OOS Assignment -4.

Q1). Difference between Java AWT and Swing in terms of their features and capabilities. Explain how you can use Layout Manager.

⇒ **Java AWT:-** The graphical programming in java is supported by the AWT package. The AWT stands for Abstract Window Toolkit. We need to import java.awt package for using these components. AWT contains large number of classes which help to include various graphical components in java program. These graphical components includes text box, buttons, labels, radio buttons, list items and so on.

Java Swing:- Swing is a GUI widget toolkit for Java. It is a part of Oracle's Java Foundation classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs. Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT).

Difference between Java AWT and Swing:

AWT	Swing
1). The Abstract window toolkit is a heavy weigh component,	The swing is a light weight component, because it's

	because every graphical unit invoke the native methods	the responsibility of JVM to invoke the native methods.
3)	The look and feel of AWT depends upon platform	As swing is based on model view controller pattern, the look and feel of swing components is independent of hardware and the operating system.
3)	AWT occupies more memory space.	Swing occupies less memory space.
4).	AWT is less powerful than swing.	Swing is extension of AWT and many drawbacks of AWT are removed in Swing.

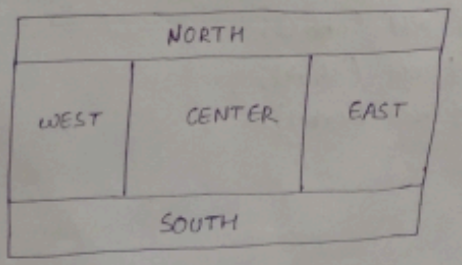
⇒ Layout Manager:- The layout managers are used to arrange components in a particular manner. The java layout manager facilitate us to control the positioning and size of the components in GUI forms. Java packages that provide the suitable classes for layout management are "import java.awt.*" and "import javax.swing.*".

Border layout is used to arrange the components in five regions: north, south, east, west and center, this is the default layout of a frame or window.

⇒ Example to show layout management using Border layout class.

```
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();
    }
}
```

O/P:-



(Q2). Write an AWT application with checkbox such that all cable TV channels will be displayed from the selected category.

⇒
```
import java.awt.*;
import java.awt.event.*;
public class myClass {
    myClass() {
        Frame f = new Frame ("CheckBox Example");
        final Label label = new Label();
        label.setAlignment (Label.CENTER);
        label.setSize (400, 100);
        Checkbox chb1 = new Checkbox ("Entertainment");
        chb1.setBounds (100, 100, 50, 50);
        Checkbox chb2 = new Checkbox ("News");
        chb2.setBounds (100, 150, 50, 50);
        Checkbox chb3 = new Checkbox ("Sports");
        chb3.setBounds (100, 200, 50, 50);
        Checkbox chb4 = new Checkbox ("Movies");
        chb4.setBounds (100, 250, 50, 50);
        f.add (chb1);
        f.add (chb2);
        f.add (chb3);
        f.add (chb4);
        f.add (label);
        chb1.addItemListener (new ItemListener() {
            public void itemStateChanged (ItemEvent e) {
                label.setText ("Entertainment: " + "\n" +
                (e.getStateChange() == 1 ? "SONY \n &TV": ""));
            }
        });
        chb2.addItemListener (new ItemListener() {
            public void itemStateChanged (ItemEvent e) {
                label.setText ("News: " + "\n" +
                (e.getStateChange() == 1 ? "India Today \n
                    ABP news": ""));
            }
        });
        chb3.addItemListener (new ItemListener() {
            public void itemStateChanged (ItemEvent e) {
                label.setText ("Sports: \n" +
                (e.getStateChange() == 1 ? "Star Sports \n
                    TEN Sports": ""));
            }
        });
        chb4.addItemListener (new ItemListener() {
            public void itemStateChanged (ItemEvent e) {
                label.setText ("Movies: \n" +
                (e.getStateChange() == 1 ? "SONY Max \n STAR
                    gold": ""));
            }
        });
```

```
        f. setSize (400, 400);
        f. setLayout (null);
        f. setVisible (true);
    }
    public static void main (String args[]) {
            new myClass();
    }
}
```

Q3). What are Java Beans ? What are the advantages of using Java Beans ? Illustrate the use of Java Beans with a simple example.

Java Beans are classes that encapsulates many objects into a single object (the bean). It is a java class that should follow following conventions:

1). Must implement Serializable.
2). It should have a public no-arg constructor.
3). All properties in java bean must be private with public getters and setters methods.

