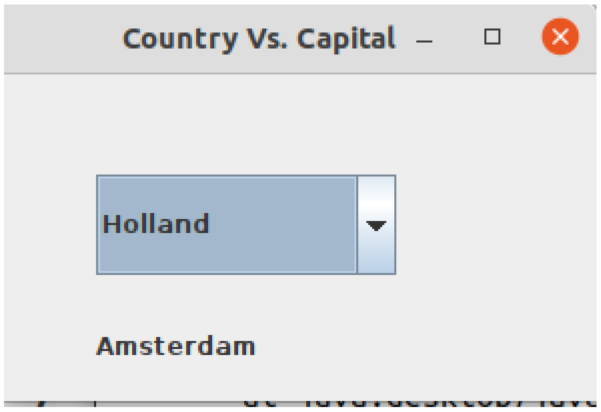
Methods of JTextArea class

1. JTextArea(String str)	**– Constructor ot create JTextArea object**
2. getText()	**- To read the string message on the object**
3. setText(string)	**– to set the string message on the object**
4. setBounds(x, y, width, height)	

<u>How to build a GUI using JComboBox?</u>

- **Import the packages –**
    - **javax.swing - it contains the classes for the component controls and the container JFrame**
    - **java.awt.event - it contains the event classes and event listeners for event handling**
- **Define a GUI class that implements ItemListener interface**
- **Inside the class, declare a JComboBox object and JLabel object**
- **Define Constructor of the class that does the following:**
    - **create a JFrame, set its size, set the layout to null**
    - **create and initialize JComboBox & JLabel objects (set the boundaries x, y, width and height within the frame)**
    - **Add ComboBox and label into frame in a proper layout**
    - **Call addItemListener() method of ComboBox object to register for receiving the eventobject generated when the user selecting an item from Combo Box**
    - **make the Visible property of Jframe to true (show the frame)**
- **Define the public void itemStateChanged(ItemEvent e) of the ItemListener interface to handle the events generated from the Button object**
- **Create mian() function and instantiate an object of GUI class (i. e. create an object of the GUI Class)**

Sample Program: Demonstrating the above steps with an exercise that displays the capital of selected country



```java
// Introducing JComboBox
import  javax.swing.*;
import java.awt.event.*;
class GUISample3 implements ItemListener
{
      JComboBox cbCountry;
      JLabel lblCapital;
      String country[]={"Afghanistan", "Brazil", "Canada", "Denmark",
                        "Finland", "Germany", "Holland"};
      String capital[]={"Kabul", "Brasilia", "Ottawa", "Copenhagen",
                        "Helsinki", "Berlin", "Amsterdam"};
      GUISample3()
      {
         JFrame  frm = new JFrame("Country Vs. Capital");
         frm.setSize(200,200);
         frm.setLayout(null);
         frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

         cbCountry = new JComboBox(country);
         cbCountry.setBounds(50, 50, 100, 50);

         lblCapital = new JLabel("Capital");
         lblCapital.setBounds(50, 110,100,50);

         frm.add(cbCountry);
         frm.add(lblCapital);
         cbCountry.addItemListener(this);
         frm.setVisible(true);
      }
      public void itemStateChanged(ItemEvent ie)
      {
         int index=cbCountry.getSelectedIndex();
         lblCapital.setText(capital[index]);
      }
      public static void main(String args[])
      {
         new GUISample3();
      }
}
```

<u>**Methods of JComboBox class**</u>

1. **JComboBox(String Array)** - Constructor to create JComboBox with a list of objects from which we can select one
2. **setBounds(x, y, width, height)** - To set the position and size of the button in frame
3. **setSelectedIndex(int)** - to select an item from the combo box list through program
4. **getSelectedIndex()** - to get the index of selected item
5. **getSelectedItem()** - to get the item selected from the combo box
6. **addItemListener(listener_object)** – to register for the action event

