Date
Page

Types of Events

There are two types of Events in Java.

i) The Foreground Events:-

- The foreground events are those events that requires direct interaction of the user.
- These types of events are generated as a result of user interaction with the GUI components.
- For example :- clicking a button, mouse movement, pressing a keyboard key, selecting an option from the list etc.

ii) The Background Events :-

- The Background events are those events that result from the interaction of the end-user. For eg:- an operating system interrupts system failure (Hardware or software).
- To handle these events, we need as an event handling mechanism that provides control over the events and responses.
- For example :- Hardware or software failure, timer expires etc.

⇒ The modern approach to handling events is based on the delegation event model.

Date
Page

Chapter 5 :- Events, Handling Events and AWT/swing.

Events

- Changing the state of an object is known as an event.
- An event in Java is an object that is created when something within a graphical user interface.
- For example :- click on button, dragging a mouse, etc.

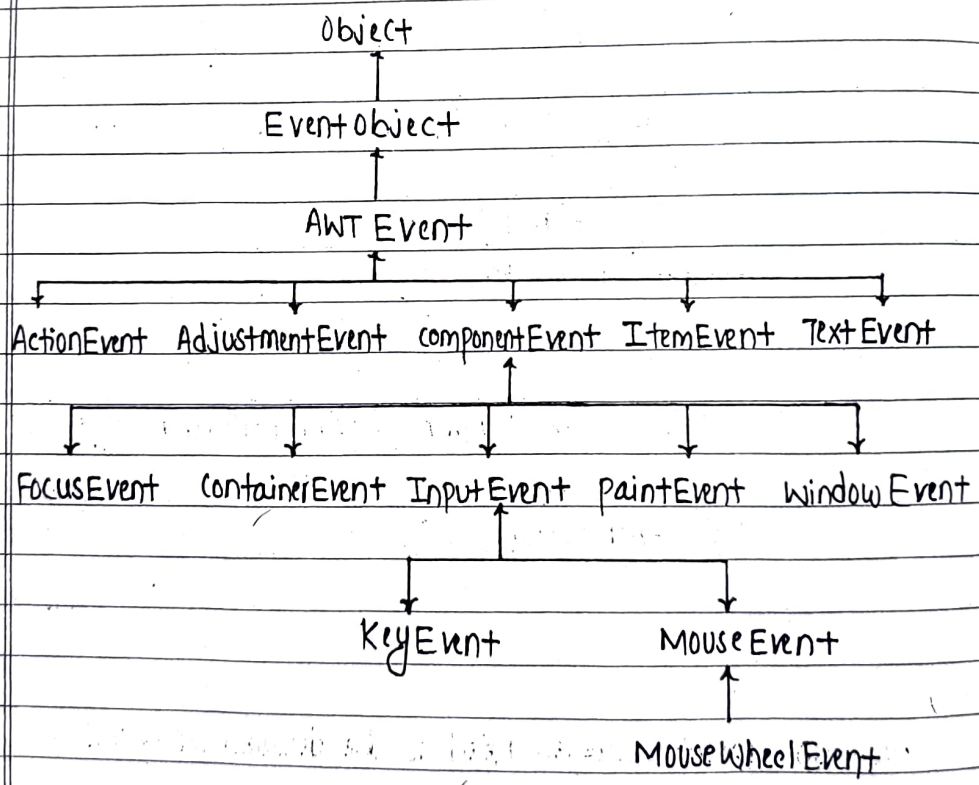


Fig:- Abstract window ToolKit (AWT) Event class Diagram.

- The java.awt.event packages provides many event classes and Listener interfaces for event handling.

- Event handling is the mechanism that controls the event and decides what should happen if an event occours.

- This mechanism have the code which is known as event handler that is enewked when an event occours.

- Java uses the delegation Event model to Handle the events. This model defines the standard mechanism to generate and handle the events.

- This Event model is based on the three components they are:-

i) Events ii) Event Sources iii) Events Listeners

ii) Event sources:- A source is an object that causes and generates an event. when the internal state is changed of object than it generates an event. Sources are allowed to generate different types of event.

- A source must register a listener to receive notification for a specific events. Each event contains registration methods. Below is an example:

public void addTypeListener (TypeListener e1)

from the above syntax, the Type is the name of an event, and e1 is a event to a event Listener. eg:- For Keybord event Listener the method as addKeyListener(), similarly for addMouseListener event has method addmouseMotionListener(). when an event is triggered using the respected Source, all events will be notified to ngisters listeners and receive the event object. This process is known as event Multicasting. In fewer cases, the event notification will only be sent to listeners that registers to receive the

- some listeners allow only listener to rgister. For eg:-
Public void addTypeListener (TypeListner e2) throws java.util.TooManyListenersException.
From the above syntax, the Type is the name of event, and e2 is the event listener's refrence. When the specified event occours, it will be notified to the registered listeners. This process is known as unicasting events.

- A source should contain a method that unregisters a specific type of event from the listener. if not needed, Below eg of the method that will remove the event from the listener.

Public void removeListener has it's method:
public void removeTypeListener (TypeListner e2?)

From the above syntax, the Type is an event name and e2 is the refrence of the listener. For eg:- To remove the keyboard listener, the removeKeyListener() method is called.

iii) Event Listeners:-

- It is also known as event handler. Listner is responsible for generating response to an event.

- From java implementation point of view the listener is also an object. Listeners waits until it receives an event. once an event is recieved, the Listener process the event an then returns.

- For eg:- MouseMotionListener interface provides two methods when the mouse is dragged and moved. if it implements this then any object can recein and process these events.

Date
Page

AWT vs Swing

AWT	Swing
• A collection of GUI components and other related services required for GUI programming in Java.	• A part of Java foundation classes (JFC) that is used to create Java-based Frontend GUI applications.
• Its components are Heavy-weight and they are platform dependent.	• Its component are Light weight and they are platform independent.
• It has less advance components.	• It has more advance components.
• Its Execution is slower.	• It's Execution is faster.
• It don't support MVC pattern.	• It supports MVC pattern.
• components requires more memory spaces.	• component requires less memory spaces.
• It doesn't support a pluggable look and feel.	• It supports a pluggable look and feel.
• AWT programs are not portable.	• Swing programs are portable.
• It is an old framework for creating GUIs.	• Swing is new framework for creating GUI's.

• programmers has to import java.awt packages to develop an AWT-based GUI.	• programmers has to import javax.swing packages to develop an swing-based GUI.
• It has less rich set of object as compared to swing	• It has very rich set of object than AWT.
eg:- JAVA AWT to print Hello world.	eg:- JAVA swing to print Hello world.

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

class Hello{
Public static void main
    (String[] args){
Frame f=new Frame("T1");
Label l=new Label("Hello
 World!", Label.CENTER);

f.add(l);
f.setSize(300,100);
f.setVisible(true);

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

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

class World{
public static void main(String[] args){
JFrame f=new JFrame("T2");

f.setMinimumSize(new
 Dimension(800,600));

f.setDefaultCloseOperation(JFrame.
          EXIT_ON_Close);

JLabel l = new JLabel("Hello World!",
SwingConstants.CENTER);

f.getContentPane().add(l);

f.Pack();
f.setVisible(true);
}
}
```

Date
Page

Event handling in AWT using ActionListener in three ways.

1) Within class

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

class Example extends Frame implements ActionListener{

    TextField tf;

Example(){
    tf = new TextField();
    tf.setBounds(60,50,170,20);

Button b = new Button("click Me");
    b.setBounds(100,120,80,30);

b.addActionListener(this);

    add(b); add(tf);
    SetSize(300,300);

    SetLayout(null);
    SetVisible(true);
}
 public void actionPerformed(ActionEvent e){
    tf.setText("Welcome");
}
    Public static void main(String[] args){
        new Example();
    }
}
```

2) Outer class Save as :- Example.java

```
import java.awt.*;
import java.awt.event.*;
class Example extends Frame {
    TextField tf;
  Example() {
    tf = new TextField();
    tf.setBounds(60,50,170,20);

  Button b = new Button("click me");
    b.setBounds(100,120,80,30);

  outer o = new outer(this);
    b.addActionListener(o);
    add(b); add(tf);
    setSize(300,300);
    setLayout(null);
    setVisible(true);
}
    public static void main(String[] args){
        New Example();
    }

// save by Outer.java
    import java.awt.event.*;
class outer implements ActionListener{
    Example obj;
  outer(Example obj){
        this.obj = obj; }
  public void actionperformed(ActionEvent e){
    obj.tf.setText("welcome");
  }
}
```