Java provide the facility of creating some user defined components using Java bean technology.

⇒ Advantages of Java Beans:

1). The java beans posses the property of "write once and run anywhere". This means that once the beans gets developed by the developer on one machine then it can be used on any other platform. The developer need not have to put extra efforts of writing different version of their software for each platform or operating system on which the target bean has to be deployed.

2). Beans can work in different local platforms. And hence they can be used globally over any platform.

3). Beans have the capability of capturing the events sent by other objects; similarly the beans can sent the events to other objects. That means bean can also be used in object communication.

4). The properties, events and methods of the beans can be controlled by the application developer. For example we can add some new properties to an existing bean.

⇒ Example to illustrate Java Beans:

```
// Java program of Java Bean class.
package Bean;
public class Student implements java.io.Serializable {
    private int id;
    private String name;
    public Student() { }
    public void setId (int id) { this.id = id;}
    public int getId () { return id;}
    public void setName (String s) { name = s;}
    public String getName() { return name;}
}
```

// Java program to access Java Bean

```
package Bean;
public class Test {
    public static void main (String args[]) {
        Student s = new Student();
        s.setName ("Swapnil");
        System.out.println (s.getName());
    }
}
```

Q14). What are Cookies? How they are useful? Create JSP code that uses a persistant cookie (i.e. a cookie with an expiration date in future) to keep track of how many times the client computer has visited the page. Use setMaxAge method to remain on the client's computer for one month. Display the number of pages hits every time the page loads.

⇒ Cookies: Cookies are text files with small pieces of data - like username and password - that are used to identify your computer as you use a computer network. Specific cookies known as HTTP cookies are used to identify specific users and improve your web browsing experience. Data stored in a cookie is created by the server upon your connection. This data is labeled with an ID unique to you and your computer. When the cookie is exchanged between your computer and the network server, the server reads the ID and knows what information to specifically serve to you.

⇒ Uses :-

1). Session Management → Cookie let websites recognize users and recall their individual login information and preferences, such as sports news versus politics.

2). Personalization → Customized advertising is the main way cookies are used to personalize your sessions. You may view certain items or parts of a site, and cookies use this data to help build targeted ads that you might enjoy.

3). Tracking → Shopping sites use cookies to track items users previously viewed, allowing the sites to suggest other goods they might like and keep items in shopping carts while they continue shopping.

⇒ newhtml.html

```
<html>
<head>
<title></title>
<meta http-equiv = "Content-Type" content = "text/html;
                                    charset = UTF-8">
</head>
<body>
<form action = "MainCookie.jsp" method = "GET">
    FIRST NAME: <input type = "text" name = "firstname">
    LAST NAME: <input type = "text" name = "lastname">
    <input type = "submit" value = "submit" />
</form>
</body>
</html>
```

Main Cookie.jsp

```
<%@ page contentType = "text/html" pageEncoding = "UTF-8" %>
<%@ page language = "java" import = "java.util.*" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv = "Content Type" content = "text/html;
                                    charset = UTF-8">

</head>
<body>
<h1> Setting Cookie values </h1>
   First Name : <% = request.getParameter ("first_name") %>
   Last Name : <% = request.getParameter ("last_name") %>
   <%
     Cookie firstName = new Cookie ("first_name", request.
                        getParameter ("first_name"));

     Cookie LastName = new Cookie ("last_name", request.
                       getParameter ("last_name"));

     firstName.setMaxAge (60 * 60 * 24 * 30);
     lastName.setMaxAge (60 * 60 * 24 * 30);
     response.addCookie (firstName);
     response.addCookie (lastName);
   %>
<a href = "ReadCookies.jsp"> Next page to view the
                             cookies values </a>

</body>
</html>
```

ReadCookies.jsp:

```
<%@ page contentType = "text/html" pageEncoding =
                                    "UTF-8" %>

<!DOCTYPE html>
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html";
                                    charset = UTF-8">

<title> JSP Page </title>
</head>
<body>
<h1> Reading Cookies </h1>
<%
   Cookie cookie = null;
   Cookie cookies[] = null;
   cookies = request.getCookies();
   if (cookies != null) {
      System.out.println ("<h2> Found cookies name and
                           value </h2>");

      for (int i=0; i< cookies.length; i++)
      {
         cookie = cookies[i];
         System.out.print ("Name : " + cookie.getName() +
                           ",");
         System.out.print ("value : " + cookie.getValue() +
                           "<br>");
      }
   } else { System.out.println ("No Cookie found");}
%>
</body>
</html>
```

Q5) Discuss the advantages / disadvantages of java servlets over other available technologies for the similar purpose. Write a java servlet that accepts user preferences (colour, body etc.) from user, saves it as a cookie on user machine and reads the cookie from the user machine.