<u>EventClass, EventListener used in the program</u>

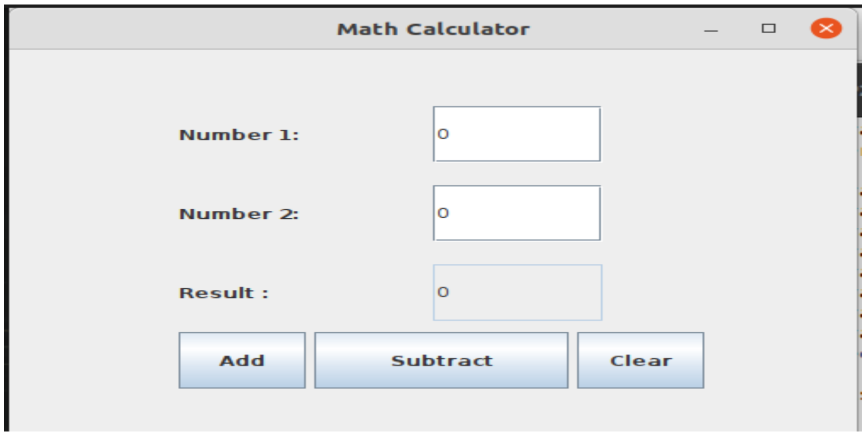
- The event class used in this program – **ItemEvent**
- The event listener used is - **ItemListener**
- The method to implement - **itemStateChanged(ItemEvent)**

<u>How to build a GUI using JTextField?</u>

**** Here we discuss the simple two operand calculator which includes three TextField objects (two numbers and the result)***

- **Import the packages –**
    - **javax.swing - it contains the classes for the component controls and the container JFrame**
    - **java.awt.event - it contains the event classes and event listeners for event handling**
- **Define a GUI class that implements ActionListener interface**
- **Inside the class, declare a three JTextField objects and necessary JLabel objects & Three Buttons**
- **Define Constructor of the class that does the following:**
    - **create a JFrame, set its size, set the layout to null**
    - **create and initialise JTextField , JLabel and JButton objects (set the boundaries x, y, width and height within the frame)**
    - **Add all components into frame in a proper layout**
    - **Call addActionListener() method of Button objects to register for button events**
    - **make the Visible property of Jframe to true (show the frame)**
- **Define the public void actionPerofrmed(ActionEvent e) of the ActionListener interface to handle the events generated from the List object**
- **Create mian() function and instantiate an object of GUI class (i. e. create an object of the GUI Class)**

Demonstrating the above steps with an exercise that implements simple calculator



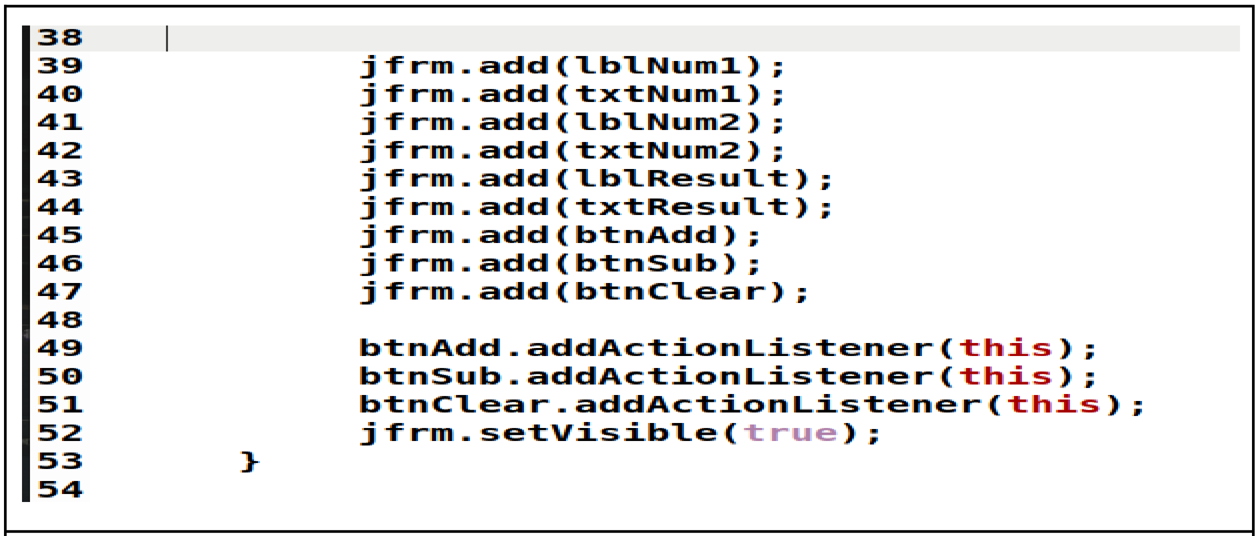
```java
// Simple Calculator using Java Swing Controls
import javax.swing.*;
import java.awt.event.*;

class Calculator implements ActionListener
{
        JLabel lblNum1,lblNum2, lblResult;
        JTextField txtNum1, txtNum2, txtResult;
        JButton btnAdd, btnSub, btnClear;

       Calculator()
       {
           JFrame jfrm=new JFrame("Math Calculator");
           lblNum1 = new JLabel("Number 1:");
           lblNum2 = new JLabel("Number 2:");
           lblResult = new JLabel("Result :");
           txtNum1 = new JTextField("0");
           txtNum2 = new JTextField("0");
           txtResult = new JTextField("0");
           txtResult.setEditable(false);
           btnAdd = new JButton("Add");
           btnSub = new JButton("Subtract");
           btnClear = new JButton("Clear");

           jfrm.setSize(500,500);
           jfrm.setLayout(null);
           jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

           lblNum1.setBounds(100,50,100,50); // (x, y, width,height)
           lblNum2.setBounds(100,120,100,50);
           lblResult.setBounds(100,190,100,50);
           txtNum1.setBounds(250,50,100,50); // (x, y, width,height)
           txtNum2.setBounds(250,120,100,50);
           txtResult.setBounds(250,190,100,50);
           btnAdd.setBounds(100,250,75,50);
           btnSub.setBounds(180,250,150,50);
           btnClear.setBounds(335,250,75,50);
```