3) Anonymous class

```
import java.awt.*;
import java.awt.event.*;
class Example extends Frame {
    TextField tf;

    Example() {
        tf = new TextField();
        tf.setBounds(60,50,170,20);
    Button b = new Button("click me");
        b.setBounds(50,120,80,30);

b.addActionListener(new ActionListener(){
        public void actionPerformed(){
                tf.setText("welcome"); }
        });

    add(b); add(tf);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
}

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


Java Swing

Java swing is a part of JFC that is used to create window-based applications.

Hierarchy of java swing

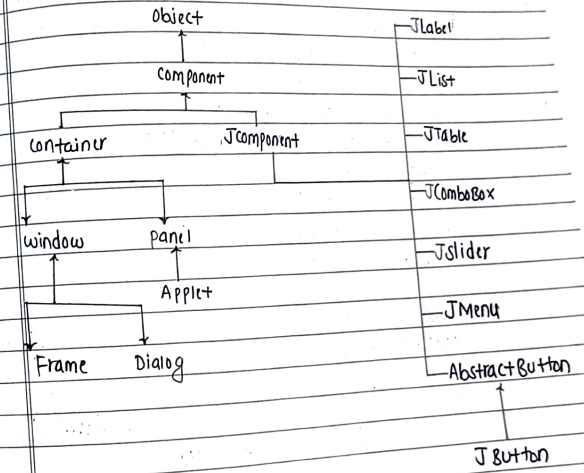


Fig:- Java swing Hierarchy

Swing event handling is same as java AWT but use import javax.swing.*; instead of import java.awt.*; and use Jcomponent instead of Component.

Date
Page

Textinputchoice.

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

Public class ChoiceExample {

ChoiceExample() {
  Frame f= new Frame();

final Label label = new Label();
  label.setAlignment (Label.CENTER);
  label.setSize(400,100);

Button b= new Button ("show");
  b.setBounds (200,100,75,75);

final Choice c= new Choice();
 c.setBounds (100,100,75,75);

  c.add("c"); c.add ("c++"); c.add ("java");
    c.add("Php"); c.add ("Android");

f.add(c); f.add(label); f.add(b);
  f.setSize (400,400)

f.setlayout(null);

f.setVisible (true);
```

Date
Page

```
b.addActionListener (new ActionListener() {

public void actionPerformed (ActionEvent e) {

String data = "programming language selected:" + c.
        getItem(c.getSelectedIndex());
  label.setText(data); }

});

}

  Public static void main (String[] args) {

        new ChoiceExample();

  }

}
```

- The object of choice class is used to show popup menu of choices.
- Choice selected by used user is shown on the top of a menu.
- It inherits Component class.

* AWT choice class Declaration.

  Public class choice extends Component implements ItemSelectable, Accessible.

Date / Page

Textinput Choice

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

Public class ChoiceExample {

ChoiceExample() {
  Frame f= new Frame();

final Label label = new Label();
  label.setAlignment (Label.CENTER);
  label.setSize(400,100);

Button b= new Button ("show");
  b.setBounds (200,100,75,75);

final Choice c= new choice();
  c.setBounds (100,100,75,75);

  c.add("c"); c.add ("c++"); c.add ("java");
    c.add("Php"); c.add ("Android");

f.add(c); f.add(label); f.add(b);
  f.setSize (400,400)

f.setLayout(null);

f.setVisible (true);
```

Date / Page

```
b.addActionListener (new ActionListener() {

Public Void actionPerformed (ActionEvent e) {

String data = "Programming language selected:" + c.
        getItem(c.getSelectedIndex());
  label.setText(data); }

});
}

  Public static Void main (String[] args) {

        new ChoiceExample();
  }
}
```

- The object of choice class is used to show popup menu of choices.
- Choice selected by used user is shown on the top of a menu.
- It inherits component class.

* AWT choice class Declaration.

  Public class choice extends component implements ItemSelectable, Accessible.

Java 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.

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

Class scrollBarExample{

scrollBarExample(){

    JFrame f= new JFrame("scrollbar Example");
final JLabel label = new JLabel();

label.setHorizontalAlignment (JLabel.CENTER);

    label.SetSize(400,100);

final JScrollBar s = new JScrollBar();
    s.setBounds(100,100,50,100);

f.add(s); f.add(label);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);

s.addAdjustmentListener(new AdjustmentListener(){
```



```
public void adjustmentValueChanged(AdjustmentEvent e){

label.setText("Vertical scrollbar value is:" +
                    s.getValue()); }

});

}

public static void main(String[] args){

        new scrollBarExample();

}

}
```

Java JMenuBar, JMenu and JMenuItem

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

JMenuBar class declaration

```
Public class JMenuBar extends JComponent implements
    MenuElement, Accessible
```

Date
Page

- JMenu class declaration

Public class JMenu extends JMenuItem implements MenuElement, Accessible

- JMenuItem class declaration

Public class JMenuItem extends AbstractButton implements Accessible, MenuElement.

Example

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

public class MenuExample implements ActionListener {
    JFrame f;
    JMenuBar mb;
    JMenu file, edit, Help;
    JMenuItem cut, copy, paste, selectAll;
    JTextArea ta;

    MenuExample(){

    f=new JFrame();
    cut=new JMenuItem("cut");
    copy = new JMenuItem("copy");
    paste= new JMenuItem("paste");
    selectAll = new JMenuItem("selectAll");
```

Date
Page

```
    cut.addActionListener(this);
    copy.addActionListener(this);
    paste.addActionListener(this);
    selectAll.addActionListener(this);

    mb=new JMenuBar();
    file=new JMenu("File");
    edit= new JMenu("Edit");
    Help= new JMenu("Help");

    edit.add("cut"); edit.add("copy"); edit.add("paste");
    edit.add("selectAll"); //Dont use double quotes.

    mb.add(file); mb.add(edit); mb.add(Help);

    ta=new JTextArea();
    ta.setBounds(5,5,360,320);

    f.add(mb); f.add(ta);

    f.setJMenuBar(mb);

    f.setLayout(null);

    f.setSize(400,400);
    f.setVisible(true);

}
```

```
public void actionPerformed(ActionEvent e){
    if(e.getSource() == cut)
        ta.cut();
    if(e.getSource() == copy)
        ta.copy();
    if(e.getSource() == paste)
        ta.paste();
    if(e.getSource() == selectAll)
        ta.selectAll();
}

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

Dialog Box in Java. (JOptionPane)

- The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box.
- These dialog boxes are used to display information or get input from the user. The JOptionPane class inherits JComponent class.
- JOptionPane class declaration

  Public class JOptionPane extends JComponent implements Accessible.

i) Java JOptionPane Example showMessageDialog()

eg 1)

```
import javax.swing.*;

Public class OptionPaneExample {
    JFrame f;
    OptionPaneExample(){
        f=new JFrame();
        JOptionPane.showMessageDialog(f,"Hello, I am Dialogbox.");
    }
    Public static void main(String[] args){
        new OptionPaneExample();
    }
}
```

Eg 2):- import javax.swing.*;

```
public class optionpaneExample {
    JFrame f;
optionpaneExample(){
    f=new JFrame();
JOptionpane.showMessageDialog(f,"successfully updated.",
    "Alert",JOptionpane.WARNING_MESSAGE);
}
    public static void main(String[] args){
        new optionpaneExample(); }
}
```

2) ShowInputDialog()

```
        import javax.swing.*;
class optionpaneExample {
    JFrame f;

optionpaneExample(){
    f= new JFrame();
String name=JOptionpane.showInputDialog(f,"Enter Name");
}
    public static void main(String[] args){
        new optionpaneExample(); }
}
```


iii) ShowConfirmDialog()

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

public class optionpaneExample extends WindowAdapter{
    JFrame f;

optionpaneExample(){
    f=new JFrame();
    f.addWindowListener(this);

    f.setSize(300,300);

    f.setLayout(null);

f.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    f.setVisible(true); }

public void windowClosing(WindowEvent e){
int a=JOptionpane.showConfirmDialog(f,"Are you sure?");

    if(a=JOptionpane.YES_OPTION){
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}
    public static void main(String[] args){

        new optionpaneExample(); }
}
```

Date
Page

Java Layout Management (Layout Manager)

- The LayoutManagers are used to arrange components in a particular manner.

- LayoutManager is an interface that is implemented by all the classes of layout managers. There are following classes that represents the layout managers:

i) Java.awt.BorderLayout
ii) java.awt.FlowLayout
iii) java.awt.GridLayout
iv) java.awt.GridBagLayout
v) java.awt.cardLayout
vi) javax.swing.BoxLayout
vii) javax.swing.GroupLayout
viii) javax.swing.scrollpaneLayout
ix) javax.swing.SpringLayout

- Layout Managers are software components used in widget toolkits which have the ability to lay out graphical control elements by their relative positions without using distance units.

- A layoutManager is an object that controls the size and the position of components in a container.

- Every container object has a LayoutManager object that controls its Layout.

Q NO 4a)

i) BorderLayout

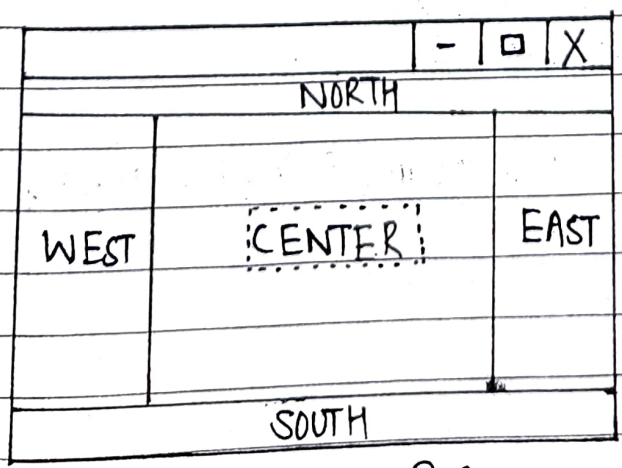
- The BorderLayout is used to arrange the components in five regions: North, south, east, west and center.
- Each region(area) may contain one component only. It is the default layout of frame or window.
- The BorderLayout provides five constants for each region:

a) public static final int NORTH
b) public static final int SOUTH
c) public static final int EAST
d) public static final int WEST
e) public static final int CENTER

Constructors of BorderLayout class.

- BorderLayout(): It creates a border layout but with no gaps between the components.
- JBorderLayout(int hgap, int vgap): It creates a border layout with the given horizontal and vertical gaps between the components.

eg:-



181546

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

public class Border {

    JFrame f;

    Border() {

        f=new JFrame();

    JButton b1= new JButton ("NORTH");;
    JButton b2= new JButton ("SOUTH");;
    JButton b3= new JButton ("EAST");;
    JButton b4= new JButton ("WEST");;
    JButton b5= new JButton ("CENTER");;

      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);

      f.setSize(300,300);
      f.setVisible(true);
    }

    Public static void main(String[] args) {

        new Border();
    }
}
```

