RMI code tp:Echo Class:----------

```
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;

public class EchoClient {
   public static void main(String args[]){String url="rmi://localhost/irisi";
   try{Echo od=(Echo)Naming.lookup(url);System.out.println(od.echo("Message "));
      }catch(Exception e){
         System.out.println("Serveur Introuvble");}}}
```

Echo IMpl:----------

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class EchoImpl extends UnicastRemoteObject implements Echo {
   private static final long serialversionUID=1L;
   protected EchoImpl() throws RemoteException {
   }
   public String echo (String str) throws RemoteException{
      return "Le serveur Repond au : "+str;}}
```

EchoAppliServer:----------

```
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
public class EchoAppliServer {
   public static void main(String args[]){
      try{
         EchoImpl objetDist = new EchoImpl();
         LocateRegistry.createRegistry(1099);
         Naming.rebind("irisi",objetDist);
         System.out.println("l objet distant (objetDist) est enregisté");
      }catch(Exception e){
         System.out.println("Serveur non lancé");}}}
```

Client:----------

```
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
public class EchoClient {
   public static void main(String args[]){
      String url="rmi://localhost/irisi";
      try{
         Echo od=(Echo) Naming.lookup(url);
         System.out.println(od.echo("Message du Client"));
      }catch(Exception e){
         System.out.println("Serveur Introuvble");}}}
```

```
########################INET
ADRESS#########################

import java.net.InetAddress;

import java.net.UnknownHostException;

import java.util.ArrayList;

import java.util.List;

public class OtherHostName {
```

```
 public static String getHostName(String name) {
       String Machine;
       InetAddress ipAddr;
       try {
          ipAddr=InetAddress.getByName(name);
//                Machine=ipAddr.getHostAddress();
          Machine=ipAddr.getAddress().toString();
       }catch(UnknownHostException ex) {
          Machine="unkown machine ";
       }
       return "la machine "+name +" a pour adress ip : "+ Machine;
   }

//get local

InetAddress ipAddr;

         try {ipAddr=InetAddress.getLocalHost();

                  myLocalMachine=ipAddr.toString();

//
   public static List<String>  getAllHostAdress(String name) {
       List<String> machine = new ArrayList<>();
       InetAddress[] ipAddr;
       try {
          ipAddr=InetAddress.getAllByName(name);
          for (InetAddress adr : ipAddr) {
             machine.add(adr.getHostAddress());
          }
       }catch(UnknownHostException ex) {
          machine.add("unkown machine ");
       }
       return machine;
   }
}
```

```
########################UDP#########################
#####
```

Server:=----------

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class Server {
	static DatagramSocket ds;
	static FileOutputStream fos;
	public static  void recept(int port ,String host) {
		byte [] tompon =new byte[1024];
		InetAddress ipAddr;
		try
		{
			ipAddr=InetAddress.getByName(host);
			DatagramSocket mysock=new DatagramSocket(port,ipAddr);
			DatagramPacket mypacket=new
DatagramPacket(tompon,tompon.length);
			mysock.receive(mypacket);
			System.out.println("le serveur recoit les donnees de la part du
"+mypacket.getSocketAddress());
			System.out.println("il a recu:\" "+new
String(mypacket.getData())+"  \" .");
		}
		catch(IOException e) {
		}
	}
	public static  void receptFile(int port ,String host) {
		byte [] tompon =new byte[65024];
		InetAddress ipAddr;
		FileOutputStream fis;
		try
		{
			ipAddr=InetAddress.getByName(host);
			DatagramSocket mysock=new DatagramSocket(port,ipAddr);
			DatagramPacket mypacket=new
DatagramPacket(tompon,tompon.length);
			fis=new
FileOutputStream("C:\\Users\\hp\\Downloads\\Received.pdf");
//			fis.write(tompon);
			mysock.receive(mypacket);
			System.out.println("received ");
			fis.write(mypacket.getData());
//			System.out.println("le serveur recoit les donnees de la part du
```

```
"+ddress());
                        System.out.println("il a recu:\" "+new
String(mypacket.getData())+"  \" .");
                                                mysock.close();
                        fis.close();
                }
                catch(IOException e) {
                }
        }
        public static void receivePDF() {
                byte[] tampon = new byte[65024];

                try {

                fos = new FileOutputStream(new
File("C:\\Users\\hp\\Downloads\\Received.pdf"));
                ds = new DatagramSocket(1234);
                DatagramPacket dp = new DatagramPacket(tampon, tampon.length);
                System.out.println("Le serveur vient de recevoir un packet de : " +
dp.getSocketAddress());
                ds.receive(dp);
                fos.write(dp.getData());
                System.out.println("Le message reu est : " + new
String(dp.getData()));
                fos.close();
                } catch (Exception e) {
                e.printStackTrace();
                }
                } // Fin receivePDF

        public static void main(String[] args)
        {
                System.out.println("RECEPTION ....");
                receivePDF();
Server.receptFile(1000, "127.0.0.1");
        }
}
```

