Name: Khushi Rana
Roll no: 8

Century Page: Date:

Questions.

1. Ans.

AWT	Swing
1. AWT components are heavy weight components.	1. Swings components are light weight components.
2. Platform dependent	2. Platform independent.
3. AWT program are not portable.	3. Swing programs are portable.
4. Execution is slower.	4. Execution is faster.
5. Components requires more memory space.	5. Components requires less memory space.
6. Has less advanced components.	6. Has more advanced components.
7. Doesn't support MVC pattern.	7. Support MVC pattern.
8. Javaxawt is the packa -ge name.	8. ~~S~~ javax.swing is the package name.

GUI:

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

public class SumandDifference Calculator extends
JFrame {
    private JTextfield num1field, num2field;
    private JLable resultlabel;
```

```
public SumAndDifferenceCalculater (){
setTitle ("Sum and Difference calculater");
setLayout (new FlowLayout ());

num1Field = new JTextField (10);
num2Field = new JTextField (10);
resultLabel = new JLabel ("Result");

JButton CalculateButton = new JButton ("Calculate");
calculateButton.addButtonMouseListener (new
MouseAdapter () {
@ override
public void mousePressed (MouseEvent e){
try {
int num1 = Integer.parseInt (num1Field.
    getText ());
int num2 = Integer.ParseInt (num2Field.get
    Text ());
resultLabel.setText ("sum: " + (num1 + num2));
} catch (NumberFormatException ex){
resultLabel.setText ("Invalid Input");
}
}
@ override
public void mouseReleased (MouseEvent e){
try {
int num1 = Integer.ParseInt (num1Field.get
    Text ());
int num2 = Integer.parseInt (num2Field.
    getText ());
resultLabel.setText ("Difference: " +
    (num1 - num2));
```

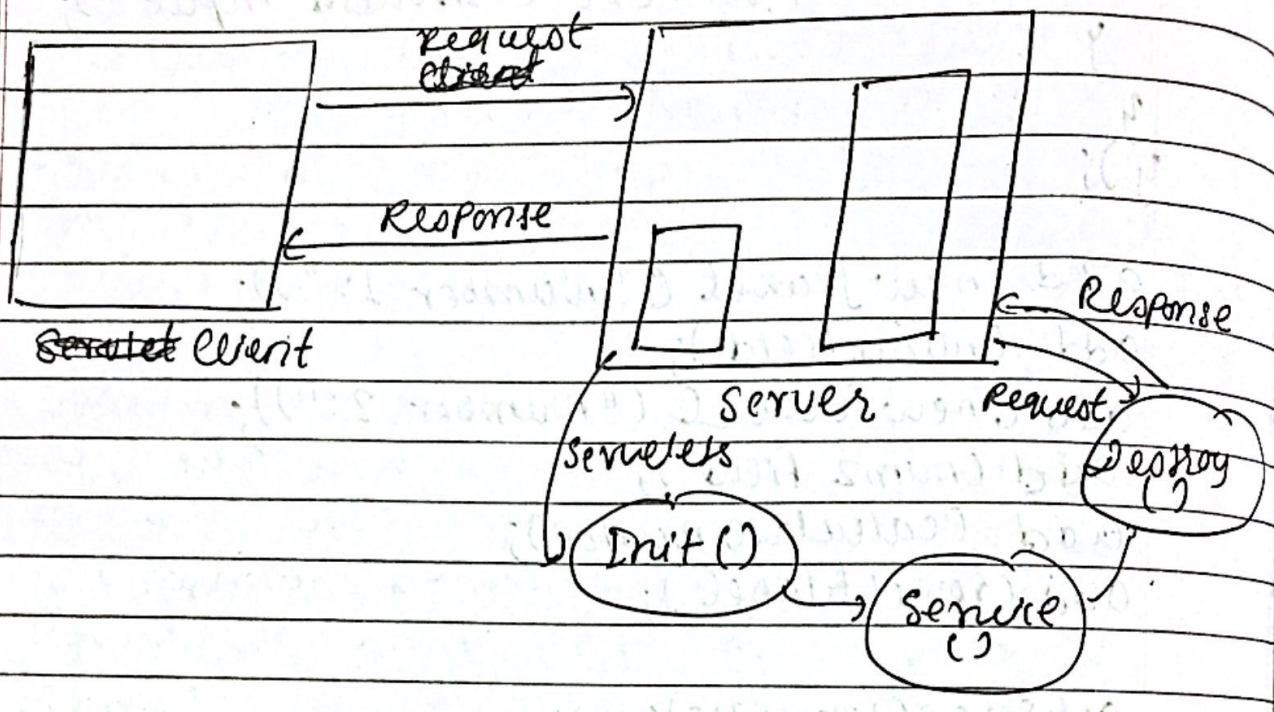
```
} catch (NumberFormatException ex) {
    resultLabel.setText("Invalid input");
  }
}
});