181546

Date
Page

ii) FlowLayout.

- The FlowLayout is used to arrange the components in a line, one after another (in a flow). It is a default layout of applet or panel.

- Fields of FlowLayout class

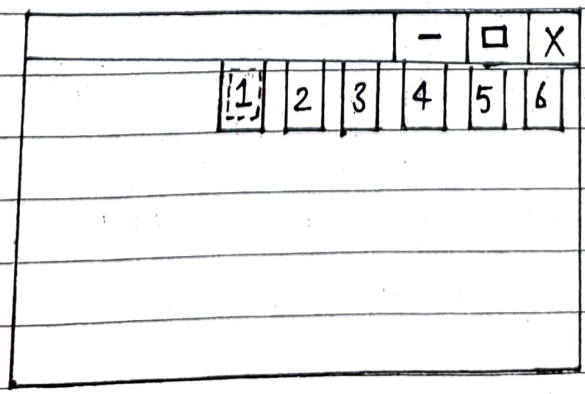
a) public static final int LEFT
b) public static final int Right
c) public static final int CENTER
d) public static final int LEADing
e) public static final int TRAILING

- Constructors of FlowLayout class

- FlowLayout(): Creates a flow layout with centerd alignment and a default 5 unit horizontal and vertical gap.
- FlowLayout(int align): Creates a flow layout with the given alignment and a default 5 unit Horizontal and vertical gap.

- FlowLayout (int align, int hgap, int vgap): Creates a flow layout with the given alignment and the given horizontal and vertical gap.

eg:-



181546

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

Public class MFlowLayout {
    JFrame f;

MFlowLayout() {

        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");
    JButton b6 = new JButton ("6");

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

f.add(b4); f.add(b5); f.add(b6);

    f.setLayout(new FlowLayout(FlowLayout.RIGHT));
    f.setsize (300,300);
    f.setVisible (true); }

Public static void main (String[] args) {
    new MFlowLayout (); }
}
```

(VVI)

Date
Page

iii) and iv) Differences between GridLayout and GridBagLayout.

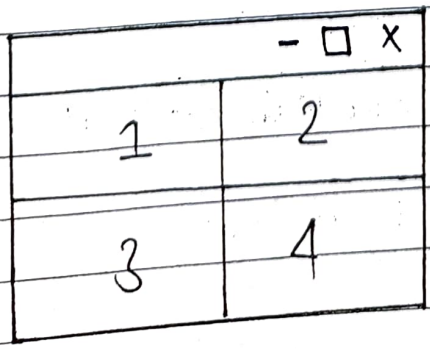
GridLayout

- A GridLayout puts all the components in a rectangular grid and is divided into equal-sized rectangles and each component is placed inside a rectangle.
- Components are placed in columns and rows.
- It takes two parameters that are column and row such as GridLayout (int rows, int columns).

Constructor of GridLayout

- GridLayout(): creates a grid layout with one column per component in a row.
- GridLayout (int rows, int columns): creates a grid Layout with the given rows and columns but no gaps between the components.
- GridLayout (int rows, int columns, int hgap, int vgap): creates a grid layout with the given rows and column along with horizontal and vertical gaps.

eg:-



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

Public class MyGridLayout(){

JFrame f;

MyGridLayout(){

    f=new JFrame();

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


f.add(b1);
f.add(b2);
f.add(b3);
f.add(b4);

f.setLayout(new GridLayout(2,2));

    f.setSize(300,300);
    f.setVisible(true);
}

Public static void main(String[] args){
        new MyGridLayout();
    }
}
```

Date
Page

GridBagLayout

- The java GridBagLayout class is used to align components vertically, Horizontally or along their baseline.

- The components may not be of same size. Each GridBagLayout object maintains a dynamic, rectangular grid of cells.

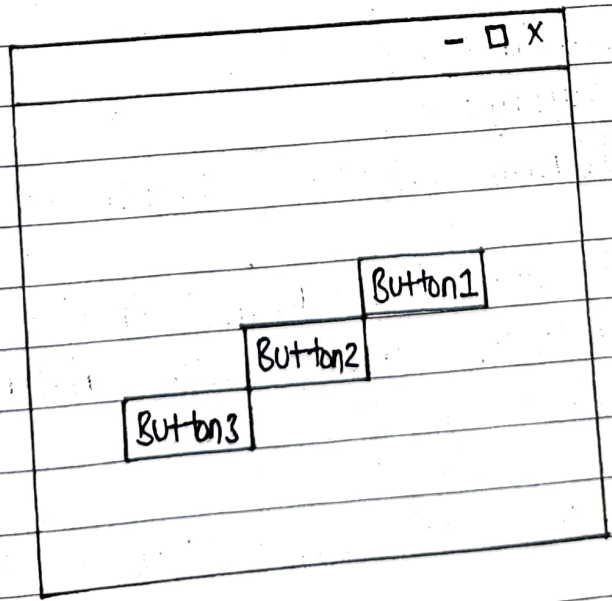
- Each component occupies one or more cells known as it's display area.

- The GridBagLayout manages each component's minimum and preffend sizes in order to determine component's size.

- Each component associates an instance of GridBagConstraints. with the help of constraints object we arrange component's display area of the grid.

eg:-

Output



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

Public class GridBagLayoutTest extends JFrame{

    Public GridBagLayoutTest(){

    setTitle("GridBaglayout Test");
    setlayout(new GridBagLayout());

GridBagconstraints gbc = new GridBagconstraints();

    gbc.gridx = 5;
    gbc.gridy = 0;

add(new JButton("Button 1"),gbc);

    gbc.gridx=0;
    gbc.gridy= 5;
add(new JButton("Button 2"),gbc);

    gbc.gridx= 2;
    gbc.gridy = 4;
    add(new JButton("Button 3"),gbc);
}

    Public static void main(String[] args){
GridBaglayoutTest gbcTest = new GridBaglayoutTest();
    gbcTest.setsize(300,300);
    gbcTest.setVisible(true);
    gbcTest.setDefaultcloseOperation(JFrame.EXIT_ON-
                                     CLOSE);
    }
}
```

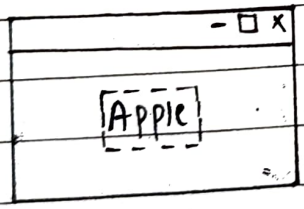
Date
Page

v) cardLayout

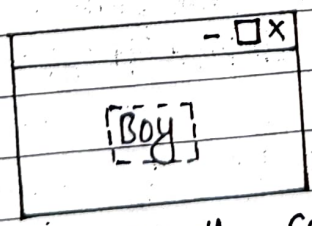
- Java cardLayout class manages the components in such a manner that only one component is visible at a time.
- It treats each component as a card that is why it is known as cardLayout.
- constructors of cardLayout
- cardLayout() :- create a cardLayout with zero horizontal and vertical gap.
- cardLayout(int hgap, int vgap) : creates a card layout with the given horizontal and vertical gap.

eg:- output

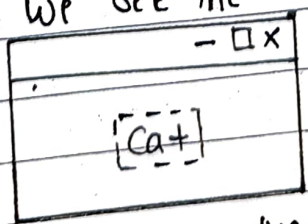
When we run the program Apple is seen.



When we click it we see, the Boy



again, we click it, we see the cat



Only one item is visible at a time.

Date
Page

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

Public class cardLayoutExample extends JFrame implements
 ActionListner{

 CardLayout card;
  JButton b1,b2,b3;
Container c;

CardLayoutExample(){

   c=getContentpane();
Card = new CardLayout(40,30);
  c.setLayout(card);

b1 = new JButton("Apple");
b2 = new JButton("Boy");
b3 = new JButton("cat");

b1.addActionListener(this);
b2.addAction Listner(this);
b3.addActionListener(this);

c.add("a",b1); c.add("b",b2); c.add("c",b3);
}

public void actionperformed(ActionEvent e){
      card.next(c);
}
```

```
public static void main (String [] args) {

CardLayoutExample cl = new CardLayoutExample ();

    cl.setSize (400,400);
    cl.setVisible (true);

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

vi) BoxLayout

- The BoxLayout is used to arrange the components either vertically or horizontally. For this purpose, BoxLayout provides four constants. They are as follows:-

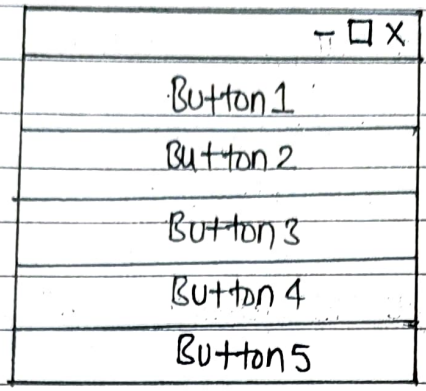
1) public static final int X_AXIS
2) public static final int Y_AXIS
3) public static final int LINE_AXIS
4) public static final int PAGE_AXIS

- Constructors of BoxLayout

BoxLayout (Container c, int axis): Creates a box layout that arranges the components with the given axis.

Date
Page

eg 1:- BoxLayout class with Y-AXIS.




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

Public class BoxLayoutExample1 extends Frame{

    Button buttons[];

Public BoxLayoutExample1(){
    buttons = new Button[5];
for(int i=0; i<5; i++){

  buttons[i] = new Button("Button" + (i+1));
    add(buttons[i]);
}
  SetLayout(new BoxLayout(this, BoXLayout.Y-AXIS));
setsize(400,400);
  Setvisible(true); }

  Public static void main(String[] args){
BoxLayoutExample1 b= newBoxLayoutExample1();
    }
}
```