Client----------

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.PrintStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
```

```
import java.net.InetAddress;

public class Client {
	static DatagramSocket ds;
public static  void envoyer(int port ,String host,String msg) {
	byte [] tompon =msg.getBytes();
	InetAddress ipAddr;
	try
	{ ipAddr=InetAddress.getByName(host);
		DatagramSocket mysock=new DatagramSocket();
		DatagramPacket mypacket=new
DatagramPacket(tompon,tompon.length,ipAddr,port);
		mysock.send(mypacket);
		System.out.println("Envoyer   "+msg +"a  "+ host);
	}
	catch(IOException e) {
	}
}
public static void EnvoyerFile(String path, String host, int port) {
	FileInputStream fis;

	try {
		fis=new FileInputStream(path);

	byte[]buffer =new byte[65024];
	ByteArrayOutputStream b=new ByteArrayOutputStream();
	fis.read(buffer);
	b.write(buffer);
	DatagramSocket mysock=new DatagramSocket();
	InetAddress ipAddr=InetAddress.getByName(host);
	DatagramPacket mypacket=new
DatagramPacket(buffer,buffer.length,ipAddr,port);
	fis.close();
	mysock.send(mypacket);

} catch (FileNotFoundException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	} catch (IOException e) {
	// TODO Auto-generated catch block
	e.printStackTrace();
}
```

```
}
public static void envoiPDF(String host, int port, String chemin) {
FileInputStream fis;
InetAddress adr;
try {
fis = new FileInputStream(chemin);
byte[] clientBuffer = new byte[65024];
ByteArrayOutputStream baos = new ByteArrayOutputStream();
fis.read(clientBuffer);
baos.write(clientBuffer);
ds = new DatagramSocket();
adr = InetAddress.getByName(host);
DatagramPacket dp = new DatagramPacket(clientBuffer, clientBuffer.length, adr,
port);
fis.close();
System.out.println("Le serveur vient d'envoyer un PDF  : " + adr.toString());
ds.send(dp);
System.out.println("Le fichier est envoy");
} catch (Exception e) {
e.printStackTrace();
}

} // Fin envoiPDF



public static void main(String[] args)
{
//
//Client.envoyer(1000, "127.0.0.1", "hi khadija");
        String path = "C:\\Users\\hp\\Downloads\\test.pdf";

        System.out.println("ENVOI ....");

        String host = "localhost";
        int port = 1234;
        String msg = "Bonjour";
        envoiPDF(host, port, path);
Client.EnvoyerFile("C:\\Users\\hp\\Downloads\\test.pdf", "127.0.0.1",1000);
}
}
########################################SOCKET
TCP#######################

@@@@@@@@@@SEND
SCREEN@@@@@@@@@@@@@@@@@@@@@@@@@@@@

import java.awt.AWTException;
import java.awt.Dimension;
```

```
import java.awt.HeadlessException;
import java.awt.MouseInfo;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.Toolkit;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.io.PrintStream;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.List;
import javax.imageio.ImageIO;
public class client {
        Socket sock=null;
          public void connexion(String host, int port)
          { try
            {     System.out.println("Le client cherche  se connecter au serveur " + host
+ "@"+port);
                  sock = new Socket(host, port);
                  System.out.println("Le client s'est connect sur serveur " + host +
"@"+port);
             }
            catch(IOException e) { }
          public void sendScreen() {
                                          Dimension sizeScreen=
Toolkit.getDefaultToolkit().getScreenSize();
                                          int width = (int)sizeScreen.getWidth();
                                          int height = (int)sizeScreen.getHeight();
                                          Point p =
MouseInfo.getPointerInfo().getLocation();
                                          double CoeffHC=height/p.getX();
                                          double CoeffWC=width/p.getY();
                                          System.out.println("transmetter  Screen
resolution : " + "width : " + width + " height : " + height);
                                          OutputStream out;
                          try {
                                  PrintStream output=new
PrintStream(sock.getOutputStream());
                                  output.println(CoeffHC);
                                  output.println(CoeffWC);
                                  output.println(p.getX());
                                  output.println(p.getY());
                                  output.flush();
                                  output.close();
                                  out = sock.getOutputStream();
                             InputStream sInFromServer = sock.getInputStream();
                             DataInputStream in = new DataInputStream(sInFromServer);
```