add(new JLabel("Number 1:"));
add(num1Field);
add(new JLabel("Number 2:"));
add(num2Field);
add(calculateButton);
add(resultLabel);

setSize(300, 200);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
}

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

Q. no. 2. Ans.



> The servlet is initialized by calling the init() method.
> The servlet calls service() method to process a client's request.
> The servlet is terminated by calling destroy() method.
> Finally, servlet is garbage collected by the garbage collector of the JVM.

1. Init() loading & instantiation.
> When a servlet is first requested, the servlet container load the servlet class.
> The container then initiates an object of the servlet.
> The init method is called only once. It is called when servlet is created, and not called for any user request afterwards.
> The init method is used to initialize the servlet.

Century
Page: Date:

Syntax of init method:

```
public void init () throws servlet Exception {
    // Initialization code ;...
}
```

2. Service ()

The servlet service () method is the main method to perform the actual task. The servlet container (web server) that calls the service () method to handle request comming from the client (browser) and to write the formatted response back to the client.

> The web container calls the service method each time when request for the servlet is received.

Syntax:

```
public void service (servlet Request request,
                     Servlet Response response)
throws servlet Exception, IO Exception
{
    // code
}
```

3. Destroy ()

When the servlet is no longer needed or the servlet is shutting down, the container calls the destroy () method of the servlet.

> This method is called only once and is used for cleanup activities & closing database connections, etc.

Syntax:

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

Once the servlet is destroyed, garbage collector componet of Jvm is responsible for collecting the garbages.

Index.html

```
<html>
<body>
<form action="demo serv" method "get">
username : <input type="text" name="username">
<br>
password: < input type="submit" value="login"
name="submit" />
</form>
</body>
</html>
```

Demo serv. java.

```
import java.io.*;
import javax.servlet.*;
import java.servlet.http.*;
import java.io.IOException.*;
import javax.servlet.http.HttpServletRequest.*;
import javax.servlet.http.HttpServletResponse.*;
public class Demoserv. extend HTTP Servlet {
    public void doget (Http servlet Request req,
        Http servlet Response res) throws servlet
        Exception
    {
        res.set (content type ("text/html").
        String username = req.get parameter
        ("username");
        String password = req.get parameter
        ("password").
        print user pw = res.getwriter().
        pw.println ("username" + username).
        pw.println ("password" + password);
    }
}
```

→ Distributed garbage collection.

Q.NO. 3. Ans.

RMI (Remote method Invocation) is a Java API that provide mechanism to create distributed application in java. It allows an object to invoke methods on a object running in another Java virtual machine (JVM).