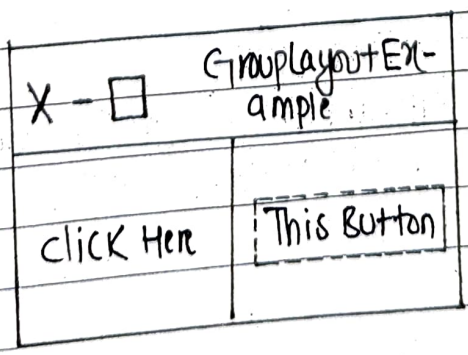
Date
Page

vii) Grouplayout in Java

- GroupLayout groups its components and places them in a container hierarchically. The grouping is done by instances of the Group classes.
- Group is an abstract class and two concrete classes which implement this group class are SequentialGroup and ParallelGroup.
- SequentialGroup positions its child sequentially one after another where as ParallelGroup aligns its child on top of each other.
- The GroupLayout class provides methods such as createParallelGroup() and createSequentialGroup() to create groups.
- Grouplayout treats each axis independently. That is, there is a group representing the horizontal axis, and a group representing vertical axis. Each component must exists in both a horizontal and vertical group, otherwise an IllegalStateException is thrown during layout, or when the minimum, preferred or maximum size is requested.
- constructors :- Grouplayout (container host) :- It creates a grouplayout for the specified constructor.

eg:-

output :-



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

JFrame frame = new JFrame ("GroupLayoutExample");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

Container contentpanel = frame.getContentPane();

GroupLayout groupLayout = new GroupLayout (contentpanel);

 contentpanel.setLayout (groupLayout);

JLabel clickMe = new JLabel ("click Here");
JButton button = new JButton ("This Button");

      groupLayout.setHorizontalGroup (
       groupLayout.createSequentialGroup()
                .addComponent (clickme).
                .addGap (10, 20, 100)
                .addComponent(button));

// groupLayout.setHorizontalGroup this creates Horizontal group.

groupLayout.setVerticalGroup(
groupLayout.createParallelGroup (GroupLayout.Alignment.BASELINE
                .addComponent (clickme);
                .addComponent (button) );
   frame.pack ();
   frame.setVisible(true);
}
}
```

Date
Page

viii) scrollpane Layout

- A JScrollPane is used to make scrollable view of a component. When screensize is limited, we use a scroll pane to display a large component or a component or a component size can change dynamically.

Constructor

JScrollPane() JScrollPane(component) JScrollPane(int, int)
JScrollPane(component, int, int)

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

public class JScrollPaneExample {
    public static final long serialVersionUID = 1L;

    public static void createAndShowGUI() {

final JFrame frame = new JFrame("Scroll pane Example");

        frame.setSize(500,500);
        frame.setVisible(true);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.getContentPane().setLayout(new FlowLayout());

        JTextArea textArea = new JTextArea(20,20);
JScrollPane scrollableTextArea = new JScrollPane(textArea);

    scrollableTextArea.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL
    _SCROLLBAR_ALWAYS);
```

```
scrollableTextArea.setVerticalScrollBarPolicy (JScrollPane.VERTICAL
    _SCROLLBAR_ALWAYS);

frame.getContentPane().add (scrollableTextArea);
}

  public static void main(String[] args) {

    javax.swing.SwingUtilities.invokeLater (new Runnable()
        {

          Public void run() {

            createAndShowGUI ();
        }
    });
  }
}
```

Springpane Layout.

- A springpane Layout arranges the children of its associated container according to a set of (constraints) are noting but horizontal and vertical distance between two component edges. Every constrains are represented by a springLayout. constraint object.

Date
Page

2011 Fall

5a) Write a program to create Menubar. The MenuBar should contain two menus (File and Help). File menu should contain MenuItems "open" and "close". Also handle the event to display a Dialog box with message "Exiting from program" when user click menuItem close.

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

Public class MenuDemo implements ActionListener {

    JFrame f;
    JMenuBar mb;
    JMenu file, help;
    JMenuItem open, close;

MenuDemo() {

    f = new JFrame();
    mb = new JMenuBar();
    file = new JMenu("File");
    help = new JMenu("Help");
    open = new JMenuItem("open");
    close = new JMenuItem("close");

mb.add(file); mb.add(help);
file.add(open); file.add(close);
```

```
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        f.setSize(300,300);
        f.setVisible(true);
        f.setLayout(null);

        f.setJMenuBar(mb);

close.addActionListener(this);
}

    public void actionPerformed(ActionEvent e){

        f.dispose();
JoptionPane.showMessageDialog(f, "Exiting from program");
}

    public static void main(String[] args){

        new MenuDemo();
    }
}
```

2013 spring

3b) Write a program to create frame in swing. The frame should contain three text fields with labels args1, args2 and result. a menu called file with submenus add, subtrac and close.

Ans:-

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

public class MenuDemo extends JFrame implements ActionListener{

    JMenuBar mb;
    JMenu file;
    JMenuItem sum, sub, close;
    JTextField arg1, arg2, result;
    JLabel l1, l2, l3;

public MenuDemo(){

    setSize (400, 500);

  mb= new JMenuBar();
  file = new JMenu ("File");
   mb.add(file);

Sum = new JMenuItem ("Add");
 Sum. addActionListener (this);

sub = new JMenuItem ("subtract");
 Sub.addActionListener (this);

 close = new JMenuItem ("close");
   close.addActionListener (this);

file.add(sum); file.add(sub); file.add(close);
   setJMenuBar (mb);
```

```
l1 = new JLabel ("Arg1");
l2 = new JLabel ("Arg2");
l3 = new JLabel ("Result");

l1.setBounds (100,100,100,30);
l2.setBounds (100,150,100,30);
l3.setBounds (100,200,100,30);

    arg1 = new JTextField();
    arg2 = new JTextField();
    Result = new JTextField();

arg1.setBounds (200,100,120,30);
arg2.setBounds (200,150,120,30);
Result.setBounds (200,200,120,30);

add(l1); add(l2); add(l3);
add(arg1); add(arg2); add(Result);

    setLayout(null);
    setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    setVisible(true);
}

public void actionPerformed (ActionEvent e){
try{
    if(e.getSource()==close){
        System.exit(0);
}
```

```
int n1 = Integer.parseInt (arg1.getText());
int n2 = Integer.paseInt (arg2.getText());

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

    Result.setText (String.valueof (n1+n2));
}

else if (e.getSource() == sub) {
    result.setText (String.valueof (n1-n2));
}

} catch (NumberFormatException ex) {

    Result.setText ("Invalid input");
}
}

    Public static void main (String[] args) {

    new MenuDemo();

}

}
```

4a) WAP to change the font of the text in textField of the Frame. the Frame should contains 3 checkBoxes named bold, italic and plain.