```
                        //Takes screenshot
                    BufferedImage screencap = new
Robot().createScreenCapture(new
Rectangle(Toolkit.getDefaultToolkit().getScreenSize()));
                    ByteArrayOutputStream baos = new ByteArrayOutputStream();
                    ImageIO.write(screencap,"jpg",baos);
                    //Gets screenshot byte size and sends it to server and flushes
then closes the connection
                    byte[] size = ByteBuffer.allocate(4).putInt(baos.size()).array();
                    out.write(size);
                    out.write(baos.toByteArray());
                    out.flush();
                    out.close();

                      } catch (IOException | HeadlessException | AWTException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();}}
        public void fermer(){
                  try{
                    System.out.println("Le client ferme la connexion au serveur ");
                    sock.close();
                            }
                    catch(IOException e){}}}
class pointer {
        double x;
        double y;
        pointer(double x,double y){
                this.x=x;
                this.y=y;
        }
}

SERVER-------------------------

import java.awt.AlphaComposite;
import java.awt.Dimension;
import java.awt.Graphics2D;
import java.awt.Toolkit;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.JFrame;
```

```
public class server  extends Thread
{ ServerSocket serveur;
static final int port=10000;
public server()
{ try
  { serveur=new ServerSocket(port);
  System.out.println("-----------Le serveur est en coute sur le "+port);
          this.start();}
  catch(IOException e) { System.exit(1); }}

public void run()
{ Socket sock;
 // Traitement t;
 String text;
  try
  { while(true)
    {
          System.out.println("-----------Le serveur est en attente ");
          sock=serveur.accept();
      System.out.println("-----------Le serveur a accept la connexion avec
"+sock.getInetAddress());
      this.receiveScreen(sock);
    }
  }
  catch(IOException e) { }
}
public void receiveScreen(Socket sock) {
DataInputStream in;
Dimension sizeScreen= Toolkit.getDefaultToolkit().getScreenSize();
// width will store the width of the screen
int width = (int)sizeScreen.getWidth();
// height will store the height of the screen
int height = (int)sizeScreen.getHeight();
int serverScreenW=1000;
int serverScreenH=2000;
BufferedReader d;
double posCurX,posCurY;
          try {
                    d = new BufferedReader(new
InputStreamReader(sock.getInputStream()));
       double tab[] = new double[4];
       int index=0;
                    for (String line; (line = d.readLine()) != null;) {
           tab[index]=Double.parseDouble(line);
           index++;}
        System.out.println("cursor position client" +tab[2]+"  "+tab[3]);
    in = new DataInputStream(sock.getInputStream());
    DataOutputStream out = new DataOutputStream(sock.getOutputStream());
    byte[] sizeAr = new byte[4];
    in.read(sizeAr);
    int size = ByteBuffer.wrap(sizeAr).asIntBuffer().get();
//    int length = in.readInt();
    byte[] imageAr = new byte[size];
    in.readFully(imageAr);
    BufferedImage image = ImageIO.read(new ByteArrayInputStream(imageAr));
```

```
//we can specify any size to try if the scaling works
   BufferedImage i=this.createResizedCopy(image, serverScreenW, serverScreenH,
true);
   System.out.println("cursor position server x= "+(serverScreenH/tab[0])+" "+"y=
"+(serverScreenW/tab[1]) );
   System.out.println("check the screen in Downloads//scaledScreen.png");
   ImageIO.write(i, "jpg", new File("C:\\Users\\hp\\Downloads\\scaledScreen.png"));
         } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
         }
}
public BufferedImage createResizedCopy(BufferedImage originalImage,
      int scaledWidth, int scaledHeight,
      boolean preserveAlpha)
{
   System.out.println("resizing  to receiver screen size... width="+scaledWidth+"
Height="+scaledHeight);
   int imageType = preserveAlpha ? BufferedImage.TYPE_INT_RGB :
BufferedImage.TYPE_INT_ARGB;
   BufferedImage scaledBI = new BufferedImage(scaledWidth, scaledHeight,
imageType);
   Graphics2D g = scaledBI.createGraphics();
   if (preserveAlpha) {
      g.setComposite(AlphaComposite.Src);
   }
   g.drawImage(originalImage, 0, 0, scaledWidth, scaledHeight, null);
   g.dispose();
   return scaledBI;
}}

