

14. SWINGS AND EVENTHANDLING

1. Java Swing:

Java Swing is a part of Java Foundation Classes (JFC) . SWINGS are used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. 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.

2. Difference between AWT and Swing:

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

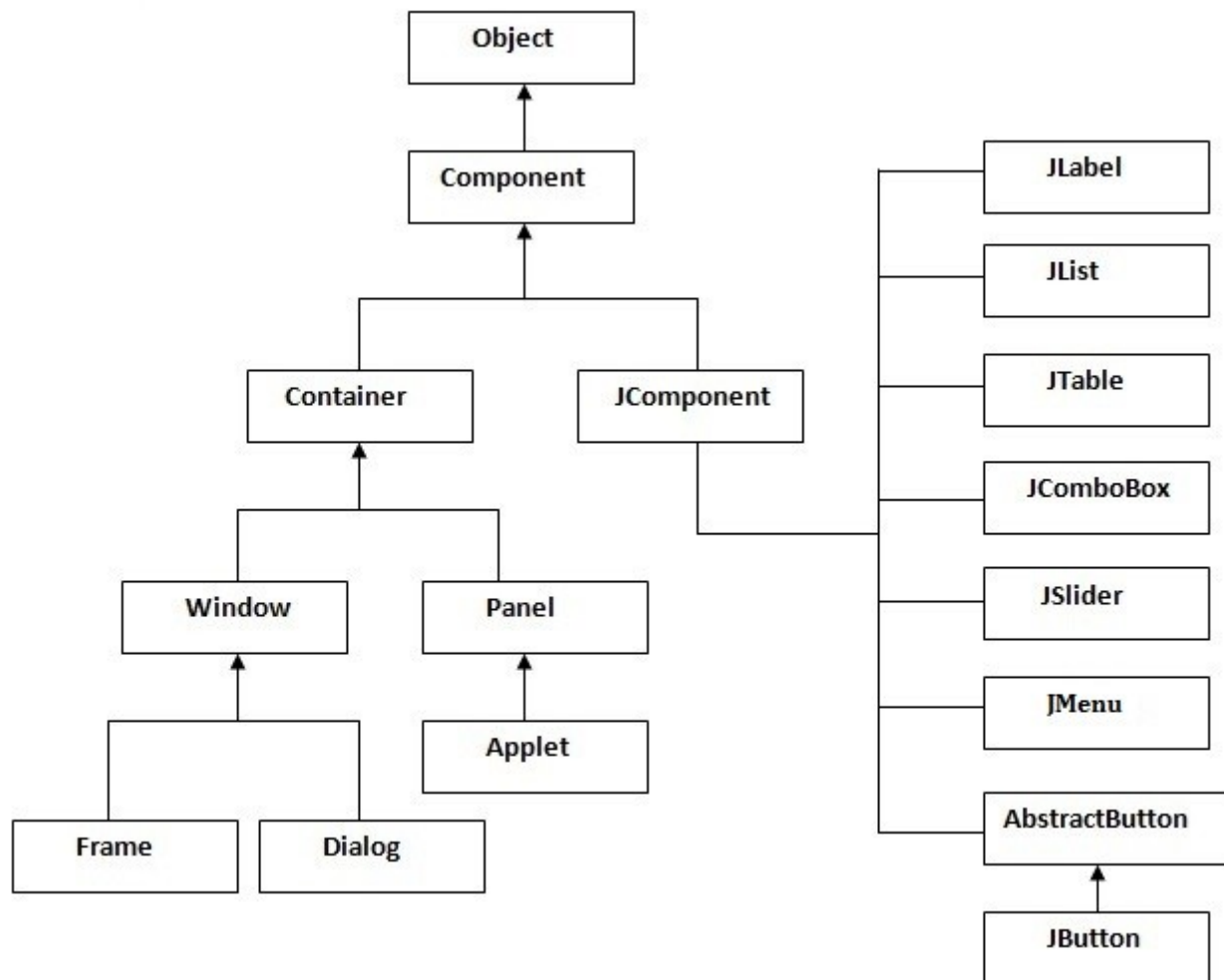
Java AWT	Java Swing
AWT components are platform-dependent.	Java swing components are platform-independent.
AWT components are heavyweight.	Swing components are lightweight.
AWT doesn't support pluggable look and feel.	Swing supports pluggable look and feel.
AWT provides less components than Swing.	Swing provides more powerful components such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
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.