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

Public class B4 extends JFrame implements ActionListener{

    JTextField tf;
JCheckBox bold, italic, plain;

Font defaultFont = new Font("Times New Roman",
                Font.PLAIN, 20);

public B4(){
        setsize (400,300);

    tf= new JTextField("Enter some text");
    tf.setFont( defaultFont );

    bold = new JCheckBox("Bold");
    italic = new JCheckBox ("Italic");
    plain = new JCheckBox ("plain");

    tf.setBounds (100,50,200,30);
    bold.setBounds (100,120,100,30);
    italic.setBounds (100,150,100,30);
    plain.setBounds(100,180,100,30);

    bold.addItemListener (this);
    italic.addItemListener (this);
    plain.addItemListener(this);
```

```
    add(tf);
add(bold); add(italic); add(plain);

    setLayout(null);
    setVisible(true);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

public void itemStateChanged(ItemEvent e){
    if(e.getSource()==bold){

    if(e.getStateChange()==1){

    tf.setFont(new Font("Times New Roman",
        Font.BOLD,20)); }
    else{
        tf.setFont(defaultFont);
    }

}else if(e.getsource()==italic){

    if(e.getStateChange()==1){

    tf.setFont(new Font("Times New Roman",
        Font.ITALIC,200)); }

    else{ tf.setFont(defaultFont); }
```

```
} else {
    tf.setFont(new Font("Times New Roman",
                Font.PLAIN,20));
  }
}

public static void main(String[] args) {
    new B4();
  }
}
```

4B) Write a simple GUI application that use two Dialogs to obtain integers from users and message dialog to show the sum.

```
import javax.swing.*;

class B4{
    public static void main(String[] args){
try{
    int n1 = Integer.paseInt(JOptionPane.showInputDialog
        ("Enter a number"));
int n2 = Integer.paseInt(Joptionpane.showInputDialog
        ("Enter a second number"));
    int sum = n1+n2;
```

```
JOptionPane.ShowMessageDialog(null, "The sum is" +
        String.valueOf(sum));
    System.exit(0);

} catch (NumberFormatException e) {

    JOptionPane.ShowMessageDialog(null, "Invalid number");
        System.exit(0);

    }

}
```

2014 Fall

4a) Write a following program using Frame.
i) it should have three textFields and one Button.
ii) it should accept two numbers.
iii) When user clicks the button, it should calculate sum of two numbers and display the result in third textField.

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

class Test implements ActionListener {
    JFrame f;
    JTextField t1, t2, t3;
    JButton b1;

Test() {
```

```
f= new JFrame("Add two number");

t1= new JTextField();
t1.setBounds(200,50,150,30);
f.add(t1);

t2= new JTextField();
t2.setBounds(200,80,150,30);
  f.add(t2);

t3= new JTextField();
 t3.setBounds(200,110,150,30);
    f.add(t3);


b1= new JButton("sum");
 b1.setBounds(90,200,100,30);

  f.add(b1);

f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  b1.addActionListener(this);
   f.setLayout(null);
   f.setsize(500,500);
   f.setVisible(true);
}
```

```
public void actionPerformed(ActionEvent e){

    int a = Integer.parseInt (t1.getText());
    int b= Integer.parseInt (t2.getText());
     int c=0;

if (e.getSource().equals(b1)){
     c= a+b;
    t3.setText (String.valueOf("Result is : " +c));
}

}

public static void main(String[] args){

      new Test();

}
}
```

4b) Write a program to display "pokahara university" inside the ellipse. Note that the string should be in serif font with size 20 and style bold.

```
import java.applet.*;
import java.awt.*;

Public class Displays extends Applet{

    public void init(){
    setBackground (Color.white);
}
```

Date
Page

```
public void paint(Graphics g)
{
    g.setColor(color.red);
g.drawOval(150,100,370,250);
Font myFont= new Font("serif", Font.BOLD,20);
    g.setFont(myFont);
    g.setColor(color.green);
String s = "pokhara university";
g.drawString(s,240,230);
}
}
/*
<applet code = Displays.class width=500 height=500>
</applet>
*/
```

2014 Spring

4a) What are heavyweight components in Java? Are there any alternatives to those heavyweight components? provide a comparative illustration.

Repeated) java awt is heavy weight component.
java swing and JFX is light weight component in java.

2015 Spring

4a) write a simple program for creating closable frames in the front and centre of your desktop.

- Java JFrame is a class and a type of container which inherits the java.awt.Frame class.

- JFrame works like a main window where components like labels, buttons, text fields are added to create a GUI.

- We can make closable JFrames in diffrent techniques. we have to call the defaultcloseOperation method of a JFrame. i.e. setdefaultcloseOperation(int) supplying the desired value like

i) JFrame.EXIT_ON_CLOSE :- A system.exit(0); call will be executed.

ii) JFrame.DISPOSE_ON_CLOSE :- The frame will be closed and disposed but the application will not exit.

iii) JFrame.DO_NOTHING_ON_CLOSE :- The frame will be closed but not disposed and the application will not exit.

- This techniques are very simple and easy, but it does not allow much flexibility to do any custom operations during the frame closing.

eg :-

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

Public Class ClosableExample extends JFrame{

  public ClosableExample(){

      setSize(500,500);
      setVisible(true);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  // It closes the frame.

setLocationRelativeTo(null); // It centres the position of
                             //JFrame.

toFront(); JFrame at Front position.
}
      public static void main(String[] args){

          new ClosableExample();
}
```

Date 1
Page 1

2016 Fall

4a) Write a program to generate a Frame with two buttons "BLACK" and "BLUE". when a button is clicked backgroundcolor of the frame should change.

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

class Test extends JFrame implements ActionListener {

    JButton black, blue;

  Test() {

    black = new JButton ("BLACK");
      add(black);

    blue = new JButton ("Blue");
      add(blue);

  black.addActionListener(this);
  blue.addActionListener(this);

SetDefaultClose operation (JFrame.EXIT_ON_CLOSE);

    SetLayout (new FlowLayout());
    Setsize (500, 500);
    Setvisible (true);
}
```

```
Public void actionPerformed (ActionEvent e){

    String s = e.getActionCommand();
System.out.println((s + "clicked");

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

    getContentPane().SetBackground(Color.black);
    }

else if (e.getSource() == blue){

    getContentPane().setBackground(Color.blue);
}
}

    Public static void main (String[] args){

        new Test();
    }

}
```

2016 spring

4b) Create a frame with following attributes:

Height = 400
width = 400
Title = MyFrame.

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

public class Example extends JFrame {

    Example() {

        setSize (400,400);
        setLayout (null);
        setVisible (true);
    setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        setTitle ("My Frame");

    }

    public static void main (String[] args) {

        new Example();
    }
}
```

5a) Create a frame with one button and one text field, When user clicks on the button the text entered should be changed to uppercase and color of text field must be changed.

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

```
Public class UppercaseExample extends JFrame
  implements ActionListener {

JTextField tf;
  JButton b;

UppercaseExample() {

    tf = new JTextField(10);
    add(tf);

    b = new JButton("click to change into uppercase");
    add(b);

Set Layout(new FlowLayout());
    Setvisible(true);
    Setsize(800,800);

b.addActionListener(this);
}

  Public void ActionPerformed(ActionEvent e) {
      String lower = tf.getText();
    if(e.getSource() == b) tf.setBackground(Color.red);
    tf.setText(lower.toUpperCase()); }

  public static void main(String[] args) {

        new UppercaseExample();
  }
}
```

2017 (Fall)

4b) Write a java program to create a swing application with 3 buttons. (Repeated)

2017 spring

3b) Create a Frame which has three textfield and one button. When user clicks on button the sum it and display the result in third textField.

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

class Test extends JFrame implements ActionListener{

    JButton jb1;
    JTextField jt1,jt2,jt3;

    Test(){
        jt1 = new JTextField();
        jt1.setBounds(90,50,150,30);
         add(jt1);

        jt2 = new JTextField();
        jt2.setBounds(90,80,150,30);
          add(jt2);

        jt3 = new JTextField();
         jt3.setBounds(90,140,150,30);
           add(jt3);
```

```
jb1 = new JButton("SUM");
jb1.setBounds(90,200,100,30);
  add(jb1);

SetLayout(null);
SetSize(600,400);
 setVisible(true);
}

public void actionperformed(ActionEvent e){

   int a= Integer.parseInt(jt1.getText());
   int b = Integer.paseInt(jt2.getText());

int c=0;

if(e.getSource().equals(jb1)){
    c = a+b;
jt3.setText(String.valueOf("SUM IS:"+" "+c));
 }

}

   public static void main(String[] args){

       new Test();
   }

}
```

2018 Fall Closable Frames in AWT

we can cl
dispose()
method

eg:-

class

add

Date
Page

- We can close the AWT window or Frame by calling dispose() or System.exit() inside the windowClosing() method.

eg:-

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

class closeFrames extends Frame {

    closeFrames() {

addWindowListener(new windowAdapter() {

    Public void windowClosing(windowEvent e) {

        dispose();
    } });

    setsize(400,400);
    setLayout(null);
    setVisible(true);
}

    Public Static void main(String[] args) {

        new closeFrames();
    }
}
```

2019 Fall

4a) create a swing GUI applications that contains a button and two text Fields when a button is clicked the first text field should display "odd number" and second "even number".

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

class GUI extends JFrame implements ActionListener{

    JButton button;
    JTextField t1, t2;

GUI() {

    setSize(500;700);
    setLayout(new FlowLayout());
     t1 = new JTextField(10);
    add(t1);

    t2 = new JTextField(20);
       add(t2);
    t2.setEditable(false);

     button = new Button("check");
    add(button);

     setVisible(true);
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    button.addActionListener(this);
}
```

