**### RMI Exp 4 code**

**\*IHello.java\***

import java.rmi.\*;

public interface IHello extends Remote{

public String message() throws RemoteException;

}

**\*\*HelloImpl.java\*\***

java

import java.rmi.\*;

import java.rmi.server.\*;

public class HelloImpl extends UnicastRemoteObject

implements IHello{

public HelloImpl() throws RemoteException {

//There is no action need in this moment.

}

public String message() throws RemoteException {

return ("Hello");

}

}

**\*\*HelloServer.java\*\***

java

import java.rmi.\*;

public class HelloServer {

private static final String host = "localhost";

public static void main(String[] args) throws Exception {

//\*\* Step 1

//\*\* Declare a reference for the object that will be implemented

HelloImpl temp = new HelloImpl();

//\*\* Step 2

//\*\* Declare a string variable for holding the URL of the object's name

String rmiObjectName = "rmi://" + host + "/Hello";

//Step 3

//Binding the object reference to the object name.

Naming.rebind(rmiObjectName, temp);

//Step 4

//Tell to the user that the process is completed.

System.out.println("Binding complete...\n");

}

}

**\*\*HelloClient.java\*\***

java

import java.rmi.ConnectException;

import java.rmi.Naming;

public class HelloClient

{

private static final String host = "localhost";

public static void main(String[] args)

{

try

{

//We obtain a reference to the object from the registry and next,

//it will be typecasted into the most appropiate type.

IHello greeting\_message = (IHello) Naming.lookup("rmi://"

+ host + "/Hello");

//Next, we will use the above reference to invoke the remote

//object method.

System.out.println("Message received: " +

greeting\_message.message());

}

catch (ConnectException conEx)

{

System.out.println("Unable to connect to server!");

System.exit(1);

}

catch (Exception ex)

{

ex.printStackTrace();

System.exit(1);

} }

}

**### RMI Exp 5 code**

**\*\*Adder.java\*\***

import java.rmi.\*;

public interface Adder extends Remote{

public int add(int x,int y)throws RemoteException;

}

**\*\*AdderRemote.java\*\***

import java.rmi.\*;

import java.rmi.server.\*;

public class AdderRemote extends UnicastRemoteObject implements Adder{

AdderRemote()throws RemoteException{

super();

}

public int add(int x,int y){return x+y;}

}

**\*\*MyServer.java\*\***

import java.rmi.\*;

public class MyServer{

public static void main(String args[]){

try{

Adder stub=new AdderRemote();

Naming.rebind("rmi://localhost:5000/sonoo",stub);

}catch(Exception e){System.out.println(e);}

}

}

**\*\*MyClient.java\*\***

import java.rmi.\*;

public class MyClient{

public static void main(String args[]){

try{

Adder stub=(Adder)Naming.lookup("rmi://localhost:5000/sonoo");

System.out.println(stub.add(34,4));

}catch(Exception e){}

}

}

### Wrapper Exp 6

**\*\*Sender.java\*\***

java

import java.net.\*;

import java.util.\*;

public class Sender {

public static void main(String[] args) throws Exception{

Scanner scn=new Scanner(System.in);

System.out.println("Enter your Message:");

String str=scn.nextLine();

DatagramSocket ds=new DatagramSocket();

InetAddress ip=InetAddress.getByName("127.0.0.1");

DatagramPacket dp=new

DatagramPacket(str.getBytes(),str.length(),ip,3000);

ds.send(dp);

ds.close();

System.out.println("Message has been sent to Receiver Class Please Check:"+str);

}

}

```

**\*Receiver.java\***

import java.net.\*;

public class Receiver {

public static void main(String[] args) throws Exception{

System.out.println("Waiting for sender to send message");

DatagramSocket ds=new DatagramSocket(3000);

byte[] buf=new byte[1024];

DatagramPacket dp=new DatagramPacket(buf,1024);

ds.receive(dp);

String str=new String(dp.getData(),0,dp.getLength());

System.out.print(str);

ds.close();

System.out.println("Message received Successfully..");

}

}

**SIMPLE WRAPPING**

public class Wrapper {

public static void main(String[] args)

{

char ch = 'a'; // char data type.

Character chrobj = new Character(ch); // Wrapping char type value into Character object.

byte a = 10; // byte data type value.

Byte byteobj = new Byte(a); // Wrapping byte type value into Byte object.

int b = 20; // int type value.

Integer intobj = new Integer(b); // Wrapping int type value into Integer object.

float c = 18.6f; // float type value.

Float floatobj = new Float(c); // Wrapping float type value into Float object.

double d = 250.5; // double data type value.

Double doubleobj = new Double(d); System.out.println("================Wrapper Class Experiment By BECOMPA69&66 ==========================");

System.out.println("Displaying values of Wrapper class objects:");

System.out.println("Character object: " + chrobj);

System.out.println("Byte object: " + byteobj);

System.out.println("Integer object: " + intobj);

System.out.println("Float object: " + floatobj);

System.out.println("Double object: " + doubleobj);

System.out.println("\n");

// Retrieving primitive data type values from objects.

// Unwrapping objects to primitive data type values.

char chr = chrobj;

byte by = byteobj;

int in = intobj;

float fl = floatobj;

double db = doubleobj;

// Displaying the values of data types.

System.out.println("Displaying unwrapped values: ");

System.out.println("char value: " + chr);

System.out.println("byte value: " + by);

System.out.println("int value: " + in);

System.out.println("float value: " + fl);

System.out.println("double value: " + db);

}

}

XADL CODE:

<XADL>

<component type="Component" id="component1">

<description type="Description">Candidate Login</description>

<interface type="Interface" id="interface1">

<description type="Description">To proceed Login</description>

<direction type="Direction">out</direction>

</interface>

</component>

<connector type="Connector" id="connector2">

<description type="Description">Apply for Jobs</description>

<interface type="Interface" id="interface1">

<description type="Description">To get filtered Jobs/description>

<direction type="Direction">in</direction>

</interface>

<interface type="Interface" id="interface2">

<description type="Description">To Apply for Job</description>

<direction type="Direction">out</direction>

</interface>

</connector>

</XADL>

**#One component & One Connector**