3. Swing features:

- **Light Weight** - Swing component are independent of native Operating System's API as Swing API controls are rendered mostly using pure JAVA code instead of underlying operating system calls.
- **Rich controls** - Swing provides a rich set of advanced controls like Tree, TabbedPane, slider, colorpicker, table controls
- **Highly Customizable** - Swing controls can be customized in very easy way as visual appearance is independent of internal representation.
- **Pluggable look-and-feel**- SWING based GUI Application look and feel can be changed at run time based on available values.

4. Hierarchy of Java Swing classes:

14. SWINGS AND EVENTHANDLING



5. Methods of Component class:

The methods of Component class are widely used in java swing that are given below.

Method	Description
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

Every SWING controls inherits properties from following Component class hierarchy.

Sr. No.	Class & Description
1	Component

14. SWINGS AND EVENTHANDLING

	A Container is the abstract base class for the non menu user-interface controls of SWING. Component represents an object with graphical representation
2	Container A Container is a component that can contain other SWING components.
3	JComponent A JComponent is a base class for all swing UI components. In order to use a swing component that inherits from JComponent, component must be in a containment hierarchy whose root is a top-level Swing container.

6. SWING UI Elements:

Following is the list of commonly used controls while designed GUI using SWING.

Sr. No.	Control & Description
1	JLabel A JLabel object is a component for placing text in a container.
2	JButton This class creates a labeled button.
3	JColorChooser A JColorChooser provides a pane of controls designed to allow a user to manipulate and select a color.
4	JCheck Box A JCheckBox is a graphical component that can be in either an on (true) or off (false) state.
5	JRadioButton The JRadioButton class is a graphical component that can be in either an on (true) or off (false) state. in a group.
6	JList A JList component presents the user with a scrolling list of text items.
7	JComboBox A JComboBox component presents the user with a to show up menu of choices.
8	JTextField A JTextField object is a text component that allows for the editing of a single line of text.
9	JPasswordField A JPasswordField object is a text component specialized for password entry.
10	JTextArea A JTextArea object is a text component that allows for the editing of a multiple lines of text.

7. Components of Swing:

14. SWINGS AND EVENTHANDLING

- **JFrame** is Swing's version of Frame and is descended directly from that class. The components added to the frame are referred to as its contents; these are managed by the contentPane. To add a component to a JFrame, we must use its contentPane instead.
- **JInternalFrame** is confined to a visible area of a container it is placed in. It can be iconified, maximized and layered.
- **JWindow** is Swing's version of Window and is descended directly from that class. Like Window, it uses BorderLayout by default.
- **JLabel** descended from JComponent, is used to create text labels.
- **JButton** is the abstract class AbstractButton extends class JComponent and provides a foundation for a family of button classes. JButton is a component the user clicks to trigger a specific action.
- **JTextField** allows editing of a single line of text. New features include the ability to justify the text left, right, or center, and to set the text's font.
- **JPasswordField** (a direct subclass of JTextField) you can suppress the display of input. Each character entered can be replaced by an echo character. This allows confidential input for passwords, for example. By default, the echo character is the asterisk, *.
- **JTextArea** allows editing of multiple lines of text. JTextArea can be used in conjunction with class JScrollPane to achieve scrolling. The underlying JScrollPane can be forced to always or never have either the vertical or horizontal scrollbar;
- **JRadioButton** is similar to JCheckbox, except for the default icon for each class. A set of radio buttons can be associated as a group in which only one button at a time can be selected.
- **JCheckBox** is not a member of a checkbox group. A checkbox can be selected and deselected, and it also displays its current state. Multiple checkboxes can be selected and deselected
- **JComboBox** is like a drop down box. You can click a drop-down arrow and select an option from a list. For example, when the component has focus, pressing a key that corresponds to the first character in some entry's name selects that entry. A vertical scrollbar is used for longer lists.