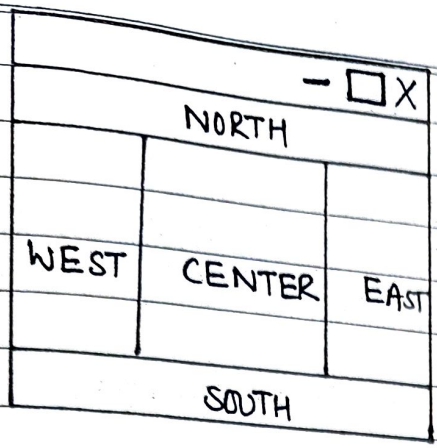
```
public void actionPerformed(ActionEvent e){
try{
int n = Integer.parseInt(t1.getText());
if(n%2==0){
t2.setText("Even number");
}
else{
t2.setText("odd number");
}
}catch(NumberFormatException en){
t2.setText("unable to check");
}
}
}

public class A4{
Public static void main(String[] args){
new GUI();
}
}
```

Date
Page

4b) Create a GUI code to display "pokhara university" in blue color with font Times New Roman, type: Bold and size 20:

```
import java.applet.*;
import java.awt.*;

public class B4 extends Applet {

    public void paint(Graphics g){

    g.setFont(new Font("Times New Roman", Font.BOLD,
                                        20));

     g.setColor(Color.blue);
    g.drawString("pokhara University", 150, 150);
  }
}

  /* <html>
<head> pokhara university </head>

<body> <applet code = "B4.class" width = "400"
                height = "400" >
  </applet> </body>

  </html>
  */
```

3b) WAP to generate the following output in java using BoxLayout.



(Repeated)

4a) create a swing GUI contains a text field (to input radius of a circle) a label and a button. when button is clicked, area of circle should be calculated and displayed in the label.

```
import javax.swing.*;
// import java.awt.*;
import java.awt.event.*;

Public Class A4 extends JFrame implements ActionListener{

Public static final double PI = Math.PI;
    JTextField tf;
    JButton button;
    JLabel label;

public A4() {

    button = new JButton ("calculate");
    tf = new JTextField ();
    label = new JLabel();
```

```
        setSize (400,400);

tf.setBounds (200,50,120,30);
label.setBounds (200,100,120,30);
button.setBounds (200,150,120,30);

add(tf); add(label); add(button);

  button.addActionListener (this);

      setVisible (true);
      setLayout (null);
setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
}

Public void actionperformed (ActionEvent e){
try{

  float radius = Float.parseFloat (tf.getText());
    if (radius <0){
        throw new NumberformatException();
    }
  double area = PI * Math.pow(radius,2);
    label.setText( String.valueOf(area));
}catch (NumberFormatException ex){

      label.setText("Invalid number");
    }
}
```

Public static void main(String[] args){

new A4();

3

3

2020 Fall

36) Create a frame with two textFields, one of which show the Mouse pointer inside or outside Frame and another should x and y cordinates when the user moves the mouse inside the frame.

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

Public Class B3 extends JFrame implements MouseMotionListener,
    MouseListener{

    JTextField tf1,tf2;

Public B3(){
    setSize(400,300);
    tf1 = new JTextField();
    tf2 = new JTextField();

    tf1.setEditable(false);
    tf2.setEditable(false);

setLayout(new FlowLayout());
```

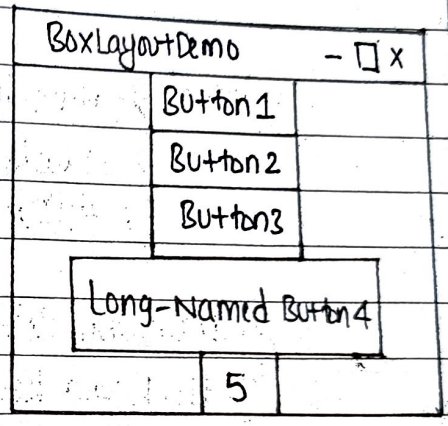
```
add(tf1); add(tf2);

addMouseListener(this);
addMouseMotionListener(this);
setVisible(true);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

Public void maseMoved(MouseEvent e){

    String x= String.valueof(e.getX());
    String y = String.valueOf(e.getY());

    tf2.setText("x:" + x + "y:" + y);
}

Public void mouseDragged(MouseEvent e){
}

public void MouseEntend(MouseEvent e){
    tf1.setText("Mouse is IN!"); }

public void mouseExited(MouseEvent e){

    tf1.setText("Mouse is out!"); }

Public void mouseClicked(MouseEvent e){ }
public void mouseReleased(MouseEvent e){ }
public void mousepressed(MouseEvent e){ }
```

Date / Page /

Public static void main (String[] args) {
new B3();
}
}

Box Layout in Java.

```
import java.awt.*;
import javax.swing.*;
public class B4 extends JFrame {

    JButton b1,b2,b3,b4,b5;

public B4() {

    SetTitle ("BoxLayout Demo");
b1 = new JButton ("Button 1");
b2 = new JButton ("Button 2");
b3 = new JButton ("Button 3");
b4 = new JButton ("Long Named Button 4");
b5 = new JButton ("5");

    Setsize (400,400);

SetLayout (new BoxLayout (getContentPane(), BoxLayout.
                          Y_AXIS));
add(b1); add(b2); add(b3); add(b4); add(b5);
  SetVisible (true);
    pack();
SetDefaultCloseOperation (JFrame.EXIT_ON-CLOSE);
}
    Public static void main (String[] args) {
            new B4(); }
}
```

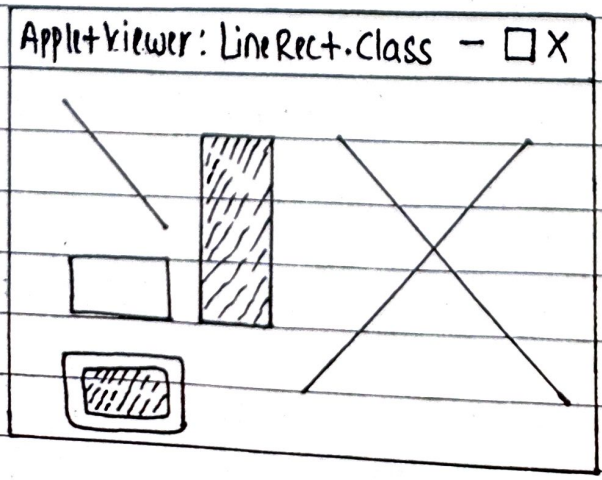


Date
Page

Chapter-6 :- Graphics and Images / Animation / MultiMedia.

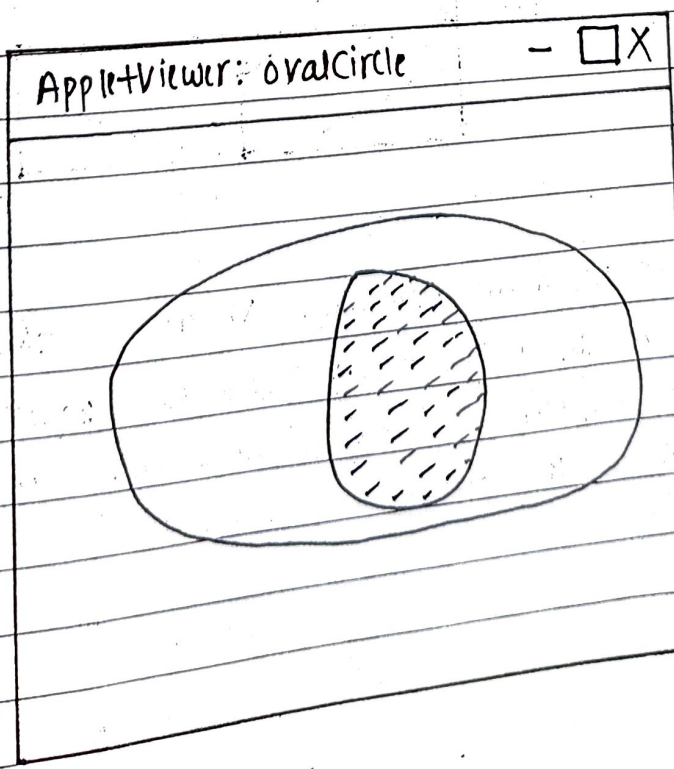
1) LINES AND RECTANGLES

```
import java.awt.*;
import java.applet.*;

Class LineRect extends Applet {

    public void paint (Graphics g) {

        g.drawLine (10, 10, 50, 50);
        g.drawRect (10, 60, 40, 30);
        g.fillRect (60, 10, 30, 80);

g.drawRandRect (10, 100, 80, 50, 10, 10) ;
g.fillRound Rect (20, 110, 60, 30, 5, 5);
 g.drawLine (100, 10, 230, 140);
  g.drawLine (100, 140, 230, 10);
3

3

    <Applet code = LineRect.class width = 250 heigh = 200>
    </Applet>
```



2) Drawing circles and Ellipses.

