# II - ASSIGNMENT

(Start Writing From Here)

① Demonstrate the usage of inter-thread communication in java with a suitable example

Inter-thread communication or co-operation:- It is all about allowing synchronized threads to communicate with each other

→ Inter-thread communication is a mechanism in which a thread is paused running in its critical section and another thread is allowed to enter (or lock) in the same critical section to be executed. It is complemented by following methods of object class

- wait()
- notify()
- notify all()

1. wait() method:- The wait method causes thread to release the lock and wait until either another thread invokes the notify() method or the notify All() method for this object, or specified amount of time has elapsed.

→ The current thread must own this objects monitor, so it must be called from the Synchronized method only otherwise it will throw exception.

| Method | Description |
|---|---|
| public final void() throws interupted exception | It waits until object is notified. |
| public final void wait(long timeout) throws interupted exception | It waits for the specified amount of |

2) notify() method:- The notify method wakes up a single thread that is waiting on this objects monitor. If any threads are waiting on this object, one of them is choosen to be awakened - The choice is arbitary and occurs at the discretion of the implementation

Syntax:-

```
public final void notify()
```

3) notifyAll() method:- Wakes up all the threads that are waiting on this object's monitor

Syntax:-

```
public final void notifyAll()
```

Example of Inter thread communication in java:-

Test.java

```
class Customer{
int amount=1000;
synchronized void withdraw(int amount){
System.out.println("going to withdraw....");
if(this.amount<amount){
System.out.println("Less balance: waiting for deposit...");
try{wait();}catch(Exception e){}
}
this.amount=amount;
System.out.println("withdraw complemented...");
```

```
synchronized void deposit (int amount){
System.out.println ("going to deposit....");
this.amount t = amount;
System.out.println ("deposit completed....");
notify ();
}
}
class Test{
 pubtic static void main (String args[]){
 final customer c = new customer ();
  new Thread(){
  public void run(). withdraw (15000); }
  }.start();
  new Thread(){
  public void run() {c.deposit (10000); }
  }.start();
  }}
```

Output:- going to withdraw....
less balance; waiting for deposit....
going to deposit....
deposit completed....
withdraw completed

2. Explain about Event classes, Event Sources, Event Listeners and the relationship among them.

A) In Java, events are the basic building blocks of user interface(UI) development. An event is an object that represents a change in state of an object. For example, a button click, a mouse move, or a key press are all events.

Events classes :- Event classes are event that indicates that a component-defined action occured like a button click or selecting an item from the menu-item list.

Event Sources :- Event sources are the objects that generate events. For example, a button is an event source that generates a button click event when the user clicks on it.

Event Listeners :- Event listeners are the objects that listen for events and respond to them. For example, a button listener is an event that listens for button click events and performs some action when the event occurs.

The relationship between event classes, event sources, and even listeners is as follows:

- Event classes are the classes that represent events. For example, the mouseEvent class represents mouse events.
- Event sources are the objects that generate events for example, a button is an event source that generates button clickevents.

• Event listeners are interfaces or classes that define methods to handle specific events. Event listeners are registered with an event sources to listen for particular types of events. When an event occurs, the corresponding method in the listener is invoked.

→ An event source (eg. button) generates a event object when a specific action happens.
→ This event object is then passed to the appropriate event listener (registered for that event type)
→ The event listener, implementing the relevant interface or extending the appropriate class, contains the logic to respond to the event.

This mechanism allows developers to create responsive applications by handling user interactions and responding to events triggered by the components in the program.

3. What is a layout manager? Explain the different types of Layout managers in detail?

A) In Java, a layout manager is a part of the java GUI framework that arranges components within a container. It helps in organizing the visual structure of a graphical user interface (GUI) by specifying how components should be positioned and sized.

There are several types of layout managers in java, each with its own rules for organizing components:

1. FlowLayout :-

- components are arranged in a row, and when the row is full, it continuous to the next row.
- Suitable for simple GUI's where components should flow naturally from left to right.

2. Border layout :-

- components are organized into five regions : North, South, East, West and center.
- Useful for creating more complex layouts with a central area and surrounding regions.

3. Grid layout :-

- components are arranged in a grid of rows and columns.
- Useful when you want to create a simple grid structure where each cell has the samesize.

4. CardLayout :-

- Allows switching between different panels or "cards" in the container.
- Useful for scenarios where you want to switch between different views or panels.

5. BoxLayout :-

- components are laid out either horizontally or vertically in a singleline
- provides more control over the arrangement of components in a specific direction.

6. GridBag Layout :-

- Offers the most flexibility among Layout managers.
- components are arranged using constraints, allowing for precise control over component placement and sizing

7. FlowLayout :-

- components are arranged in a single line, either horizontally or vertically.
- Supports alignment options for placing components within the container.

8. GroupLayout :-

- Introduced in Java SE 6, GroupLayout is designed for complex layouts.
- Allows developers to create move sophisticated and maintainable GUIs.

When designing a GUI, you can choose the layout manager that best fits your requirements based on the desired structured and organization of components

within the container. It's common to use a combination of layout managers to achieve more complex and dynamic layouts.

4. What are the different types of applets and explain.

A) Applet is a special type of java program that runs in a web browser is reffered to as Applet. It has less response time because it works on the client-side. It is much secured executed by the browser under any of the platforms such as windows, Linux and Macos etc. There are 2 types of applets that webpage can contain.