8. Layouts in Swing:

- **FlowLayout** when used arranges swing components from left to right until there's no more space available. Then it begins a new row below it and moves from left to right again. Each component in a FlowLayout gets as much space as it needs and no more.
- **BorderLayout** places swing components in the North, South, East, West and center of a container. You can add horizontal and vertical gaps between the areas.
- **GridLayout** is a layout manager that lays out a container's components in a rectangular grid. The container is divided into equal-sized rectangles, and one component is placed in each rectangle.
- **GridBagLayout** is a layout manager that lays out a container's components in a grid of

14. SWINGS AND EVENTHANDLING

cells with each component occupying one or more cells, called its display area. The display area aligns components vertically and horizontally, without requiring that the components be of the same size.

- **JMenuBar** can contain several JMenu's. Each of the JMenu's can contain a series of JMenuItem 's that you can select. Swing provides support for pull-down and popup menus.

9. Creating Frame using Swing :

There are two ways to create a frame:

- By creating the object of Frame class (association)
- By extending Frame class (inheritance)

We can write the code of swing inside the main(), constructor or any other method.

Example :

using object: JFrame jf=new JFrame()

using inheritance: class must extend JFrame class. Then no need to create JFrame object explicitly

10. Event Handling :

10.1 Event in Java:

Change in the state of an object is known as event i.e. event describes the change in state of source. 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, selecting an item from list, scrolling the page are the activities that causes an event to happen.

10.2. Types of Event:

The events can be broadly classified into two categories:

- **Foreground Events** - Those events which require the direct interaction of user. They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page etc.
- **Background Events** - Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

10.3. Event Handling in Java:

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. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events. Let's have a brief introduction to this model.

The Delegation Event Model has the following key participants namely:

- **Source** - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provide as with classes for source object.

14. SWINGS AND EVENTHANDLING

- **Listener** - It is also known as event handler. Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener process the event and then returns.

10.2 Benefit of Delegation Event Model :

The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event. The user interface element is able to delegate the processing of an event to the separate piece of code. In this model ,Listener needs to be registered with the source object so that the listener can receive the event notification. This is an efficient way of handling the event because the event notifications are sent only to those listener that want to receive them.

Steps involved in event handling

- The User clicks the button and the event is generated.
- Now the object of concerned event class is created automatically and information about the source and the event get populated with in same object.
- Event object is forwarded to the method of registered listener class.
- the method is now get executed and returns.

Note: In java source will be any GUI component. When any action is done on GUI, in java type of event object will get created and the type of listener interface method will get called. In that method event will get processed.

11. Event Handling Examples :

1. Example of Swing by Association inside constructor

We can also write all the codes of creating JFrame, JButton and method call inside the java constructor.

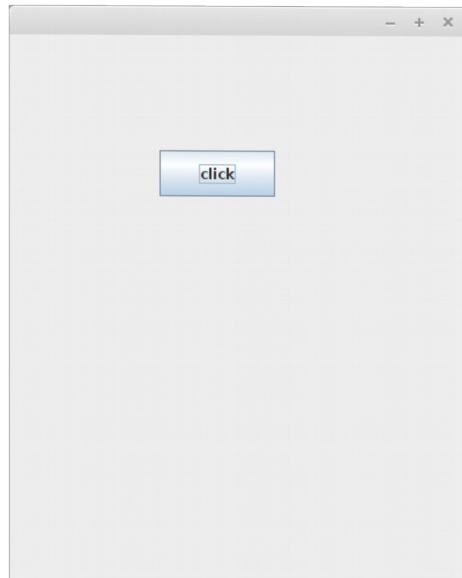
```
import javax.swing.JButton;
import javax.swing.JFrame;

public class Simple {
    JFrame f;
    Simple() {
        f=new JFrame();//creating instance of JFrame
        JButton b=new JButton("click");//creating instance of JButton
        b.setBounds(130,100,100,40);
        f.add(b);        //adding button in JFrame
        f.setSize(400,500);    //400 width and 500 height
        f.setLayout(null);    //using no layout managers
        f.setVisible(true);    //making the frame visible
    }
    public static void main(String[] args) {
```

14. SWINGS AND EVENTHANDLING

```
        new Simple();  
    }  
}
```

Output:



2. Working on Buttons

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

```
import javax.swing.JButton;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
import javax.swing.JPanel;
```

```
public class SwingControlDemo implements ActionListener{  
    private JFrame mainFrame;  
    private JLabel headerLabel;  
    private JLabel statusLabel;  
    private JPanel controlPanel;  
  
    public SwingControlDemo(){  
        prepareGUI();  
    }  
  
    public void prepareGUI(){  
        mainFrame = new JFrame("Java SWING Examples");  
        mainFrame.setSize(400,400);
```

14. SWINGS AND EVENTHANDLING

```
mainFrame.setLayout(new GridLayout(3, 1));

headerLabel = new JLabel("",JLabel.CENTER );
statusLabel = new JLabel("",JLabel.CENTER);

statusLabel.setSize(350,100);

controlPanel = new JPanel();
controlPanel.setLayout(new FlowLayout());

mainFrame.add(headerLabel);
mainFrame.add(controlPanel);
mainFrame.add(statusLabel);
mainFrame.setVisible(true);
}
public void showEventDemo(){
    headerLabel.setText("Control in action: Button");

    JButton okButton = new JButton("OK");
    JButton submitButton = new JButton("Submit");
    JButton cancelButton = new JButton("Cancel");

    okButton.setActionCommand("OK");
    submitButton.setActionCommand("Submit");
    cancelButton.setActionCommand("Cancel");

    okButton.addActionListener(this);
    submitButton.addActionListener(this);
    cancelButton.addActionListener(this);

    controlPanel.add(okButton);
    controlPanel.add(submitButton);
    controlPanel.add(cancelButton);
    mainFrame.setVisible(true);

}
public void actionPerformed(ActionEvent e) {
    String command = e.getActionCommand();

    if( command.equals( "OK" ) ) {
        statusLabel.setText("Ok Button clicked.");
    }

    else if( command.equals( "Submit" ) ) {
        statusLabel.setText("Submit Button clicked.");
    }

    else {
        statusLabel.setText("Cancel Button clicked.");
    }
}
```


14. SWINGS AND EVENTHANDLING

```
    }  
}  
public static void main(String[] args) {  
    SwingControlDemo swingControlDemo = new SwingControlDemo();  
    swingControlDemo.prepareGUI();  
    swingControlDemo.showEventDemo();  
}  
}
```

Output:



3. Program to design login frame and handling the event when clicking Login button.

```
public class GUILogin extends JFrame implements ActionListener{  
    JLabel lname,lpswd, lmsg;  
    JButton blogin,breset;  
    JTextField tuname,tmsg;  
    JPasswordField pswd;  
    GUILogin() {  
        setSize(500,400);  
        setTitle("Login");  
        lname=new JLabel("User Name");  
        lpswd=new JLabel("Password");  
        lmsg=new JLabel("Message");  
        blogin=new JButton("LOGIN");  
        breset=new JButton("RESET");  
    }  
}
```

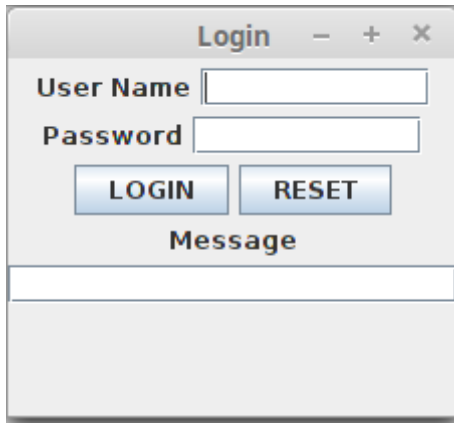
14. SWINGS AND EVENTHANDLING

```
tuname=new JTextField(10);
pswd=new JPasswordField(10);
tmsg=new JTextField(20);
add(luname);
add(tuname);
add(lpswd);
add(pswd);
add(blogin);
add(breset);
add(lmsg);
add(tmsg);
setLayout(new FlowLayout());
setVisible(true);
blogin.addActionListener(this);
breset.addActionListener(this);
pswd.setEchoChar('*');
}
@Override
public void actionPerformed(ActionEvent ae) {
    String blabel=ae.getActionCommand();
    if(blabel.equals("LOGIN")) {
        String uname=tuname.getText();
        //String pswd=pswd.getText();
        if(uname.equals("ctuser01") && pswd.equals("c0d3r123")) {
            tmsg.setText("Login Sucessfully");
        }
        else {
            tmsg.setText("Login Unsuccesfully");
        }
    }
    else if(blabel.equals("RESET")) {
        tuname.setText("");
        pswd.setText("");
        tmsg.setText("");
    }
}

public static void main(String[] args) {
    JFrame jf=new GUILogin();
}
}
```

