Page No.
Date :

Ch-8 - Event and GUI Programming.

* Explain Event with Event Handling.

=> The change in the state of an object or behavior by performing actions is called event.

For event in java, we have to use java event packege which is java.awt.event.

There are different types of an event in java.

like, Key event, Mouseevent, Item event, Focus event, Container event.

For event, We have to require event handling.

In event handling, we have to different listener for different event

Brain Spot

like, Key event - Key Listener
Item event - Item Listener
Text event - Text Listener
Focus Event - Focus Listener

We can write, event handling code in three ways.

1) Within class
2) Other class
3) Anonymous class

1 Within class :

In this method, we have to write event handling code into the same class.

Ex.
```
import java.awt.*;
import java.awt.event;

class Java extends Frame
    implements ActionListener
{
  TextField tf = new TextField();

  Java()
  {
    tf.setBounds(60, 50, 170, 20);
    add(tf);
    setSize(300, 300);
```

BrainSpot

નિષ્ફળતા ગુપ્ત શક્તિઓને સતેજ કરે છે.

```
public void actionPerformed
        (ActionEvent e)
{
   tf.setText("Khushi");
}
public static void main(String.
    args[])
{
    new Java();
}

}
```

2 Outer class

=> In this method, we have to create class for event handling.

Ex.
```
import java.awt.*;
import java.awt.event.*;

class Java extends Frame
{
  Java()
  {
    TextField tf = new TextField();
     tf.setBounds(60, 50, 170, 20);
    Button b = new Button("click
      me");
    b.setBounds(100, 120, 80, 30);
```

Brain Spot

```
    b.addActionListener(this);
    add(b);
    add(tf);

    setsize(300, 300);
}
public void action

    Outer o = new Outer(this);
}
public static void main(
    String, args[])
{
    new Java();
}
}
```

→ Outer Class

```
import java.awt.event.*;

class Outer implements
        ActionListener
{
    Java obj;
    Outer(Java obj);
    this.obj = obj;
}
public void actionPerformed
    (ActionEvent e)
```

Brain Spot

```
    {
        obj.tf.setText("Khushi");
    }
}
```

* Explain Mouse and Key event with event handling.

=> Mouse Event:

Mouse event is use to performed action using mouse.

This are the different Mouse Event.

Ex. mouseClicked, mouseEntered, mouseExited, mousePressed, mouseReleased.

For mouse event, we have to use MouseListener for mouse event handling.

Ex.
```
import java.awt.*;
import java.awt.event.*;

class Java extends Frame
  implements MouseListener
{
```

Brain Spot

```
Java()
{
  Label l = new Label();
  addMouseListener(this);
  l.setBounds(20, 50, 100, 20);
  add(l);
  setSize(300, 300);
}

public void mouseClicked
      (MouseEvent e)
{
  l.setText("Clicked");
}

public void mouseEntered
    (MouseEvent e)
{
  l.setText("Entered");
}

public static void main(String,
      args[])
{
  new Java();
}

}
```

BrainSpot

=> Key event :

Key event is used to perform action using Keyboard.

There are different types of Keyboard event.

Ex. KeyPressed (KeyEvent e);
KeyReleased (KeyEvent e);
KeyTyped (KeyEvent e);

For Key event, we have to use KeyListener.

Ex.
```
import java.awt.*;
import java.awt.event.*;

class Java extends Frame
  implements KeyListener
{
   Java()
   {
     Label l = new Label();
     l.setBounds(20, 50, 100, 20);
     add(l);
     area = Textarea area = new
     setsize(400, 400);          Textarea
     area.setBounds(20, 80, 300, 300);
     area.addKeyListener(this);
     add(area);
```
}

Brain Spot

```
public void keyPressed(Key
    Event e)
{
    l.setText("Key Pressed");
}

public static void main
    (String. args[])
{
    new Java();
}

}
```

* Explain Panel and Frame with example.

=> Panel:

Panel is an AWT component which represents a simple container that can attach other GUI components.

Panel does not contain any title bar and border.

Panel can have more than one in Frame.

Brain Spot

Without using Frame, we cannot add Panel.

=> Frame :

Frame is an AWT component which is a top level window.

Frame have a title bar and Border.

Multiple Frame is not possible. We can not add One Frame into the other Frame.