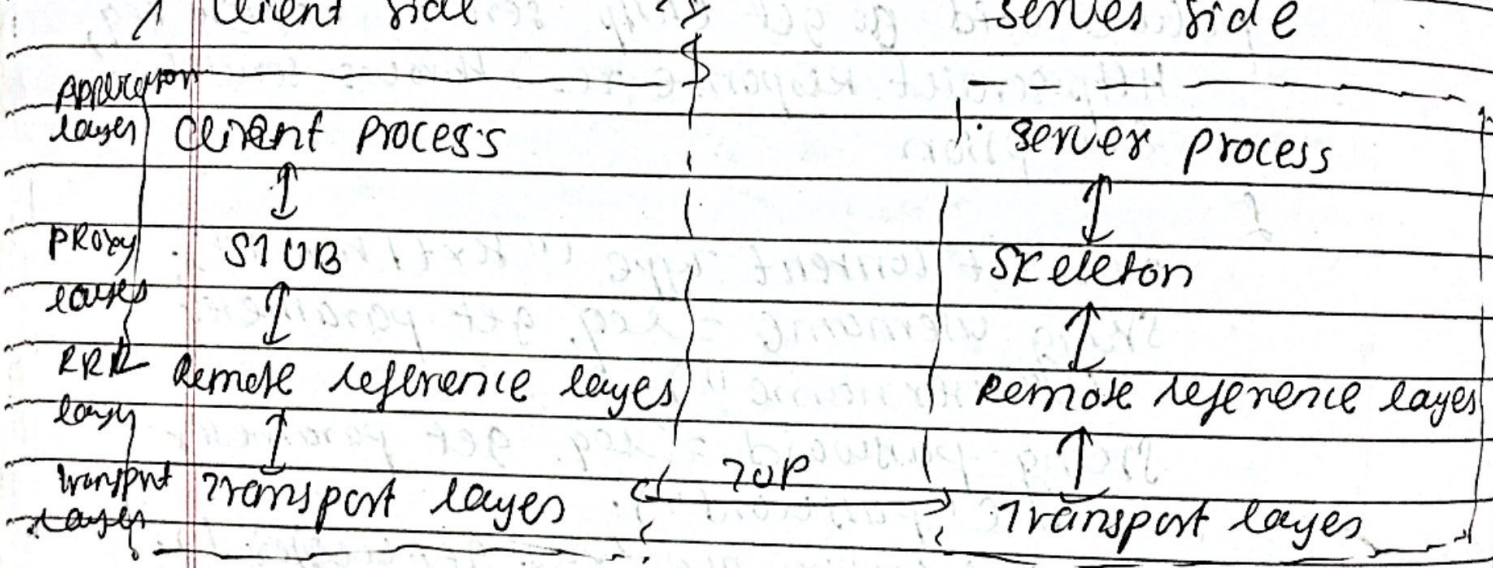


Fig: Architecture of RMI

1. Application layer
In this layer, client & server are involved in comm. The java program on Client side communicates with java program on server sides.

2. Proxy layer
This layer contains the Client STUB and Server Skeleton object.

a. Stub: A Stub is an object, that acts as a gate way for client side. All the outgoing

request are routed through it.

b. Skeleton: The skeleton is an object, that acts as a gateway for server side. All the ~~outgoing~~ incoming requests are routed through it.

3. RRL layer (Remote reference)
It is the layer which manages the reference made by the client to remote object.
It is also responsible to maintain session during the method call.
It is also responsible for handling duplicate objects.

4. Transport layer.
This layer connects the client and the server. This layer uses standard TCP/IP protocol for conn^n. This layer perform the actual transportation of data.

Q.
1. Define the remote interface
It defines the set of methods that can be invoked remotely.
It extends the java RMI remote interface.

```
import java.rmi.*;
import java.rmi.Remote.*;
public interface calc extends Remote
{
    int multiply (int a, int b) throws Remote
    Exception;
}
```

2. Creating a server
It implements the remote interface.
It extends java.rmi.server.UnicastRemoteObject or java.rmi.server remote object to facilitate comm^n.

```
import java.rmi.*;
import java.rmi.registry.*;

public class server {
    public static void main (String [] args)
    {
        try
        {
            calc Remote obj = new calcRemote();
            Registry registry = locate Registry.create
                                registry (9000)
            registry.rebind ("Product", obj);
            System.out.println ("server ready");
        }
        catch (Exception e)
        {
            System.out.println("server failed" +e);
        }
    }
}
```

3. Creating clients

It looks up the remote object in the RMI registry and invokes the remote method.

```
import java.rmi.*;
import java.rmi.registry.*;
public class client {
    public static void main(String[] args)
    {
        try
        {
            Registory registry = local registry.get
                                 Registry (9000);
            calc.obj = (calc.registry.lookup
                        ("product");
            int value = obj.multiply (6,5);
            System.out.printlen (value);
        }
        ceten (Exception.e)
        {
            system out. printlen ("server fail
                to (connect " + e)
        }
    }
}
```

Page: Date:

Q. NO. 4. Ans.

A Package is a group of similar types of classes, interfaces and sub-package.

creating a Package.