Output:

14. SWINGS AND EVENTHANDLING



4. Program on BorderLayout

```
public class BorderLayoutLayout extends JFrame {  
  
    JButton btnSouth,btnNorth,btnEast,btnWest,btnCenter;  
  
    public BorderLayoutLayout() {  
        btnSouth=new JButton("South");  
        btnNorth=new JButton("North");  
        btnEast=new JButton("East");  
        btnWest=new JButton("West");  
        btnCenter=new JButton("Center");  
  
        add(btnSouth,BorderLayout.SOUTH);  
        add(btnNorth,BorderLayout.NORTH);  
        add(btnEast,BorderLayout.EAST);  
        add(btnWest,BorderLayout.WEST);  
        add(btnCenter,BorderLayout.CENTER);  
        setSize(200,200);  
        setTitle("first frame");  
        setVisible(true);  
    }  
    public static void main(String[] args) {  
        BorderLayoutLayout frame=new BorderLayoutLayout();  
    }  
}
```

5. Program on FlowLayout

```
public class FrameFlowLayout extends JFrame  
{
```

14. SWINGS AND EVENTHANDLING

```
JButton btnSouth,btnNorth,btnEast,btnWest,btnCenter;
```

```
public FrameFlowLayout() {  
    btnSouth=new JButton("South");  
    btnNorth=new JButton("North");  
    btnEast=new JButton("East");  
    btnWest=new JButton("West");  
    btnCenter=new JButton("Center");  
  
    setLayout(new FlowLayout());  
  
    add(btnSouth);  
    add(btnNorth);  
    add(btnEast);  
    add(btnWest);  
    add(btnCenter);  
    setSize(200,200);  
    setTitle("first frame");  
    setVisible(true);  
}  
  
public static void main(String[] args) {  
    FrameFlowLayout frame=new FrameFlowLayout();  
}  
}
```

6. Program on GridLayout

```
public class FrameGridLayout extends JFrame {  
    JButton btnSouth,btnNorth,btnEast,btnWest,btnCenter;  
  
    public FrameGridLayout() {  
        btnSouth=new JButton("South");  
        btnNorth=new JButton("North");  
        btnEast=new JButton("East");  
        btnWest=new JButton("West");  
  
        btnCenter=new JButton("Center");  
        setLayout(new GridLayout(3,2));  
  
        add(btnSouth);  
        add(btnNorth);  
        add(btnEast);  
    }  
}
```

14. SWINGS AND EVENTHANDLING

```
        add(btnWest);
        add(btnCenter);
        setSize(200,200);
        setTitle("first frame");
        setVisible(true);
    }
    public static void main(String[] args) {
        GridLayout frame=new GridLayout();
    }
}
```

7. Program on designing Registration form using frame controls

```
import java.awt.GridLayout;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JTextField;

public class Registration extends JFrame
{
    int j=0;
    JLabel l1,l2,l3,l4,l5,l6,l7,l8,l9;
    JTextField tf1,tf2,tf3,tf4,tf5,tf6,tf7;
    JButton btn1,btn2;
    JComboBox j1,j2,j3;
    JRadioButton rb1,rb2;
    JPanel jp1,jp2;
    ButtonGroup bg;
    String day[]=new String[31];
    String month[]=new String[12];
    String year[]=new String[67];

    public Registration() {
        for(int i=1;i<=31;i++)
        {
            day[j]=i+"";
            j++;
        }
        j=0;
        for(int i=1;i<=12;i++)
        {
            month[j]=i+"";