⇒ There are many advantages of servlet over CGI (common gateway interface) (CGI is a scripting language common as a server-side programming language). The web container creates threads for handling the multiple requests to the servlet. Thread have many benefits over the processes such as they share a common memory area, lightweight, cost of communication between the threads are low. The advantages of Servlets are as follows:

1). Better Performance because it creates a thread for each request, not process.

2). Portability because it uses Java language.

3). Robust JVM manages servlets, so we don't need to worry about the memory leak, garbage collection etc.

4). Secure because it uses java language.

⇒ Java Servlet :

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
import javax.servlet.http.*;

public class store extends HttpServlet {
    protected void doPost (HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
                                             IOException {

        response.setContentType ("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            String s1 = request.getParameter ("colour");
            if (s1.equals("RED") || s1.equals("BLUE") ||
                                    s1.equals("GREEN"))
            {
                Cookie ck1 = new Cookie("colour", s1);
                response.addCookie (ck1);
                out.println ("<html>");
                out.println ("<body>");
                out.println ("You selected : " +s1);
                out.println ("<form action='retrieve' method
                                    ='post'>");

                out.println ("<input type='submit' value
                                    ='submit'/>");

                out.println ("</form> \n </body>\n
                                    </html>");

            }
        } finally {
            out.close();
        }
    }
}
```

⇒ retrieve:

```
import java.io.*;
import javax.servlet.*; import javax.servlet.http.*;
public class retrieve extends HttpServlet{
    protected void doPost (HttpServletRequest request,
                           HttpServletResponse response)
                    throws ServletException, IOException
    {
        response.setContentType("text/html; charset=
                                UTF-8");
        PrintWriter out = response.getWriter();
        try {
            Cookie ck[] = request.getCookies();
            out.println("<html> \n <head> \n
                        <title> servlet </title> \n
                        </head>");

            out.println("<body bgcolor=" + ck[0].getValue()
                        + ">");
            out.println("You selected color is:" +
                        ck[0].getValue() + "</ht>");

            out.println("</body> \n </html>");
        } finally {
            out.close();
        }
    }
}
```

Q6). What are two different ways of including a file in the JSP main documents? Illustrate the advantages of each using suitable example.

⇒ 1). Using include directive:

The include directive is used to include a file during the translation phase. This directive tells the container to merge the content of other external files with the current JSP during the translation phase.

General Usage:

```
<%@ include file="relative url">
```

We can use include directives anywhere in the JSP page:

Example:-

Header.jsp:

```
<%! int pageCount = 0;
    void addCount() { pageCount++; }
%>
<% addCount(); %>
<html>
<head>
<title> Include Directive </title>
</head>
<body>
<center>
  <h2> </h2>
  <p> </p>
</centre>
```

footer.jsp:

```
<center> <p></p> </center>
</body>
</html>
```

main.jsp:

```
<%@ include file = "header.jsp" %>
   <p> </p>
<%@ include file = "footer.jsp" %>.
```

Using JSP : include action tag

jsp: include action tag is used to include the content of another resources it may be jsp, html or servlet.

Advantage: Code reusability - We can use a page many times such as including header and footer pages in all pages. So it saves a lot of time.

General Usage:

```
<jsp: include page = "relativeURL | <%= expression %>" />
```

Example:-

index.jsp:-

```
<h2> this is index page </h2>
<jsp: include page = "printdate.jsp" />
<h2> end section of index page </h2>.
```

printdate.jsp simply prints the date.

Q7) Write a program in java to show the mouse click event. The program should change the background color of window randomly at each mouse click.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
< applet code = "Button1" width = 350 height = 200>
</ applet>
*/
public class Button1 extends Applet implements ActionListener
{
    Button button = new Button ("change the color");
    boolean flag = true;
    public void init () {
            add (button);
            button.addActionListener (this);
    }
    public void paint (Graphics g) {
            if (flag)
                setBackground (Color.yellow);
            else
                setBackground (Color.red);
    }
    public void actionPerformed (ActionEvent e) {
            String str = e.getActionCommand ();
            if (str.equals ("change the color")) {
                flag = !flag;
                repaint ();
            }
        }
}
```

Q8). Write a program of threads in java showing inter leaving of action from two threads : t1 & t2 synchronizing on a shared object. Let t1 print message Ping and t2 prints message Pong. Take as command line arguments the following inputs to the program :

Sleep interval for thread t1.
Sleep interval for thread t2.
Messages per cycle.
No. of cycles.