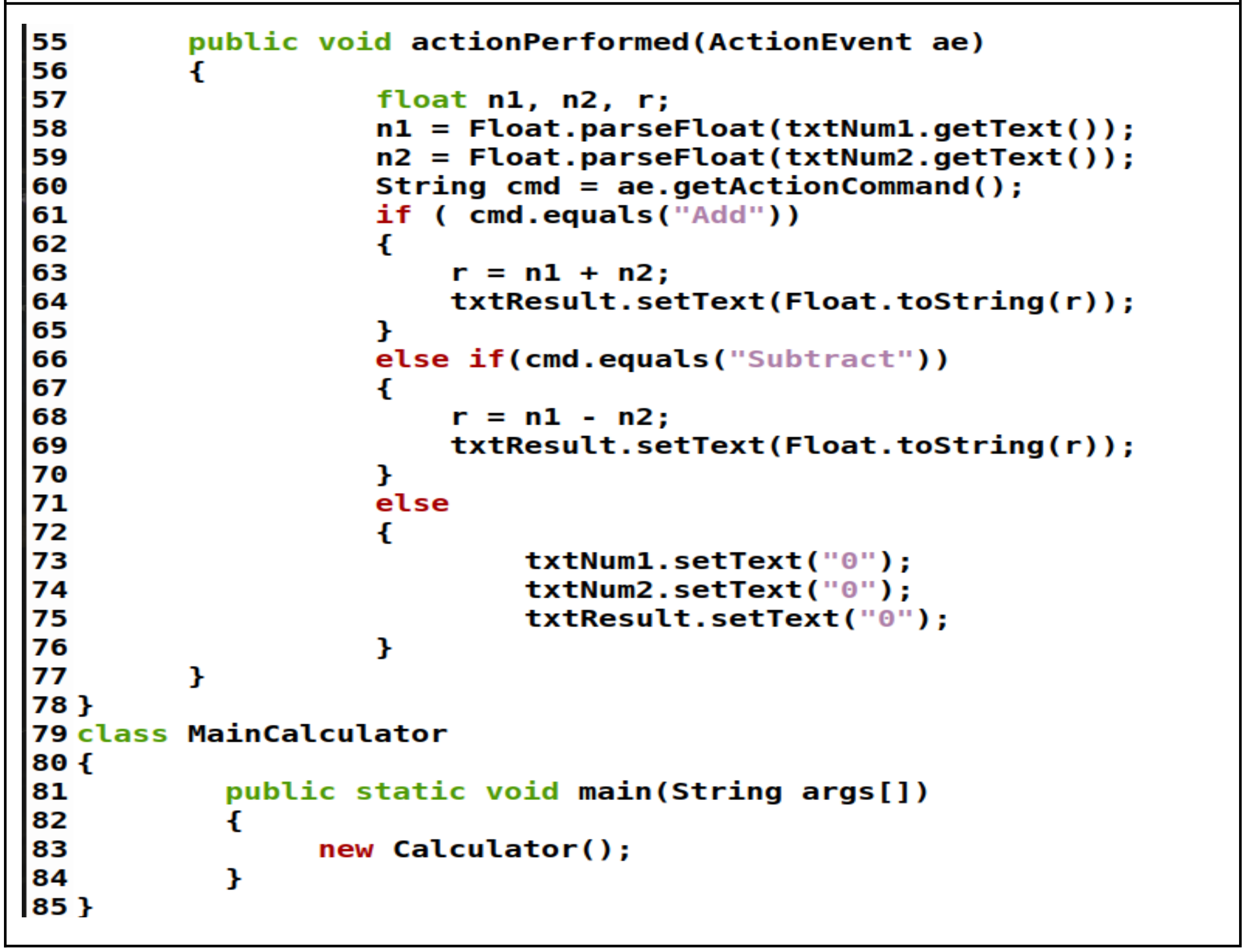


```

                jfrm.add(lblNum1);
                jfrm.add(txtNum1);
                jfrm.add(lblNum2);
                jfrm.add(txtNum2);
                jfrm.add(lblResult);
                jfrm.add(txtResult);
                jfrm.add(btnAdd);
                jfrm.add(btnSub);
                jfrm.add(btnClear);

                btnAdd.addActionListener(this);
                btnSub.addActionListener(this);
                btnClear.addActionListener(this);
                jfrm.setVisible(true);
        }