```

14. SWINGS AND EVENTHANDLING

```
        j++;
    }
    j=0;
    for(int i=1950;i<=2016;i++)
    {
        year[j]=i+"";
        j++;
    }
    l1=new JLabel("name");
    l2=new JLabel("email_id");
    l3=new JLabel("create pasword");
    l4=new JLabel("confirm password");
    l5=new JLabel("country");
    l6=new JLabel("state");
    l7=new JLabel("phone_no");
    l8=new JLabel("dob");
    l9=new JLabel("gender");
    rb1=new JRadioButton("male");
    rb2=new JRadioButton("female");

    tf1 = new JTextField();
    tf2 = new JTextField();
    tf3 = new JTextField();
    tf4 = new JTextField();
    tf5 = new JTextField();
    tf6 = new JTextField();
    tf7 = new JTextField();
    btn1 = new JButton("Submit");
    btn2 = new JButton("Clear");
    j1=new JComboBox(day);
    j2=new JComboBox(month);
    j3=new JComboBox(year);

    jp1=new JPanel();
    jp2=new JPanel();
    bg= new ButtonGroup();

    bg.add(rb1);
    bg.add(rb2);
    add(rb1);
    add(rb2);
    add(l1);
    add(tf1);
    add(l2);
    add(tf2);
    add(l3);
    add(tf3);
    add(l4);
    add(tf4);
```

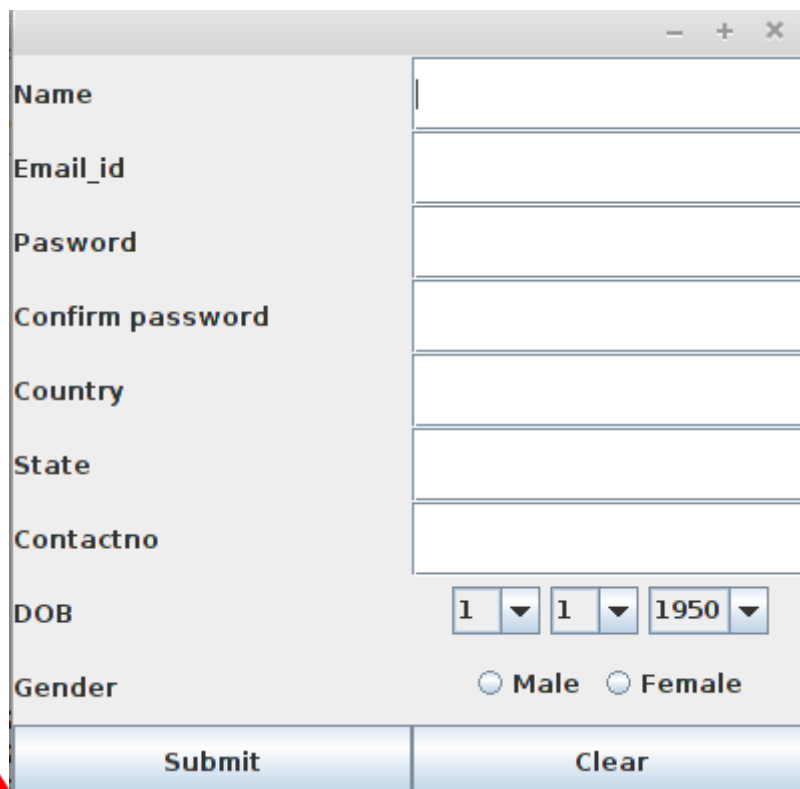
14. SWINGS AND EVENTHANDLING

```
add(l5);
add(tf5);
add(l6);
add(tf6);
add(l7);
add(tf7);
add(l8);
jp1.add(j1);
jp1.add(j2);
jp1.add(j3);
add(jp1);
add(l9);
jp2.add(rb1);
jp2.add(rb2);
add(jp2);
add(btn1);
add(btn2);

setLayout(new GridLayout(10,2));
setSize(400,400);
setVisible(true);
}

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

Output:



Name	<input type="text"/>
Email_id	<input type="text"/>
Pasword	<input type="text"/>
Confirm password	<input type="text"/>
Country	<input type="text"/>
State	<input type="text"/>
Contactno	<input type="text"/>
DOB	<input type="text" value="1"/> <input type="text" value="1"/> <input type="text" value="1950"/>
Gender	<input type="radio"/> Male <input type="radio"/> Female
<input type="button" value="Submit"/> <input type="button" value="Clear"/>	

14. SWINGS AND EVENTHANDLING

8. Program on Calculator.

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

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.JTextField;

public class Calculator extends JFrame implements ActionListener {

    JTextArea tf1;
    JButton btn0, btn1, btn2, btn3, btn4, btn5, btn6, btn7, btn8, btn9, btnadd, btnsub, btnmul,
    btnmod, btndiv, btnclr, btneql;
    JPanel jp1, jp2, jp3;
    double number1, number2;
    double addc, subc, mulc, divc, modc, result;

    public Calculator() {
        tf1 = new JTextArea(10,10);
        btneql = new JButton("=");
        btn0 = new JButton("0");
        btn1 = new JButton("1");
        btn2 = new JButton("2");
        btn3 = new JButton("3");
        btn4 = new JButton("4");
        btn5 = new JButton("5");
        btn6 = new JButton("6");
        btn7 = new JButton("7");
        btn8 = new JButton("8");
        btn9 = new JButton("9");
        btnadd = new JButton("+");
        btnsub = new JButton("-");
        btnmul = new JButton("*");
        btnmod = new JButton("%");
        btndiv = new JButton("/");
        btnclr = new JButton("CLR");

        jp1 = new JPanel();
        jp1.setLayout(new GridLayout(3, 2));
        jp1.add(tf1);
```