1. Local Applet
2. Remote Applet

* Local Applet :- Local Applet is written on our own, and then we will embed it into webpages. Local Applet is developed locally and stored in the local system. A web page doesn't need the get the information from the internet when it finds the local Applet in the system.

→ There are two attributes used in defining an applet, i.e, the codebase that specifies the path name and code that defined the name of the file that contains Applet's code.

```
<applet
    codebase = "tictactoe"
    code = "FaceApplet.class"
    width = 120
    height = 120>
</applet>
```

→ First, we will create a local Applet for embedding in a web page.

→ After that, we will add that local Applet to the web page.

Example:-

FaceApplet.java

```
import java.applet.*;
import java.awt.*;
import java.util.*;
import java.awt.event.*;
public class FaceApplet extends Applet
{
public void paint (Graphics g) {
  g.setcolor (color.red);
  g.drawstring ("welcome", 50, 50);
  g.drawLine (20, 30, 20, 300);
  g.drawRect (70, 100, 30, 30);
  g.fillRect (170, 100, 30, 30);
  g.drawoval (70, 200, 30, 30);
```

```
        g.setColor(color.pink);
        g.fillOval(170, 200, 30, 30);
        g.drawArc(90, 150, 30, 30, 30, 270);
        g.fillArc(270, 150, 30, 30, 0, 180);
    }
}
```

RemoteApplet:- A remote applet is designed and developed by another developer. It is located or available on a remote computer that is connected to the internet.

→ In order to run the applet stored in the remote computer, our system is connected to the internet then we can download and run it.

→ To locate and load a remote applet, we must know the applet's address on the web that is referred to as uniform Resou Recourse Locator (URL).

```
<applet
    codebase = "http://www.myconnect.com/applets/"
    code = "FaceApplet.class"
    width = 120
    height = 120>
</applet>
```

Example:-

```
import java.applet.Applet;
import java.aust.Graphics;
import java.rmi.register.locateRegisters;
import java.rmi.registers.Registers;
public class RemoteAppletclient extends
Applet {
  private Remote.AppletInterface
  remoteApplet;
  @Override
    public void init(){ try{
      Registaryg registary=
  LocateRegistaryg.getRegistary("localhost", 1099);
    remoteApplet = (RemoteAppletInterface)regis
    registry.lookup("RemoteApplet");
    }
    catch(Exception e){
      e.printStackTrace();
    }}
    @override
    public void paint(Graphics g){
      try{
          String remoteData = remoteApplet.fetchData();
          g.drawString("Recieved from remote:" + remoteData,20,20);
      }
    catch(Exception e){
      e.printStackTrace();
    }
}}
```

5. Explain any 5 swing components along with its methods swing components in java

Ans → Swing components are the basic building blocks of an applet/application. we know that swing is a GUI widget toolkit for java.

→ Every application has some basic interactive interface for the user. For example, a button, check-box, radio-button, text field etc, These together from the components in swing.

→ Swing components are the interactive elements in a java application.

Swing Components

Imageicon JButton Jlabel JPasswordfield Jtext

JcheckBox JRadioButton JList JcomboBox JFile

JTextfield JTextArea

1. ImageIcon:- The Imageicon component creates an icon sized-image from a image residing at the source URL

ex:- Imageicon homeIcon = newImageIcon("srelimages home.jpg"))

Note:- We would be using this image icon in further examples also.

2. JButton:- JButton class is used to create a push-button on the UI. The button can contain some display text or image. It generates an event when clicked and double-clicked.

→ A JButton can be implemented in the application by calling one of its constructors

Examples:-

(i) JButton OkBtn = new JButton ("OK");

→ This constructor returns a button with text on it.

(ii) JButton homeBtn = new JButton (homeIcon);

→ It returns a button with a homeIcon on it.

(iii) JButton btn2 = new JButton (homeIcon, "Home");

→ It returns a button with the homeIcon and text Home.

3. JLabel:- JLabel class is used to render a read-only text label or images on the UI. It does not generate any event.

Example:-

JLabel textLbl = new JLabel ("this is a text label.");

→ This constructor returns a label with text.

JLabel imglabel = new JLabel (homeIcon);

→ It returns a label with a home icon.

4. JTextField:- JTextField renders an editable single-line text box. A user can input non-formatted text in the box. To initialize the textfield, call its constructor and pass an optional integer parameter to it. This parameter send the width of the box measured by the no. of columns. It does not limit the no. of characters that can be input in the box.

Example:-

```
JTextField textBox = new JTextField (20);
```

→ It renders a text box of 20 column width.

5. JTextArea:- JTextArea class renders a multi-line text box. similar to the JTextfield, a user can input non-formatted text in the field. The constructor of JTextArea also expects two integer parameters which define the height and width of the text-area in columns. It does not restrict the no. of characters that the user can input in the text area

Example:-

```
JTextAreatxtArea = new JTextArea ("This text is default
                   text for text area.", 5, 20);
```

→ The above code renders a multi-line text area of height 5 rows and width 20 columns, with default text initialized in the text area.

6) JPasswordField :- JPassword Field is a subclass of JTextField class. It renders a text-box that masks the user input text with bullet points. This issued for inserting passwords into the application.

Example :-

```
JPasswordField pwdField = new JPasswordField(15);
var pwdvalue = pwdField.getpassword();
```