```



```
        public void actionPerformed(ActionEvent ae)
        {
                float n1, n2, r;
                n1 = Float.parseFloat(txtNum1.getText());
                n2 = Float.parseFloat(txtNum2.getText());
                String cmd = ae.getActionCommand();
                if ( cmd.equals("Add"))
                {
                    r = n1 + n2;
                    txtResult.setText(Float.toString(r));
                }
                else if(cmd.equals("Subtract"))
                {
                    r = n1 - n2;
                    txtResult.setText(Float.toString(r));
                }
                else
                {
                        txtNum1.setText("0");
                        txtNum2.setText("0");
                        txtResult.setText("0");
                }
        }
}
class MainCalculator
{
        public static void main(String args[])
        {
                new Calculator();
        }
}
```

Methods of JTextField class

1. **JTextField(String)** - **Constructor to create Text field (Editable field) with a list of objects from which we can select one**
2. **getText()** - **To get the string from text field**
3. **setText(string)** – **To set a string in the text field**
4. **setBounds(x, y, width, height)** - **To set the position and size of the button in frame**
5. **cut() , paste(), copy()** - **to cut, copy and paste the selected text**
6. **getSelectedText()** - **to get the index of selected item**
7. **addActionListener(listener_object)** – **to register for the action event**
8. **setEditable(boolean)** – **if editable is false, user cannot edit the text**
9. **setEnabled(boolean)** – **if enabled is false, user cannot use the control**

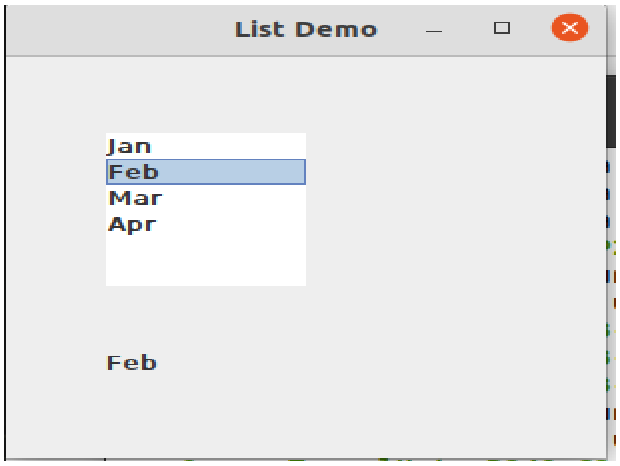
EventClass, EventListener used in the program

- **The event class used in this program –** ActionEvent
- **The event listener used is -** ActionListener
- **The method implemented –** **actionPerformed(ActionEvent)**

How to build a GUI using JList?