14. SWINGS AND EVENTHANDLING

```
add(jp1);

jp2 = new JPanel();
jp2.setLayout(new GridLayout(3, 6));
jp2.add(btn0);
jp2.add(btn1);
jp2.add(btn2);
jp2.add(btn3);
jp2.add(btn4);
jp2.add(btn5);
jp2.add(btn6);
jp2.add(btn7);
jp2.add(btn8);
jp2.add(btn9);
jp2.add(btnadd);
jp2.add(btnsub);
jp2.add(btnmul);
jp2.add(btnmod);
jp2.add(btndiv);
jp2.add(btnclr);
jp2.add(btneql);
add(jp2);

btneql.addActionListener(this);
btn0.addActionListener(this);
btn1.addActionListener(this);
btn2.addActionListener(this);
btn3.addActionListener(this);
btn4.addActionListener(this);
btn5.addActionListener(this);
btn6.addActionListener(this);
btn7.addActionListener(this);
btnsub.addActionListener(this);
btn8.addActionListener(this);
btn9.addActionListener(this);
btnadd.addActionListener(this);
btnsub.addActionListener(this);
btnmul.addActionListener(this);
btnmod.addActionListener(this);
btndiv.addActionListener(this);
btnclr.addActionListener(this);

setLayout(new GridLayout(2, 0));
setSize(500, 500);
setTitle("calculator");
setVisible(true);
}

public double number_Reader() {
```

14. SWINGS AND EVENTHANDLING

```
double num1;
num1 = Double.parseDouble(tf1.getText());
return num1;
}

@Override
public void actionPerformed(ActionEvent e) {

    if (e.getSource() == btn0) {
        tf1.setText(tf1.getText() + "0");
    }

    if (e.getSource() == btn1) {
        tf1.setText(tf1.getText() + "1");
    }

    if (e.getSource() == btn2) {
        tf1.setText(tf1.getText() + "2");
    }
    if (e.getSource() == btn3) {
        tf1.setText(tf1.getText() + "3");
    }
    if (e.getSource() == btn4) {
        tf1.setText(tf1.getText() + "4");
    }

    if (e.getSource() == btn5) {
        tf1.setText(tf1.getText() + "5");
    }
    if (e.getSource() == btn6) {
        tf1.setText(tf1.getText() + "6");
    }
    if (e.getSource() == btn7) {
        tf1.setText(tf1.getText() + "7");
    }
    if (e.getSource() == btn8) {
        tf1.setText(tf1.getText() + "8");
    }
    if (e.getSource() == btn9) {
        tf1.setText(tf1.getText() + "9");
    }
    if (e.getSource() == btnadd) {
        number1 = number_Reader();
        tf1.setText(" ");
        addc = 1;
        mulc = 0;
        divc = 0;
        subc = 0;
        modc = 0;
    }
}
```