1. Choose a package name according to the naming convention.
2. Write package name at the top of every source file (classes, interface, enumerations and annotations).
3. Remember that there must be only one package statement in each source file.

```
// saved by A.java
package pack;
public class A
{
    public void msg()
    {
        System.out.println("Hello world");
    }

// saved by B.java
package my pack;
import pack;

class B
{
    public static void main(String args[])
    {
        A obj = new A();
```

```
        obj.msg();
    }
}
```

Q.NO. 5. ANS.

We need swing components because of the following reasons:

1. It procurde numerous no. of swing components for better GUI.
2. It is platform independent GUI toolkit.
3. It is light-weighted.
4. Swing components are executed in lesser amt. of time.
5. Swing components are building blocks for designing, developing and implementing an application.

Among the available numerous components checkboxes and radiobuttons are two of the component explained below:

1. Checkboxes

> It is the component that makes the GUI for selected and unselected for a list.

> They are used for enabling / disabling specific features or selecting multiple items simultaneously.

> For example, in an email application, checkboxes can be used to select multiple emails for deletion or archiving.

2. Radio buttons.
It is also widely used ~~swo~~ another swing component that is used in order to select one option & among the existing list of option. It can be used in scenario like answering question by selecting the correct answer from the given options.

Q. No. 6 & Ans.

We use ⊛ java Action event listener package that provide the diff java classes and interface. The action event listener is a method through which we can handle the events.

By using swing to handle the event there comes two participants the source & listener.

Swing
The object that generates the events.

listener
The object that notify when the event occur.

To handle the event we have diff listener interfaces and the methods through which we can handle them.

Interface	Method
Action listener	Void Action performed (Action Event e)
Adjustment listner	Void adjustment value changed (Adjust)
component listner	Void component hidden (component event e) Void component placed (component event e) Void component moved (component event e)
Container listner	Void component Added (Container event e)
Mouse listner	Void mouse Clicked (mouse event e) Void mouse entered (mouse event e) void mouse Exited (mouse event e) vaid mouse prened (mouse event e)

OR

Let us suppose a following program;

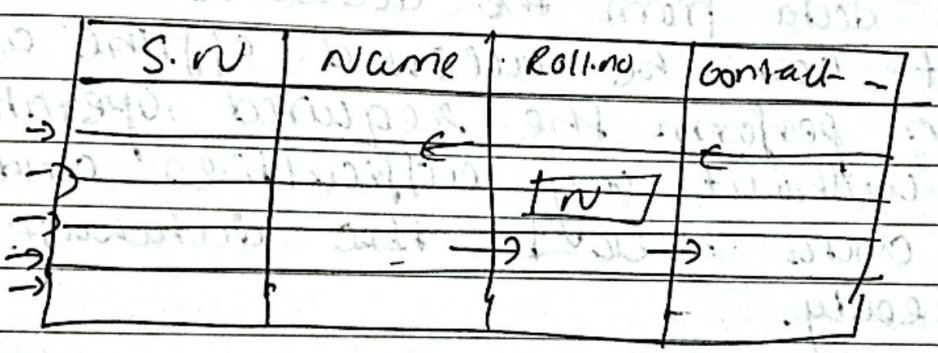
```
Class A
{
    JFrame F = new Frame();
    JButton b = new JButton ("Click");
    F.add (b);
    b. Action Listener (void actionperformed
        (Action listener e)
```

```
l
    JTextfield t = new JTextfield
        ("I am cucked");
    F.add(t);
}
F.setSize(1201, 1201);
F.setLayout(null);
F.setVisible(true);
}
```

Listener Interface	Adapter Class
1. Defines methods to handle specific types of events.	1. Provides default implementations for methods in a event listner interface.
2. You directly implement the Listener interface in your class.	2. You extend the adapter class which provides empty implements of listner method.
3. You must implement all methods defined by listner interface.	3. You only need to override the methods you are interset -ed in handling.
4. Provide more flexibility as you can implement multiple listner interface in a single class.	4. less flexible compared to listener interface as you can only extend on adapter class per listener.