Ex.

```
import java.awt.*;

class Java
{
   Java()
   {
     Frame F = new Frame();
     Panel P = new Panel();
       P.setBounds(40, 80, 200, 200);
       P.setBackground(Color.red);
     F.add(P);
     f.setsize(400, 400);
   }
```

Brain Spot

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

}
```

Brain Spot

* Explain Layout Managers with its types.

=> The Layout Managers are used to arrange components in a particular manner.

The Java Layout Managers control the positioning and size of the components in GUI Forms.

There are Three types of Layout Managers.

1) Border Layout
2) Flow Layout
3) Grid Layout

1 Border Layout :

The Border Layout is used to arrange the components in five regions : North, South, East, Wast and Center.

Each region may contain one component only.

We can create Border Layout with two type.

Brain spot

(ii) BorderLayout(): Create border Layout with no gaps between the components.

(iii) BorderLayout(int hgap, int vgap): Create Border Layout with horizontal and vertical gaps between the components.

Ex.

```
import java.awt.*;
import java.swing.*;

class Java
{
   Java()
{
   JFrame F = new JFrame();

   JButton b1 = new JButton("1");
   JButton b2 = new JButton("2");
   JButton b3 = new JButton("3");
   JButton b4 = new JButton("4");
   JButton b5 = new JButton("5");

   F.add(b1, BorderLayout.NORTH);
   F.add(b2, BorderLayout.SOUTH);
   F.add(b3, BorderLayout.EAST);
   F.add(b4, BorderLayout.WEST);
   F.add(b5, BorderLayout.CENTER);
```

Brain Spot

```
        F.setsize(300, 300);
    }

    public static void main(String
        args[])
    {
        ~~Ja~~ t new Java();
    }

}
```

2. Flow Layout:

The Java Flow Layout is used to arrange the components in a line means in one flow.

We can create Three type Flow Layout:

(i) FlowLayout(): Flow Layout with centered alignment.

(ii) FlowLayout(int align): Flow Layout with given alignment.

(iii) FlowLayout(int align, int hgap, int vgap): Flow Layout with given alignment and given horizontal and vertical gap.

Brain Spot

Ex
```
import java.awt.*;
import java.swing.*;

class Java
{
    Java()
    {
        JFrame f = new JFrame();

        JButton b1 = new JButton("1");
        JButton b2 = new JButton("2");
        JButton b3 = new JButton("3");

        b1.add
        f.add(b1);
        f.add(b2);
        f.add(b3);

        f.setLayout(new FlowLayout());

        f.setSize(300, 300);
    }

    public static void main(String.
        args[])
    {
        new Java();
    }

}
```

Brain Spot

3 Grid Layout:

The Java Grid Layout is used to arrange the components in a rectangular grid.

One component is display in each rectangle.

We can Create GridLayout in Three ways.

(i) GridLayout(): Create a grid Layout with one column per component in a row.

(ii) GridLayout(int rows, int columns): Create a grid Layout with the given row and columns.

(iii) GridLayout(int rows, int columns, int hgap, int vgap): Create a grid Layout with given row and column and given horizontal and vertical gaps.

Ex.
```
import java.awt.*;
import java.swing.*;

class java
```

Brain Spot

```
{
    Java ()
    {
    JFrame F = new JFrame();

    JButton b1 = new JButton("1");
    JButton b2 = new JButton("2");
    JButton b3 = new JButton("3");

     F.add(b1);
     F.add(b2);
     F.add(b3);

     F.setLayout(new GridLayout())

     F.setSize(300, 300);
    }

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

}
```

Brain Spot

* Explain GUI Components.

=> This are the Basic GUI components.

(a) Button():

A Button is a labeled component when clicked performs an event.

Syntax:

```
Button button = new Button
        object          ("name");
```

Using this method we can Pass value on the button.

(b) Check Box():

It is used to create a Checkbox in the container.

It can get a value either true or false by checking and unchecking the checkbox.

Syntax:

```
Checkbox checkbox = new Checkbox
        object      ("value", true
                              or
                              null);
```

Brain Spot

(c) RadioButton ():

It is use to select one option from the multiple option.

It should be added in Button Group to select one radio button only.

Syntax:

```
RadioButton RadioButton = new
                          Object
    RadioButton("Value");
```

(d) Label :

It is use to show text in the container.

It will displays text in the form of Read-Only.

Syntax:

```
Label Label = new Label("Value");
      Object
```

Brain Spot

(e) Textarea:

Textarea is used to entered text in the Frame.

It allows the editing of multiple line text.

Syntax:

```
TextArea Textarea = new
                    object
                    TextArea("Value");
```

(f) TextField:

TextField is used to entered text in the Frame.

It allows to editing only single line text.

Syntax:

```
TextField Textfield = new
                      Object
                      TextField("Value");
```

Brain Spot

(g) ComboBox:

The Object of choice is used show popup menu of choice.

Choice selected by user is shown on the top of a menu.

Syntax:

```
ComboBox Combobox = new
                    Object

    ComboBox(Combobox value);
                    name
```

(h) List:

The List Object creates a list of item in which we can choose one or multiple item at a time.

Syntax:

```
List List   = new List(List
    Object                size);
```

Brain Spot

(i) Scrollbar :

Object of scrollbar class is used to add horizontal and vertical scrollbar.

Scrollbar is allows us to see invisible number of rows and column.

Syntax :

Scrollbar Scrollbar = new Scrollbar();
Object

(ii) Slider :

Object of Slider class is used to create slider in Frame.

Using Slider we can select specific range.

Syntax :

Slider Slider = new
Object

Slider (horizontal
or
Vertical
side

Brain Spot

Slider(Slider.side, min max value, value, specific value);

CK) Menu:

The object of Menu class adds simple labeled menu item on menu.

Syntax:

Menu Menu Object = new Menu("Menu name");

For add Item in Menu,

MenuItem MenuItem = new object

MenuItem("Item name");

* Explain GUI Container:

Brain Spot

(L) Windows :

The window is a container that doesn't include borders and menu bar.

We must use another window, Frames and dialogue box to create a window.

(M) Frame :

(N) Panel.

* Explain Applet. with its write method.

=> Applet is a special type of Program that is embedded in the webpage to generate the dynamic content.

Applet runs inside the browser and works at client side.

Applet does not have any type of main method in the program.

Brain Spot

Applets are not stand-alone programs.

Applets run either within a web browser or an applet viewer.

We can write Applet into the two Method.

1) By html File

2) By appletViewer tool

1 By html File:

- To execute the applet by html file, create an applet and create class file.

After that create an html file and place the applet code in html file.

Ex. Java Applet code File:

```
import java.applet.Applet;
import java.awt.Graphics;

class Java extends Applet
```

Brain Spot

```
{
    public void paint(Graphics g)
    {
        g.drawString("Khushi", 150, 150);
    }
}
```

- html File:

```
<html>
<body>

<applet code = "Java.class" width =
    "300" height = "300" >
</applet>

</body>

</html>
```

2 By applet Viewer tool

- To execute the applet by applet viewer, we have to create java applest file.

And Inside this file we have to write applet code in the comment.

Brain Spot

Ex.
```
import java.applet.Applet;
import java.awt.Graphics;

class java extends Applet
{
   public void pain(Graphics g)
   {
      g.drawstring("Khushi", 150, 150)
   }
}

/*

<applet code = "Java.class"
 width = "300" height = "300">
</applet>

*/
```

- Drawback of Applet

For Applet run, we have to required plugin at client browser.

Brain Spot

* Explain Applet life Cycle.

=> This are the Stages of Applet Life Cycle.

1) Initializing an Applet
2) Starting the Applet
3) Painting the Applet
4) Stopping the Applet
5) Destroying the Applet

1 Initializing an Applet:

This stage is use to initialize the applet.

For initializing the applet, we have to use init() method.

Syntax:

```
public void init()
{
    // code
}
```

In this method, Variables can be initialize.

This method can be called only once during the run time of Applet.

Brain Spot

2 Starting the Applet:

After the inilization, we have to use starting method stage.

For starting the Applet, we have to use start() method.

Syntax:

```
public void start()
{
   // code
}
```

This method is called to restart an applet after it has been stopped.

3 Painting the Applet.

After the starting the Applet, we have to use Painting the Applet.

For Painting the Applet, we have to use paint() method.

Brain Spot

Syntax:

```
public void paint(Graphics graphics)
{
    // code
}
```

Paint() method is used for painting any shapes like square, rectangle, etc.

Paint() method has one parameter of type Graphics class.

4 Stopping the Applet

For stopping the applet, we have to use this method.

For Stopping the applet, we have to use stop() method.

Syntax:

```
public void stop()
{
    // code
}
```

After stop() method, we can also

Brain Shot

use start() method.

The stop() method is called when a web browser leaves the HTML Document.

5 Destroying the Applet

This method is use to destroy the Applet.

For Destroying the Applet, we have to use destroy() method.

Syntax:

```
public void destroy()
{
  // code
}
```

Once applet is destroyed, we can not start the applet.

Brain Spot