14. SWINGS AND EVENTHANDLING

```
}

if (e.getSource() == btnsub) {
    number1 = number_Reader();
    tf1.setText(" ");
    addc = 0;
    mulc = 0;
    divc = 0;
    subc = 1;
    modc = 0;
}

if (e.getSource() == btnmul) {

    number1 = number_Reader();
    tf1.setText(" ");
    addc = 0;
    mulc = 1;
    divc = 0;
    subc = 0;
    modc = 0;
}

if (e.getSource() == btndiv) {

    number1 = number_Reader();
    tf1.setText(" ");
    addc = 0;
    mulc = 0;
    divc = 1;
    subc = 0;
    modc = 0;
}

if (e.getSource() == btnmod) {
    number1 = number_Reader();
    tf1.setText(" ");
    addc = 0;
    mulc = 0;
    divc = 0;
    subc = 0;
    modc = 1;
}

if (e.getSource() == btneql) {
    number2 = number_Reader();

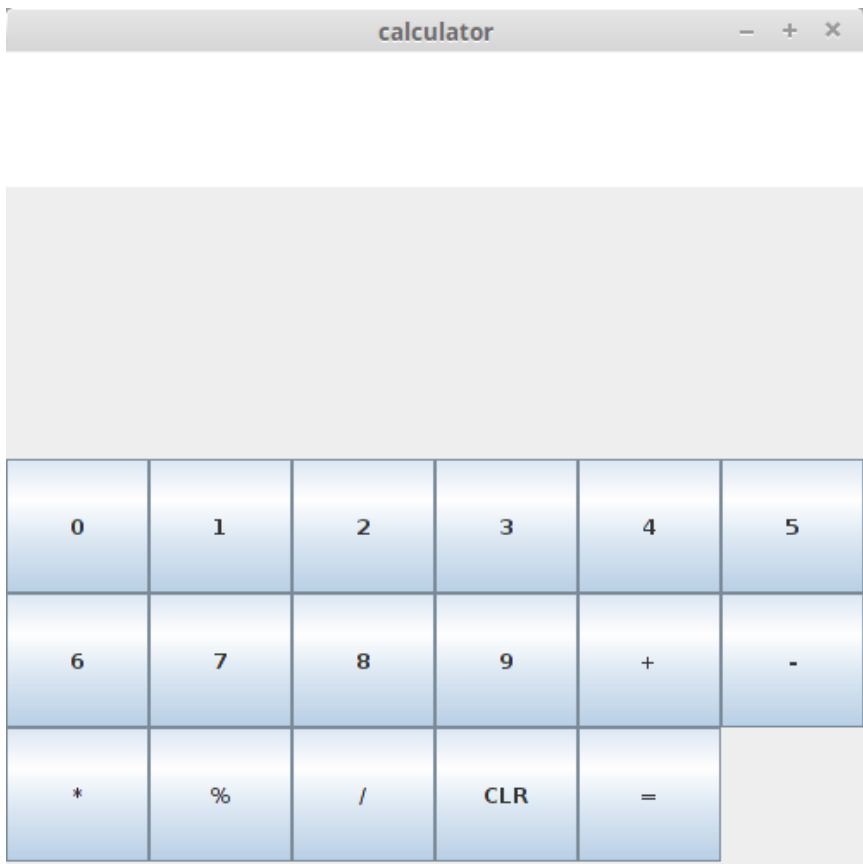
    if (addc > 0) {
        result = number1 + number2;
        tf1.setText("" + result);
    }
    if (subc > 0) {
        result = number1 - number2;
```

14. SWINGS AND EVENTHANDLING

```
        tf1.setText("" + result);
    }
    if (mulc > 0) {
        result = number1 * number2;
        tf1.setText("" + result);
    }
    if (divc > 0) {
        result = number1 / number2;
        tf1.setText("" + result);
    }
    if (modc > 0) {
        result = number1 % number2;
        tf1.setText("" + result);
    }
}

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

Output:



14. SWINGS AND EVENTHANDLING

ASSIGNMENT

1. Fill out the student application form and save the information of the student into file when student submit the form. Handle ActionEvent for Submit button.
2. Design the calculator which will perform sine, cosine, tan, cot, sec and cosec for angle.
3. Design the Converter which will convert the number to binary, decimal and hexadecimal.