5. More explicit	5. Less Explicit
6. Can sometimes leads to longer and less readle code, especially for classes implementing multiple interfaces.	6. Cleaner and more concise.
7- Eg: Action listener, mouse listener, Keylistener, etc.	7. Eg: Action adapter, mouse Adapter, Key adapter, etc.

Q.No. 7. Ans.



Row set is a Java API that is build on the top of Jule set.

It is extendable and more flexiable than result set.

In JDBC, while creating a conn betn Java program and Databases we need API like Rowset that helps in the smooth bridging betn them. While implementing Rowset on a Java program an object is first declared from the class Rowset factory and only after then the functionality of Rowset can be implements to get reach to the database.

and sending & receiving mails through ... available interface is Java mail API.

Century
Page: Date:

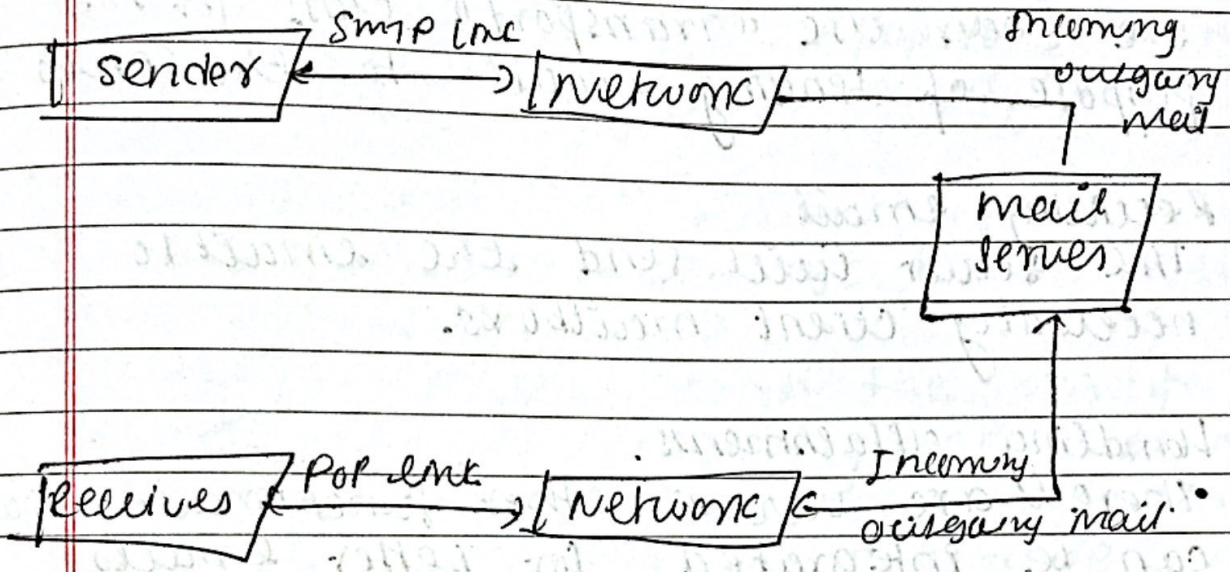
Rowset are of diff. types.
i.e. JDBC rowset, cached rowset, web rowset, Filter rowset, etc.
The explanation on the cached row set is in below paragraph.

Cached rowset
cached row set is one of the widely used available rowset.
cached rowset is a type of rowset which is portable and in-memory version of the data from the database. The cached rowset can be accessed offline and we can perform the required operations in it without any difficulties and put the data back to the database when your ready.

Q. No. 8. Ans.
(E-mail is used for sending or receiving information in any platform. The Java mail is a library or a framework that provides a platform independent (i.e. it can be sent, receive and open from any platform like window, Linux or mac, etc) and protocol-independent framework to build mail and messaging application.) ~~for this~~ In the Java we follow the SMTP protocol to send the message. the SMTP protocol is readily available in Java mail package. The package provide all the necessary

Scanned with CamScanner

classes and interfaces through which we send our email message. The Java mail API is a popular and well used method by the developers through. the message are transferred from the sender to the receiver.