```
import java.awt.*;
import java.applet.*;

class ovalcircle extends Applet {

  Public void Paint(Graphics g){

    g.drawOval (20,20,200,120);
    g.setColor (color.green);

    g.fillOval (70,30,100,100); // This makes circle.

}

<applet code = ovalcircle.class width = 250 height = 200>
  </applet>
```



Date
Page

3) Drawing Arcs.

- An arcs is a part of oval. In fact, we can think of an oval as a series of arcs that are connected together in an orderly manner.

- The drawArc() designed to draw arcs takes six arguments. The first 4 are the same as the arguments for drawOval() method and the last two represent the starting angle of the arc and the numeber of degrees (sweep angle) around the arc.

- In drawing arcs, java considers three o'clock position as zero degree position (as zero distance) and degrees increase in anti-clockwise direction as shown in fig 1. so, to draw arc from 12:00 clock position to 6:00 o'clock positions, the starting angle would be 90° and the sweep angle would be 180°.

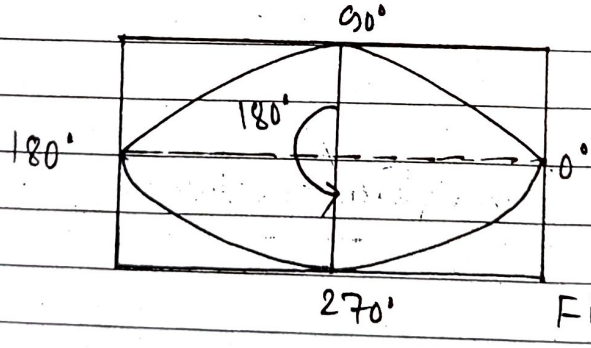


Fig 1 :- Arc as a part of oval.

- We can also draw also draw an arc in the backward direction by specifying the sweep angle as negative. For eg:- if the last arguments is -135° and the starting angle is 45°, then the arc drawn is shown in Fig 2:-

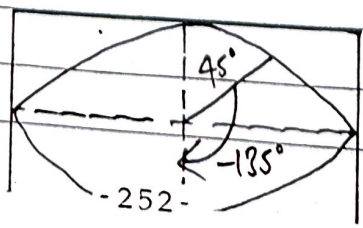


Fig:-2 Drawing arc in Anticlockwise direction.

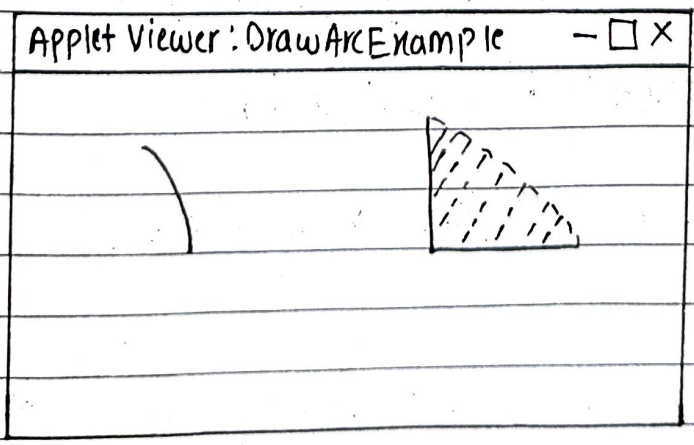
Date
Page

We can use the fillArc() method to fill the arc. Filled arc are drawn as if they were sections of a pie.

eg:-

```
/*
<applet code = "DrawArcExample.class" width = 500 height = 500>
</applet>
*/

import java.applet.*;
import java.awt.*;

Public class DrawArcExample entends Applet {

Public void paint (Graphics g) {
  setForeground (color.red);

  g.drawArc (10,10,50,100, 10,45);
  g.fillArc (100,10,100,100, 0,90);
}
}
```

output:-



Date
Page

4) Drawing polygons in Applets.

- Polygons are shape with many sides. A polygon may be considered a set of lines connected together.

- We drawpolygon using drawplygon() method of Graphic class.
  This method takes three arguments.

i) An array of integers containing x coordinates.
ii) An array of integers containing y coordinates.
iii) An integer for the total number of points.

- It is obvious that x and y array should be of same size and we must repeat the 1st point at the end of the array for closing the polygon. The polygon shown in fig 1 can be drawn by using the drawpolygon() method as follows

```
Public void paint ( Graphics g)
{
    int xpoints [] = { 10, 170, 80, 10 };
    int ypoints [] = { 20, 40, 140, 20 };
    int npoints [] = xpoints.length;
    g.drawpolygon ( xpoints, ypoints, npoints);
}
```

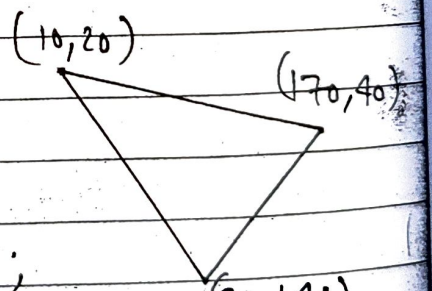
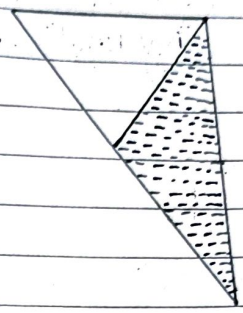


Fig 1:- Polygon with 3 sides.

- We can also draw a filled polygon will fillpolygon() method

Eg:-

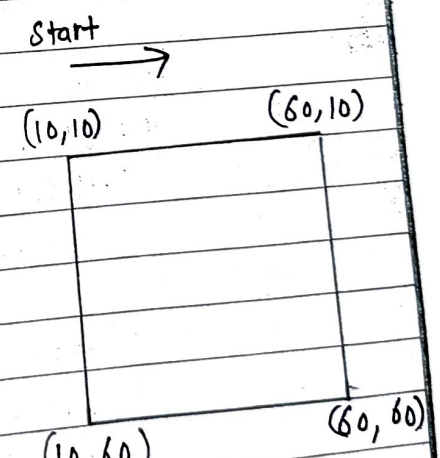
```
import java.awt.*;
import java.applet.*;
public class poly extends Applet{

int x1[] = { 20,120,220,20};
int y1[] = { 20,120,20,20};
 int n1=4;

int x2[] = {120,220,220,120} ;
int y2[] = {120,20,220,120} ;
 int n2 = 4;

Public void paint (Graphics g){
    g.drawPolygon (x1,y1,n1);
    g.fillPolygon (x2,y2,n2);
}
}
```



output

Making square using drawpolygon

```
import java.awt.*;
import java.applet.*;
Public class Drawsquare extends Applet{

Public void paint (Graphics g){

    int[] a = { 10,60,60,10};
    int[] b = { 10,10,60,60};
g.drawPolygon (a,b,4);
    }
}
```



Date |
Page |

2011 Fall

4b) Write a program to draw a 2D rectangle in green color. Next draw the flag of Nepal inside the rectangle. Write a string "My Nepal" below the rectangle.

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

Public Class NepaliFlag extends Jpanel{

    Public void paint(Graphics g){

        g.setColor(color.green);
        g.fillRect(10,20,300,300);

int x[]={20,200,100,240,20,20};
int y[]={20,150,150,300,300,20};
    g.setColor(color.blue);
    g.fillPolygon(x,y,6);

int xx[]={30,170,75,215,30,30};
int yy[]={40,140,140,290,290,35};
    g.setColor(color.red);
     g.fillpolygon(xx,yy,6);

    g.setColor(color.white);
    g.fillArc(50,80,40,40,0,-180);
    g.fillOval(50,200,50,50);
Font f1=new Font("Times New Roman",Font.BOLD,30);
```

Date
Page

```
        g.setFont(f1);
        g.setColor(color.black);
        g.drawString("My Nepal", 10, 350);
    }

    public static void main(String[] args)
    {

        JFrame f = new JFrame("Nepal Flag");
        f.setSize(500,500);

f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        NepaliFlag nepal = new NepaliFlag();
        f.add(nepal);
        f.setLocationRelativeTo(null);
        f.setVisible(true);
    }
}
```

2014 Fall

4b) Write a program to draw a Barchar of Number of students giving enam for java, c and C++

```
import java.awt.*;
import java.applet.*;

Public class Barchart extends Applet
{
```

```
int n=0;
String label[];
int value[];

public void init() {

  setBackground(Color.PINK);
try{

int n = Integer.parseInt(getParameter("columns"));

  label = new String[n];
  value = new int[n];

label[0] = getParameter("label1");
label[1] = getParameter("label2");
label[2] = getParameter("label3");

value[0] = Integer.parseInt(getParameter("c1"));
value[1] = Integer.parseInt(getParameter("c2"));
value[2] = Integer.parseInt(getParameter("c3"));

} catch(NumberFormatException e) {
      e.printStackTrace();
}

  public void paint(Graphics g) {

    for(int i=0; i<3; i++) {
  g.setColor(Color.black);
  g.drawString(label[i], 20, i*50+30);
  g.setColor(Color.red);
```

```
g.fillRect (50, i*50+10, value[i], 40);
}
}
}

/*
<applet code = Barchart width = 400 height = 400>

<param name = c1 value = 110>
<param name = c2 value = 150>
<param name = c3 value = 100>

<param name = label1 value = java>
<param name = label2 value = C>
<param name = label3 value = C++>

<param name = columns valve = 3>

</applet>
*/
```