⇒
```
import java.util.*;
public class Main {
    static Thread t1, t2;
    static int sleep1, sleep2, msgPerCycle, cycle;
    public static void main (String args[]) {
        Scanner sc = new Scanner (System.in);
        sleep1 = sc.nextInt();
        sleep2 = sc.nextInt();
        msgPerCycle = sc.nextInt();
        cycle = sc.nextInt();
        msgPerCycle *= 2;
        int temp = msgPerCycle;
        int q = 1;
        while ( cycle-- > 0) {
            System.out.println ("Cycle : " + (q++));
            t1 = new Thread ( new test1());
            t2 = new Thread ( new test2());
            t1.start();
            t2.start();
            try { t1.join(); }
            catch ( Exception e) { System.out.println(e); }
            try { t2.join(); }
            catch ( Exception e) { System.out.println(e); }
            msgPerCycle = temp;
        }
    }
}
class test1 implements Runnable {
    public void run() {
        while ( e.msgPerCycle > 1) {
            while ( e.msgPerCycle % 2 = 1) {
                try {
                    Thread.sleep (e.sleep1);
                } catch (Exception e) {
                    System.out.println(e);
                }
            }
            System.out.print ("Ping");
            e.msgPerCycle--;
        }
    }
}
class test2 implements Runnable {
    public void run () {
        while (e.msgPerCycle > 0) {
            while (e.msgPerCycle % 2 == 0) {
                try {
                    Thread.sleep (e.sleep2);
                } catch (Exception e) {
```

```
        System.out.println(e);
      }
    }
    System.out.println("Pong");
    e.msgPerCycle--;
  }
}
```

Input:- sleep 1 → 100
sleep 2 → 100
msgPerCycle → 1
cycle → 3.

Output:-
```
Cycle : 1
Ping
Pong
Cycle : 2
Ping
Pong
Cycle : 3
Ping
Pong
```

Q8). Write a java applet program using swing methods and classes to find the factorial of a number, which is accepted from keyboard. Use the recursive method for factorial. Also, write the HTML code for displaying the applet.

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

public class myClass extends Applet implements KeyListener
{
    String s = "";
    int num = 0;
    public void init() {
        addKeyListener(this);
        requestFocus();
    }
    public void KeyPressed(KeyEvent k)
    {
        showStatus("Key pressed");
    }
    public void keyReleased(KeyEvent k)
    {
        showStatus("Key Released");
    }
    public void keyTyped(KeyEvent k)
    {
        s = k.getKeyChar();
        num = factorial(Integer.parseInt(s));
        repaint();
    }
    public void paint(Graphics g)
    {
        g.drawString(String.valueOf(num), 30, 70);
    }
    public int factorial(int num)
    {
        if(num == 1) return 1;
        return num * factorial(num-1);
    }
    public static void main(String args[]){
        new myClass();
    }
}
```

HTML code:

```
<html>
<head>
<title> Keyboard Event </title>
</head>
<body>
<applet code = "myClass" width = 500 height
                                    = 300 >
</applet>
</body>
</html>.
```

Q10). Describe the use of an applet in Java. Develop an applet that asks for date of birth of a user in the format (DD/MM/YYYY) and displays the sum of all digits as "Lucky Number" of the user. Write a HTML page that embeds this applet.

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class myClass extends Applet implements KeyListener
{
    String msg = "";
    public void init()
    {
        TextArea ta = new TextArea ("", 10, 80,
                                    TextArea.SCROLLBARS_BOTH);
        ta.setEditable (true);
        ta.addKeyListener (this);
        add ("Center", ta);
    public void KeyPressed (KeyEvent k)
    {
        showStatus ("Key pressed");
    }
    public void KeyReleased (KeyEvent k)
    {
        showStatus ("Key released");
    }
    public void keyTyped (KeyEvent k)
    {
        msg = k.getKeyChar();
        String s[] = msg.trim().split("/n");
        int num = 0;
        for (int i=0; i< s.length; i++) {
            num += Integer.parseInt (s[i]);
        }
        msg = String.valueOf (num);
        repaint();
    }
    public void paint (Graphics g)
    {
        g.drawString (msg, 30, 70);
    }
    public static void main (String args[])
    {
        new myClass();
    }
}
```

HTML Code :-

```
<html>
<head>
<title> </title>
</head>
<body>
<applet code = "myClass" width = 500 height = 300>
</applet>
</body>
</head>
</html>.
```

⇒ Uses of Applet in Java :

Applet are small java programs that can be transferred over the internet from one computer to another and can be displayed on various web browsers. Various applets perform arithmatic operations, display graphics, play sounds, create animation and so on.