We can use the following methods to send the email message from the sender to the receiver using Java mail API.

1. Set-up
At first one is required to import necessary files if need they are required to add Jar files or can also use technologies like Gradle.

2: Configure server-connection
After that it is required to make conn^n with server. For that we are supposed to give username, password and asked necessary inputs.

3. Creation of email
After that email is created which deals with writing data in subject, title, description and other component.

4. Sending email
We can use "Transport" class for the purpose of sending email to the server.

5. Receiving email
The server will send the email to necessary client mailboxs.

6. Handling attachments.
There are various other functionalities that can be integrated for better emails implementation.

Q. No. 9. Ans.

Javafx	Swing
1. They have more no. of components then swing.	1. They have less no. of components than Javafx.
2. It is more light-weight compared to swing.	2. It is heavy-weight compared to Javafx.
3. The program of Javafx executes faster than swing.	3. The program of swing executes slower than Javafx.
4. Javafx is more reliable & easy for the application development.	4. The swing less reliable than the Javafx.

of diff r appl cation.

5. With the help of the Javafx it is more easier to develo -p the web based appli- -cations and gaming applications.	5. It is less easy to develop the applications and the games with the help of the swing.
6. Java fx is the modernes developed API with more and advanced facilities.	6. Swing was used popularly before the development of Javafx.

1. Hbox layout

Hbox are used to arrange the single rows and columns horizontally.

Hbox layout deals with the horizontal displayment of the component. We may have a no. of nodes in a program. Using Hbox layout, all those code can be displayed horizontally.

ii. VBox

VBox are used to arrange the single rows and column vertically.

VBox layout deals with the vertical displayment of the used component. We may have a no. of nodes in a program. Using VBox layout, all those node can be displayed vertically.

10. Ans.

Servlet is a technology that is used to accept client request, accept it and give response to handle the web application dynamically.

There are two types of servlet:

1. Web server
2. Application server

```
<!DOCTYPE html>
<html>
<head>
<title> JSP Program to display "Tribhuvan University" 10 times </title>
</head>
<body>
<h1> displaying "Tribhuvan University" 5 times
</h1>
<table>
<% for (int i=1; i<=5, i++){
%>
<tr><td> Tribhuvan University </td></tr>
<% } %>
</table>
</body>
</html>
```

output:

Tribhuvan University
Tribhuvan University
Tribhuvan University
Tribhuvan University
Tribhuvan University

Century
Page: Date:

1. I. Ans.

CORBA stands for common object request Broker Architecture is a model that serves the comm^n between a server and a client. It has huge significance. Some are:

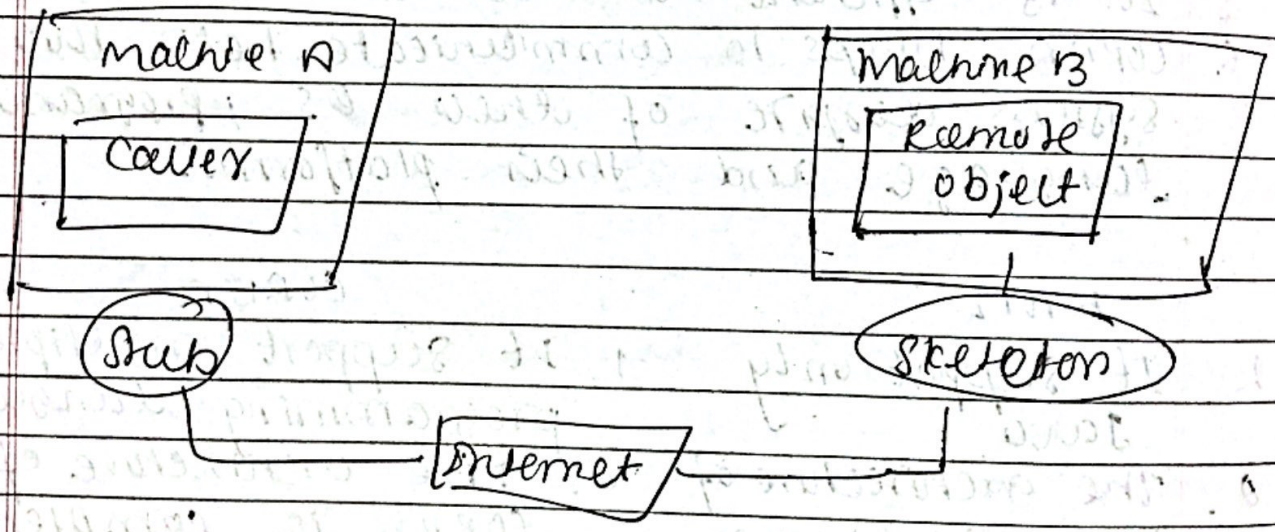
1. It is independent of any programming language.
2. It reduce redundancy of the code.
3. It can be used to access both remote and locally available data.
4. It provided greater functionalities than other existing models.
5. It is efficient & well-structured architecture.
6. CORBA helps to communicate betn two system despite of their OS / programming language and their platforms.