- **Import the packages –**
  - **javax.swing - it contains the classes for the component controls and the container JFrame**
  - **java.awt.event - it contains the event classes and event listeners for event handling**
- **Define a GUI class that implements ListSelectionListener interface**
- **Inside the class, declare a JList object and JLabel object**
- **Define Constructor of the class that does the following:**
  - **create a JFrame, set its size, set the layout to null**
  - **create and initialise JList & JLabel objects (set the boundaries x, y, width and height within the frame)**
  - **Add ListBox and label into frame in a proper layout**
  - **Call addListSelectionListener() method of ComboBox object to register for receiving the eventobject generated when the user selecting an item from List Box**
  - **make the Visible property of Jframe to true (show the frame)**
- **Define the public void valueChanged(ListSelectionEvent e) of the ListSelectionListener interface to handle the events generated from the List object**
- **Create mian() function and instantiate an object of GUI class (i. e. create an object of the GUI Class)**

Demonstrating the above steps with an exercise that displays the the selected text on the label control



```java
// Demo of JList Swing control
import javax.swing.*;
import javax.swing.event.*;
import java.awt.event.*;
class ListDemo implements ListSelectionListener
{
        JList lstMonths;
        JLabel lblMessage;
        String months[]={"Jan", "Feb","Mar", "Apr"};
        ListDemo()
        {

            JFrame frm=new JFrame("List Demo");
            frm.setLayout(null);
            frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frm.setSize(300,300);

            lstMonths= new JList<String> (months);
            lstMonths.setBounds(50,50,100,100);
            lblMessage = new JLabel("........");
            lblMessage.setBounds(50,175, 100,50);

            frm.add(lstMonths);
            frm.add(lblMessage);
            lstMonths.addListSelectionListener(this);

            frm.setVisible(true);
        }
        public void valueChanged(ListSelectionEvent le)
        {
                int index=lstMonths.getSelectedIndex();
                lblMessage.setText(months[index]);
        }
        public static void main(String args[])
        {
            new ListDemo();
        }
}
```

Methods of JList class

1. **JList(String Array) - Constructor to create List box with a list of objects from which we can select one**
2. **setBounds(x, y, width, height) - To set the position and size of the button in frame**
3. **setSelectedIndex(int) - to select an item from the combo box list through program**
4. **getSelectedIndex() - to get the index of selected item**
5. **addListSelectionListener(listener_object) – to register for the action event**

EventClass, EventListener used in the program

- **The event class used in this program – ListSelectionEvent**
- **The event listener used is - ListSelectionListener**
- **The method implemented - valueChanged(ListSelectionEvent)**