2015 spring

4b) Write a program to draw a Bar chart of the total number of students appearing in an examination in java. Also provide the Barchart for the number of male and female.

```
import java.awt.*;
import java.applet.*;

public class Barchar extends Applet {
```

```
int n=0;
String label[];
int value[];

Public void init(){
  SetBackground(color.pink);
  try{

  int n = Integer.parseInt(getparameter("columns"));

    label = new String[n];
    value = new int[n];

label[0] = getparameter("label1");
label[1] = getparameter("label2");
label[2] = getparameter("label3");

value[0] = Integer.parseInt(getParameter("c1"));
value[1] = Integer.paneInt(getParameter("c2"));
value[2] = Integer.paneInt(getparameter("c3"));

} catch(NumberFormatException e){ }

    public void paint(Graphics g){

    for(int i=0; i<3; i++){
      g.setcolor(color.black);
      g.drawString[label[i], 20, i*50+30);

      g.setColor(color.red);
```

```
            g.fillRect(50, i*50+10, value[i], 40);
        }
    }
}

/*

<applet code=Barchart width=400 height=400>

<param name=c1 value=400>
<param name=c2 value=150>
<param name=c3 value=250>

<param name=label1 value=Total students>

<param name=label2 value=male>

<param name=label3 value=female>

<param name=columns value=3>

</applet>

*/
```

2017 Fall 5b)

What is GUI programming? Write a program to draw Nepali Flag using graphics.

Date |
Page |

A GUI programming means a Graphical user interface (GUI) allows to intract with the computer program using a pointing device that manipulates small pictures on a computer screen. The style of programming is called "event driven programming". The GUI programs are event-driven programs.

⇒ Best Nepali Flag.

```
import java.awt.*;
import java.applet.*;

/*
<applet code="Flag" width="500" height="500">
</applet>
*/
Public class Flag extends Applet{

Static color NEPALIFLAG =new color(221, 12, 39);

public void paint(Graphics g){

// For upper triangle

int upperTrianglex[] = {210, 360, 210};
int upperTriangley[] = {10, 160, 160};

Polygon upperTriangle = new polygon(upperTrianglex, upperTriangley, 3);

g.drawpolygon(upperTriangle);
g.setColor(NEPALIFLAG);
g.fillpolygon(upperTriangle);
```

Date 1
Page 1

```
// For lower triangle

int lowerTrianglex[] = {210, 390, 210};
int lowerTriangley[] = {160, 340, 340};

Polygon lowerTriangle = new Polygon (lowerTrianglex, lowerTriangley,
                                     3);
    g.drawPolygon (lowerTriangle);
    g.setColor (NEPALI FLAG);
    g.fillPolygon (lowerTriangle);

// Stick of the flag

int stickx[] = {207, 210, 210, 207};
int sticky[] = {7, 7, 500, 498};

Polygon stick = new polygon (stickx, sticky, 4);

    g.drawPolygon (stick);
    g.setcolor (color.black);
    g.fillpolygon (stick);

// For Borders.
int borderx[] = {210, 210, 360, 210, 390, 210, 210, 396, 216, 363};
int bordery[] = {7, 10, 160, 160, 340, 340, 343, 343, 163, 163};
Polygon border = new polygon (borderx, bordery, 10);
    g.drawpolygon (border);
    g.setcolor (color.blue);
// flag border using fill polygon

    g.fillpolygon (border);
```

Date
Page

```
// For lower triangle

int lowerTrianglex[] = {210, 390, 210};
int lowerTriangley[] = {160, 340, 340};

Polygon lowerTriangle = new Polygon (lowerTrianglex, lowerTriangley,
                                     3);
    g.drawPolygon (lowerTriangle);
    g.setColor (NEPALI FLAG);
    g.fillPolygon (lowerTriangle);

// Stick of the flag

int stickx[] = {207, 210, 210, 207};
int sticky[] = {7, 7, 500, 498};

Polygon stick = new Polygon (stickx, sticky, 4);

    g.drawPolygon (stick);
    g.setcolor (color.black);
    g.fillPolygon (stick);

// For Borders.
int borderx[] = {210, 210, 360, 210, 390, 210, 210, 396, 216, 363};
int borderY[] = {7, 10, 160, 160, 340, 340, 343, 343, 163, 163};
Polygon border = new polygon (borderx, bordery, 10);
    g.drawpolygon (border);
    g.setcolor (color.blue);
// flag border using fill polygon

    g.fillpolygon (border);
```

```
// for moon

    g.setColor(color.white);
    g.fillOval(240, 100, 30, 30);
    g.setColor(NEPALIFLAG);
      g.fillOval(240, 101, 30, 30);


// for star

int starX[] = {240, 250, 255, 260, 270, 260, 255, 250, 240, 245};
int starY[] = {260, 260, 250, 260, 260, 270, 280, 280, 290, 280, 280, 270};

polygon star = new polygon(starX, starY, 12);

// flag.drawPolygon is used.

g.drawPolygon(star);
  g.setColor(color.white);
    g.fillPolygon(star);
}

}
```

2020 Fall

4b) As per a recent survey, among all the javascript framework 40% prefer React while Angular comes second with 30% developers preferring it and 20% prefer VUE while 10% use other frame work. Create a piechart to show information of the survery.

Date
Page

```
import java.applet.*;
import java.awt.*;

Public class piechart extends Applet {

  int [] data_values;
  color[] data_clr;
   int total;

  public void init() {

data_values = new int[]{ 40,30,20,10};
    data_clr= new Color[] { Color.red, Color.blue, Color.green,
                            Color.yellow};
}

     public void start(){
     int n = data_values.length;
        int i;
        total = 0;
     for( i=0; i<n ; i++) {
          total += data_values[i];
        }
}

     public void paint (Graphics g){

          int i;
          int start_angle =0;

for( i=0; i< data_values.length ; i++) {
```

```
int arc_angle =(int) (data_values[i] * 360 / total);
g.drawArc(100,100,300,300, start_angle, arc_angle);
    g.setColor (data_clr[i]);
g.fillArc(100,100,300,300, start_angle, arc_angle);
    start_angle += arc_angle;
}
}
}

/*
<applet
    code = "piechart.class"
    width = "500"
    height = "500">
</applet>
*/
```

Short notes

1) Graphics object

- A Graphics object is an object that encapsulates state information needed for the basic rendering operations that Java supports.

- State information includes the following properties :-

i) The component object on which to draw.

ii) A translation origin for rendering and clipping coordinates.

iii) The current clip.

iv) The current colour.

v) The current font.

vi) The current logical pixel operation function.

vii) The current XOR alternation colour.

- Declaration of Graphics (GUI) class are :-
  public abstract class Graphics extends object

- All coordinates that appears as arguments to the methods of this Graphics object are considered relative to the translation origin of this Graphics object prior to the invocation of the method.

Date
Page

- All rendering operations modify only pixels which lie within the area bounded by the current clip, which is specified by a shape in user space and is controlled by the program using the Graphics object.

Repaint Method in JAVA

- The Repaint method in java is avilable in java.applet.Applet class which is a final method used whenever we want to call update method along with the call to paint Method.

- The repaint() method has four forms.
  i) void repaint()
  ii) void repaint (int left, int top, int width, int height)
  iii) void repaint (long maxDelay)
  iv) void repaint(long maxDelay, int x, int y, int width, int height)

- repaint() → update() → paint()

- Repaint() Method
  - use when a window needs to be updated.
  - Calls the paint() method.
  - Creates a graphics objects.

SYNTAX:-

```
// Class extending Applet
Public class <classname> extends Applet
public method <methodName> (<arguments>)
    repaint(); // Calling repaint method when required.
}
}
```

Date
Page

- Repaint method() can't be overridden.

eg:- 1

```
import java.awt.*;
import java.applet.*;

// public class RepaintExample implements .

class RepaintExample extends Applet {
        int i;
    public void paint (Graphics g)
    {
        g.drawString ("i=" + i, 100, 100);

    try {
        Thread.sleep (1000);
    } catch (InterruptedException e) {}

        i++;
        repaint ();
    }
}
```

eg 2:-

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

public class RepaintExample extends Applet implements ItemListener{
    Choice c = null;
```

```
public void init(){

    c = new Choice();

    c.add("Apple");
    c.add("Mango");
    c.add("Grapes");
  add(c);

  c.addItemListener(this);

}

public void paint(Graphics g)
{

g.drawString(c.getSelectedItem()+"Selected Item", 20,
                    30);
}
@Override
public void itemStateChanged(ItemEvent e){

    repaint();
}

}
```