RMI	CORBA
2. It support only Java	1. It support multiple programming language.
2. The architecture of RMI is simpler than CORBA.	2. The architecture of CORBA is complex than RMI.
3. Java RMI is a server-centric model.	3. CORBA is a peer to peer system.
4. It's execution is slower than CORBA.	4. It's execution is faster than RMI.
5. Provides security services such as authentication & encryption.	5. Provides security services such as authentication, encryption and authori-zation.
6. Platform dependent	6. Platform independent.

Q.NO. 1. Ans.

The significeene of STUB and skeleton areas follows:

B. A STUB is an object, that acts a gateway for client side. All the outgoing request are routed through it.
The skeleton is an object that acts a gateway for server side. All the incoming request are routed through it.
STUB communicates with this skeleton to pass request to the remote object.



Defining the remote interface (Client Program)

```
import java.io.*;
import java.rmi.*;
import java.rmi.Remote;

public class Count
public interface Factorial extends Remote
{
```

int computeFactorial(int number) throws RemoteException;
}

2. creating a server

```
import java.rmi.*;
import java.rmi.registry.*;

public class FactorialServer{
    public static void main(String[] args)
    {
        try
        {
            FactorialRemote obj = new FactorialRemote();
            Registry registry = locateRegistry.createRegistry(6060);
            registry.rebind("Factorial", obj);
            System.out.println("Server ready");
        }
        catch(Exception e)
        {
            System.out.println("Server failed" + e);
        }
    }
}
```

3. Creating a Client.

```
import java.rmi.*;
import java.rmi.registry.*;
public class FactorialClient {
    public ~~void~~ static void main(String[] args)
{
    try
    {
        Registory registory = local registry.get
            Registory(9900);
        Factorial obj = factorial.registory.lookup
            ("Factorial");
        int ~~value~~ number = 5;
        int value = factorial.compcite factorial
            (number);
        System.out.println("Factorial of " +
            number + " is" + result);
    }
    catch (Exception e).
    {
        System.out.println("Server Fail to
            connect " + e)
    }
}
}
```

Q. No. 5. Ans.

We use need Adapter class in event handling because:

Adapter classes are commonly used in GUI (Graphical User Interface) programming where there are many diff. events that need to be handled, such as button clicks, mouse movements and key presses.

3 The use of adapter classes simplifies the code by providing a template for handling event.

1. Mouse Adapter
Mouse Adapter is used to handle mouse events. It contains empty methods for each type of mouse event such as mouseClicked(), mouse Pressed() and mouseReleased().

2. Key Adapter
is used to handle key event. It contains empty method for each type for key events, such as KeyPressed(), keyReleased() and keyTyped().

3. Window Adapter
is used to handle window events. It contains empty methods for each type of window event, such as windowOpened(), Window Closed() and windowConfred().

3 Instead of writing code for every possible

event, the programmer only need to write code for the events that are relevant to the application. This makes code more readable, easier to maintain, and less prone to errors.

Delegation event model means: a source generates an event and send it to one or more listners.

a. Source: Event are generated from the source. There are various source like buttons, checkboxes, list, menu-items, choice, text component, windows etc. to generate event.

b. Listeners: Listners are used for handling the event generated from source. Each of these listeners represents interfaces that are responsible for handling events.