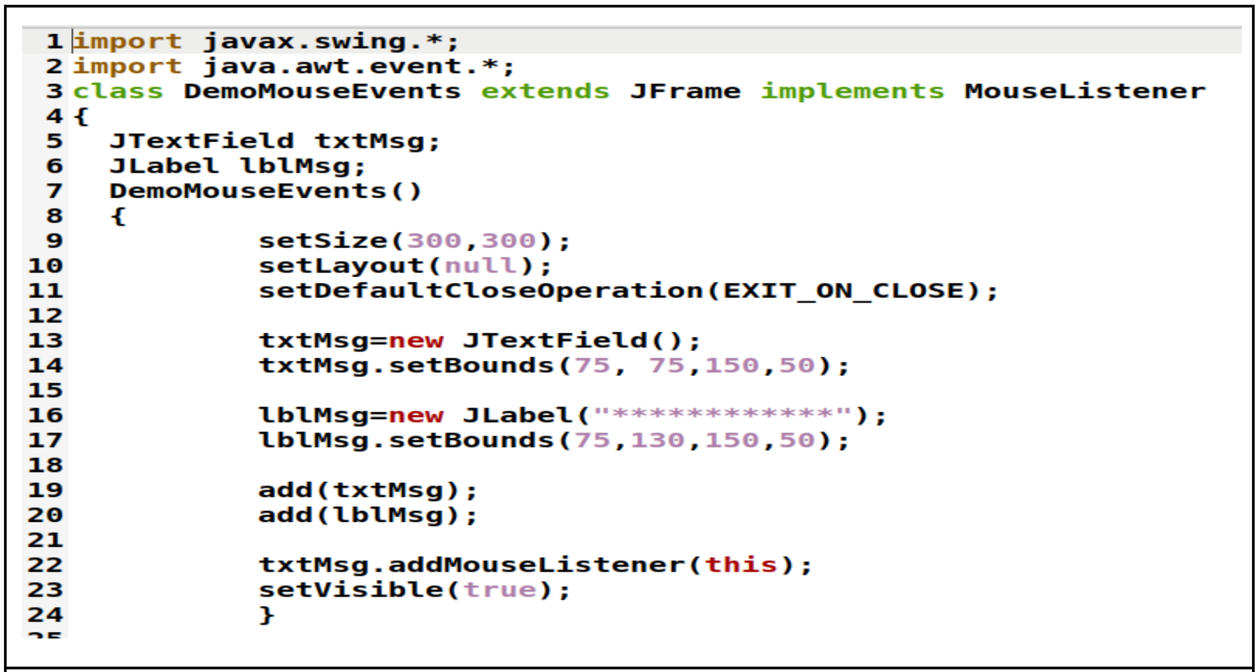
Handling Mouse Events

To handle mouse events that occurs upon a swing control , We have to do the following tasks in our GUI class

1. **Implement MouseListener interface in your GUI class (In this program, GUI class is extended from JFrame and implemented MouseListener)**
2. **As usual, design your user interface by creating and adding the components in a container frame.**
3. **Register for MouseEvents using the method addMouseListener()**
4. **Define the five abstract methods of MouseListener interface. Those methods are:**
    - **a) mouseClicked(MouseEvent)**
    - **b) mousePressed(MouseEvent)**
    - **c) mouseEntered(MouseEvent)**
    - **d) mouseExited(MouseEvent)**
    - **e) mouseReleased(MouseEvent)**

5) Finally create an instance (object) of your GUI class in the main() method to display the user interface so that you can interact with it.

Sample program is given below:



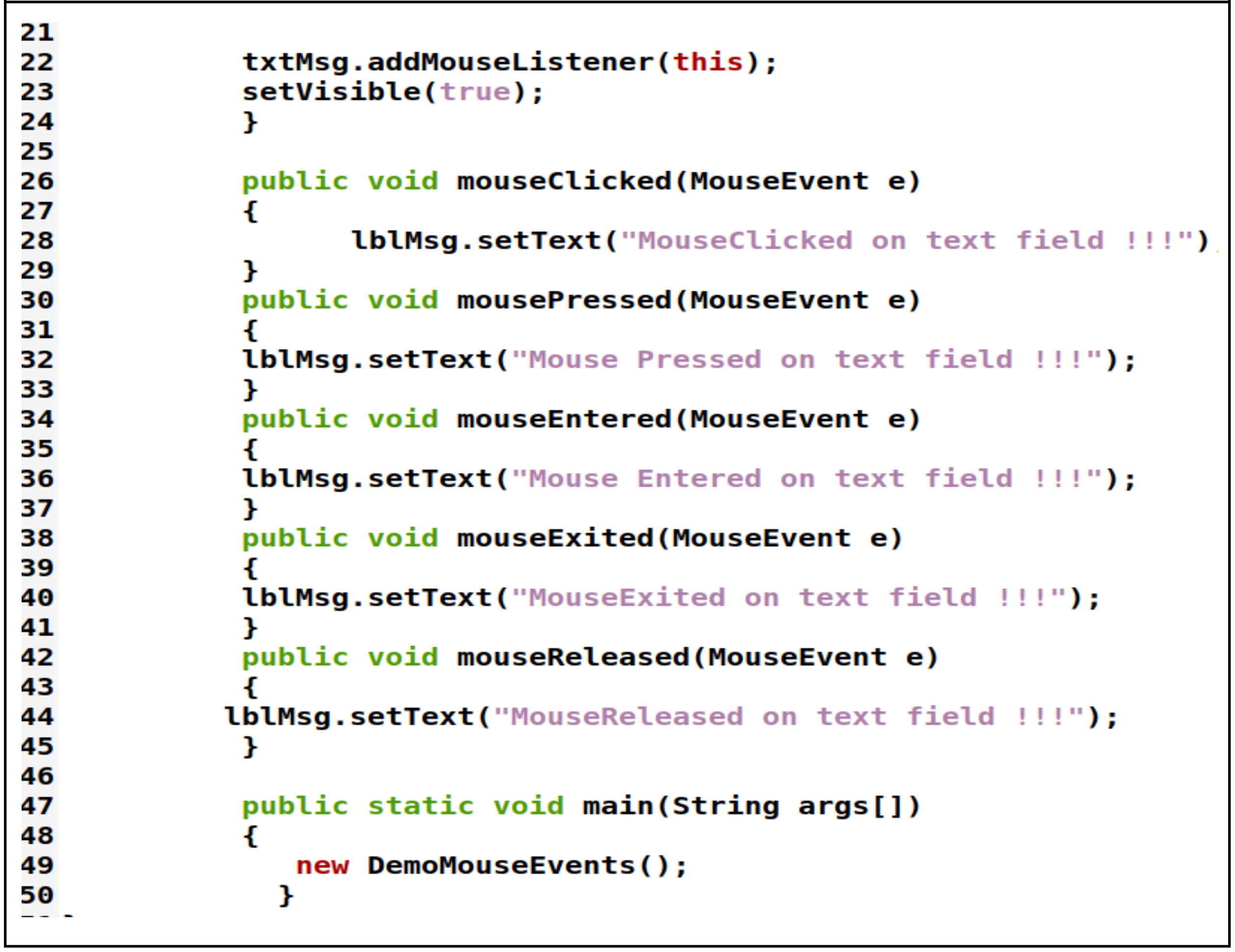
```
import javax.swing.*;
import java.awt.event.*;
class DemoMouseEvents extends JFrame implements MouseListener
{
  JTextField txtMsg;
  JLabel lblMsg;
  DemoMouseEvents()
  {
          setSize(300,300);
          setLayout(null);
          setDefaultCloseOperation(EXIT_ON_CLOSE);

          txtMsg=new JTextField();
          txtMsg.setBounds(75, 75,150,50);

          lblMsg=new JLabel("************");
          lblMsg.setBounds(75,130,150,50);

          add(txtMsg);
          add(lblMsg);

          txtMsg.addMouseListener(this);
          setVisible(true);
          }
```



```

          txtMsg.addMouseListener(this);
          setVisible(true);
          }

          public void mouseClicked(MouseEvent e)
          {
                lblMsg.setText("MouseClicked on text field !!!")
          }
          public void mousePressed(MouseEvent e)
          {
          lblMsg.setText("Mouse Pressed on text field !!!");
          }
          public void mouseEntered(MouseEvent e)
          {
          lblMsg.setText("Mouse Entered on text field !!!");
          }
          public void mouseExited(MouseEvent e)
          {
          lblMsg.setText("MouseExited on text field !!!");
          }
          public void mouseReleased(MouseEvent e)
          {
         lblMsg.setText("MouseReleased on text field !!!");
          }

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

Handling Key Events

1. **To handle Key board events that occurs upon a swing control , ee have to do the following tasks in our GUI class**

2. **Implement KeyListener interface in your GUI class (In this program, GUI class is extended from JFrame and implemented KeyListener)**
3. **As usual, design your user interface by creating and adding the components in a container frame.**
4. **Register for KeyEvents using the method addKeyListener()**
5. **Define the three abstract methods of KeyListener interface. Those methods are:**
   - **a) keyPressed(KeyEvent)**
   - **b) keyReleased(KeyEvent)**
   - **c) keyTyped(KeyEvent)**
6. **Finally create an instance (object) of your GUI class in main() method to display the user interface so that you can interact with it.**

Sample program is given below (Displays number of characters you typed in a JTextField control):

```java
import javax.swing.*;
import java.awt.event.*;
class DemoKeyEvents extends JFrame implements KeyListener
{
  JTextField txtMsg;
  JLabel lblMsg;
  DemoKeyEvents()
  {
            setSize(300,300);
            setLayout(null);
            setDefaultCloseOperation(EXIT_ON_CLOSE);

            txtMsg=new JTextField();
            txtMsg.setBounds(75, 75,150,50);

            lblMsg=new JLabel("************");
            lblMsg.setBounds(75,130,150,50);

            add(txtMsg);
            add(lblMsg);

            txtMsg.addKeyListener(this);
            setVisible(true);
            }

          public void keyPressed(KeyEvent e)
          {
                // Empty definition
          }
          public void keyReleased(KeyEvent e)
          {
                String txt = txtMsg.getText();
                lblMsg.setText("Character Count:"+ txt.length());
          }
          public void keyTyped(KeyEvent e)
          {
                //Empty definition
          }

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