Q11). What is a thread ? How do you threads behave in java ? Write a program in java showing the action from three threads using a suitable example. Also show how synchronization among threads is achieved ?

⇒ A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently. Every thread has a priority.

```
⇒ public class myClass extends .
{
    public static void main ( String args[])
    {
        Thread th1 = new Thread ( new myClass1());
        Thread th2 = new Thread (new myClass2());
        Thread th3 = new Thread (new myclass3());
        td1. start();
        td2. start();
        td3. start();
    }
}
class myClass1 implements Runnable {
    public void run() {
        System. out. println (" class 1");
    }
}
class myClass2 implements Runnable {
    public void run(){
        System. out. println (" class 2");
    }
}
class myClass3 implements Runnable {
    public void run(){
        System. out. println (" class 3");
    }
}
```

Output :-

```
class 1          or      class 1
class 3                  class 2
class 2                  class 3
```

⇒ Thread Synchronization is a process of allowing only one thread to use the object when multiple threads are trying to use the particular object at the same time. To achieve this thread synchronization we have to use a java keyword or modifier called "synchronized".

Q12) Explain the mechanism of introspection during the design process of an application using Java Beans. What is the role of design patterns (Naming patterns) for properties and event in it?

⇒ Introspection can be defined as the technique of obtaining information about bean properties, events and methods. Basically introspection means analysis of bean capabilities.

Suppose - we have downloaded a set of beans from the internet, installed it on our PC and then if we want to configure that set of beans with some working system then we require a builder tool. The job of builder tool is to deploy the bean components within the already existing working system. But this builder tool needs to present the information about the bean component to software designer. Hence builder tool uses a technique called introspection in order to collect information about beans' properties, events and methods. The introspection technique is always needed by Java beans.

⇒ Design patterns for properties:

A property is a subset of a Bean's state. The values assigned to the properties determine the behaviour and appearence of that component. A property is set through a setter method. A property is obtained by a getter method. There are two types of properties: simple and indexed.

(i) Simple Properties:

A single property has a single value. It can be identified by the following design patterns, where N is the name of the property and T is its type:

public T getN() public void setN(T arg)

A read/write property has both of these methods to access its values. A read-only property has only a get method. A write-only property has only a set method.

Here is read/write simple properties along with their getter and setter methods:

⇒ private double a;
public double getA() {return a;}
public double setA(double a) {
this.a = a;
}

(ii) Indexed properties:

An indexed property consists of multiple values. It can be identified by the following design patterns, where N is the name of property and T is its type:

```
public T getN(int index);
public void setN(int index, T value);
public T[] getN();
```

Here is an indexed property called data along with its getter and setter methods:

```
private double data[];
public double getData(int index)
{
    return data[index];
}
public void setData(int index, double value)
{
    data[index] = value;
}
```

⇒ Design patterns for Events:

Beans use the delegation event model ~~that was~~. Beans can generate events and send them to other objects. These can be identified by the following design patterns, where T is the type of the event:

```
public void addTListener(TListener eventListener)
    public void addTListener(TListener eventListener)
        throws java.util.TooManyListenersException
public void removeTListener(TListener eventListener)
```

These methods are used to add or remove a listener for the specified event. The version of addTListener() that does not throw an exception can be used to multicast an event, which means that more than one listener can register for the event notification. The version that throws TooManyListenersException unicasts the event, which means that the number of listeners can be restricted to one. In either case, removeTListener() is used to remove the listener. For example, assuming an event interface type called TemperatureListener, a Bean that monitors temperature might supply the following methods:

```
public void addTemperatureListener(TemperatureListener
                                   tl)
{ ---- }

public void removeTemperatureListener(TemperatureListener
                                      tl)
{ --- }
```

⇒ Design patterns are not used for naming nonproperty methods. The introspection mechanism finds all of the public methods of a Bean. Protected and private methods are not presented.