Start Client----------------

client c=new client();
         c.connexion("127.0.0.1", 10000);
                  double mouseX = MouseInfo.getPointerInfo().getLocation().getX();
     double mouseY = MouseInfo.getPointerInfo().getLocation().getY();

         c.sendScreen();
     c.fermer();

----------new server();

####################TCP
############################

CLIENT---------------------

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;
```

```
public class Client {
Socket s;
public Client(int port,String host) {
        try
   {    System.out.println("Le client cherche  se connecter au serveur " + host +
"@"+port);
        s = new Socket(host, port);
        System.out.println("Le client s'est connect sur serveur " + host + "@"+port);
   }
   catch(IOException e) { }
}
void requete(String msg) {
         try{
                        System.out.println("Le client cherche  rcuprer le canal de
communication ");
                        PrintStream output=new PrintStream(s.getOutputStream());
                        System.out.println("Le client cherche  envoyer la donne au
serveur " );
                        output.println(msg);
                 }
                 catch(IOException e){
                 }
}
public void recevoir(){
         try{
                 System.out.println("Le client cherche  rcuprer le canal de
communication ");
                 BufferedReader entree=new BufferedReader(new
InputStreamReader(s.getInputStream()));
      String text=entree.readLine();
        System.out.println("j'ai recu "+text);
    }
         catch(IOException e){
         }
 }
public void fermer(){
            try{
                 System.out.println("Le client ferme la connexion au serveur ");
                 s.close();
                        }
                 catch(IOException e){}}}

 public void envoiFileByte(String path){

try{ FileInputStream fs=new FileInputStream(path);

byte b[]=new byte[1000000];

fs.read(b,0,b.length);

PrintStream pr=new PrintStream(sock.getOutputStream());

pr.write(b, 0, b.length);}       catch(IOException e){}}

SERVER----------------------------------
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
public class Serveur {
   ServerSocket ss;
         public Serveur(int port) {
                  try {
                           this.ss = new ServerSocket(port);
                  } catch (IOException e) {
                           // TODO Auto-generated catch block
                           e.printStackTrace();}}
   public void servir() {
         Socket sock;
           String text;
         System.out.println("Le serveur est en attente ");
         try {
         sock=ss.accept();
       System.out.println("Le serveur a accept la connexion avec
"+sock.getInetAddress());
     //t=new Traitement(sock);
       BufferedReader entree= new BufferedReader(new
InputStreamReader(sock.getInputStream()));
       text=entree.readLine();
       System.out.println(text);
       System.out.println("Le client cherche  rcuprer le canal de communication ");
                  PrintStream sortie=new PrintStream(sock.getOutputStream());
                  System.out.println("Le client charche  envoyer la donne au serveur " );
                  sortie.println(text+" a ete recu par le serveur");
                  sock.close();}
         catch(IOException e) {
                           e.printStackTrace();}



 public void receiveByteFile(Socket sock) {

           InputStream in ;OutputStream out ;

           try {sleep(10000);

         } catch (InterruptedException e) {

         e.printStackTrace();} // takes input from the client socket

          try {in = new DataInputStream(sock.getInputStream());

         //writes on client socket

           out = new DataOutputStream(sock.getOutputStream());

           // Receiving data from client
```

```
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            byte buffer[] = new byte[1024];
           baos.write(buffer, 0 , in.read(buffer));
          System.out.println("------------byte file");
            byte result[] = baos.toByteArray();
            FileOutputStream fs = new FileOutputStream(new
File("C:\\Users\\hp\\Downloads\\received.png"));
            BufferedOutputStream  bs = new BufferedOutputStream(fs);
          bs.write(buffer);bs.close(); bs=null;
            String res = Arrays.toString(result);out.write(result);
            System.out.println("------------Closing connection");
                  sock.close();
          } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();}}
```

##

Sockets and threads:

ou will see that the second client cannot be connected until the first client closes its connection. To allow simultaneous connections we should know multithreaded programming. Here in the following Multithreaded Socket Programming , you can connect more than one client connect to the server and communicate.

How it works?

For each client connection, the server starts a child thread to process the request independent of any other incoming requests.

Main:

```
import java.net.*;
import java.io.*;
public class MultithreadedSocketServer {
  public static void main(String[] args) throws Exception {
    try{ ServerSocket server=new ServerSocket(8888);int counter=0;
      System.out.println("Server Started ....");
```

```
    while(true){counter++;

      Socket serverClient=server.accept();  //server accept the client connection
request

      System.out.println(" >> " + "Client No:" + counter + " started!");

      ServerClientThread sct = new ServerClientThread(serverClient,counter); //send
the request to a separate thread

      sct.start();}    }catch(Exception e){

    System.out.println(e);

}}}

-------------------SERVER SIDE----------------

class ServerClientThread extends Thread {

  Socket serverClient;int clientNo;int squre;

  ServerClientThread(Socket inSocket,int counter){

    serverClient = inSocket;

    clientNo=counter;}

  public void run(){try{

     DataInputStream inStream = new
DataInputStream(serverClient.getInputStream());

     DataOutputStream outStream = new
DataOutputStream(serverClient.getOutputStream());

     String clientMessage="", serverMessage="";

     while(!clientMessage.equals("bye")){

       clientMessage=inStream.readUTF();

       System.out.println("From Client-" +clientNo+ ": Number is :"+clientMessage);

       squre = Integer.parseInt(clientMessage) * Integer.parseInt(clientMessage);

       serverMessage="From Server to Client-" + clientNo + " Square of " +
clientMessage + " is " +squre;

       outStream.writeUTF(serverMessage);

       outStream.flush();}

     inStream.close();outStream.close(); serverClient.close();

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

    }finally{
```

```
    System.out.println("Client -" + clientNo + " exit!! ");}}}
```

Client ----------

```
import java.net.*;import java.io.*;

public class TCPClient {

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

    Socket socket=new Socket("127.0.0.1",8888);

    DataInputStream inStream=new
DataInputStream(socket.getInputStream()); DataOutputStream
outStream=new DataOutputStream(socket.getOutputStream());

   BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));

    String clientMessage="",serverMessage="";

    while(!clientMessage.equals("bye")){

      System.out.println("Enter number :");clientMessage=br.readLine();

      outStream.writeUTF(clientMessage);outStream.flush();

      serverMessage=inStream.readUTF();
System.out.println(serverMessage);}

    outStream.close();outStream.close(); socket.close();

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

################MULTI CASST#############

The following code contains a program that creates a multicast socket that receives datagram packets addressed to the 230.1.1.1 multicast IP address.

```
import java.io.IOException;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.net.MulticastSocket;

public class UDPMultiCastReceiver {
```

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

        int mcPort = 18777;

        InetAddress mcIPAddress = InetAddress.getByName("230.1.1.1");

        try (MulticastSocket  mcSocket = new MulticastSocket(mcPort); ){

      System.out.println("Multicast Receiver running at:"
+mcSocket.getLocalSocketAddress());

            // Join the group

            mcSocket.joinGroup(mcIPAddress);              DatagramPacket packet = new
DatagramPacket(new byte[1024], 1024);              while (true) {

                System.out.println("Waiting for a multicast message...");
mcSocket.receive(packet);

                String msg = new String(packet.getData(),packet.getOffset(),
packet.getLength());

                System.out.println("[Multicast Receiver] Received:" + msg); }
mcSocket.leaveGroup(mcIPAddress);

        } catch (Exception e) {  e.printStackTrace(); }}}
```

The following code contains a program that sends a message to the same multicast address.

You can run multiple instances of the UDPMulticastReceiver class and all of them will become a member of the same multicast group.

```
import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class UDPMultiCastSender {

    public static void main(String[] args) {

        int mcPort = 18777; String mcIPStr = "230.1.1.1";

        try( DatagramSocket udpSocket = new DatagramSocket();  ) {

            // Prepare a message

            InetAddress mcIPAddress = InetAddress.getByName(mcIPStr);              byte[]
msg = "Hello multicast socket".getBytes();
```

```
        DatagramPacket packet = new DatagramPacket(msg, msg.length);

        packet.setAddress(mcIPAddress);

        packet.setPort(mcPort); /*  ww    w.  d   e  m   o 2 s    . c o   m */

        udpSocket.send(packet); System.out.println("Sent a multicast message.");
System.out.println("Exiting application");

    } catch (Exception e) { e.printStackTrace(); }}}
```