# INTRODUCTION TO JAVA SOCKET PROGRAMMING

## Using TELNET Client to Communicate with NIST Time Code:

import java.io.IOException; import java.net.Socket; import java.util.Scanner;

public class hellosocketprogramming\_q1 {

public static void main(String args[]) throws IOException { Socket s = new Socket ("time.nist.gov", 13); Scanner ins = new Scanner (s.getInputStream()); while (ins.hasNextLine()) {

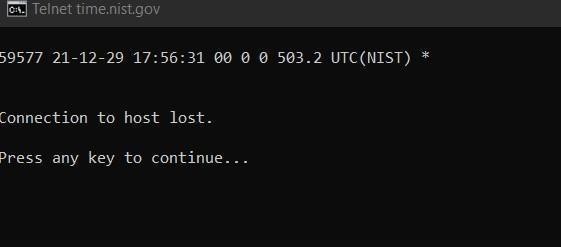
System.out.println (ins.nextLine());

};

}

}

## Output:



1. **Using Socket API to Implement Java Client to Communicate with NIST Time of the Day Service Code:**

import java.io.IOException; import java.net.Socket; import java.util.Scanner;

public class hellosocketprogramming\_q1 {

public static void main(String args[]) throws IOException { Socket s = new Socket ("time.nist.gov", 13); Scanner ins = new Scanner (s.getInputStream()); while (ins.hasNextLine()) {

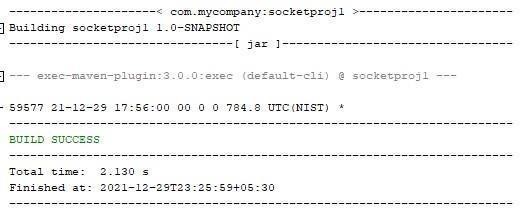
System.out.println (ins.nextLine());

};

}

}

## Output:



1. **Implementing a Java Program to Convert Between Domain Name and IP Address. Code:**

import java.net.InetAddress; import java.io.IOException;

import java.net.UnknownHostException; import java.util.Scanner;

public class socketprogram\_q3 {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) throws UnknownHostException { if (args.length > 0){

String domainname = args[0];

InetAddress[] inet = InetAddress.getAllByName(domainname); for(InetAddress addr : inet){

System.out.println(addr);

}

}else{

}

System.out.println(InetAddress.getLocalHost());

// TODO code application logic here

}

}

**Output:**



# IMPLEMENTING CLIENT-SERVER COMMUNICATION

1. Implementing a Simple Echo Chat Server Using Java Server Socket API

## Code:

import java.io.\*; import java.net.\*; import java.util.\*;

public class EchoServer {

public static void main(String args[]) throws IOException { ServerSocket ss = new ServerSocket (8189); System.out.println("I am about to listen on 8189"); Socket conn = ss.accept();

InputStream inS = conn.getInputStream(); OutputStream outS = conn.getOutputStream(); Scanner in = new Scanner(inS);

PrintWriter out = new PrintWriter(outS, true);

out.println("Hello!.. I am the chat server. Let's Chat"+" Say BYE to disconnect");

boolean bye = false;

while(!bye && in.hasNextLine()){ String cMsg = in.nextLine(); out.println("Echo: " +cMsg);

if (cMsg.trim().equals("BYE")) { bye= true;}

}

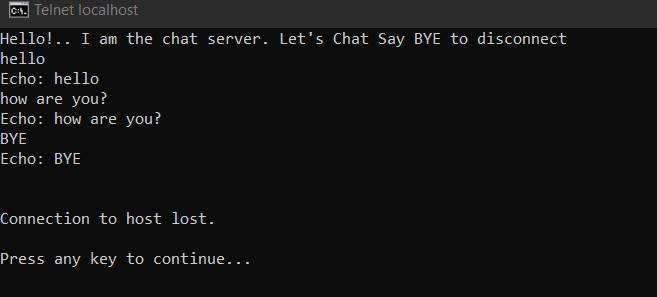
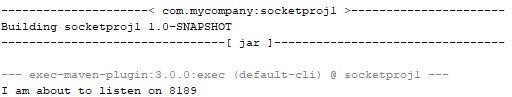
conn.close();

ss.close();

}

}

## Output:



1. **Making the Echo Chat Server Multi-Threaded to Support Multiple Simultaneous Chatting Sessions Code:**

import java.io.IOException; import java.net.ServerSocket; import java.net.Socket;

public class MultiUserEchoChatServer {

public static void main(String[] args) throws IOException{ ServerSocket server = new ServerSocket (8189); System.out.println("I am about to listen on 8189"); int userCnt =0;

while (true){

Socket userSocket = server.accept(); userCnt++;

EchoChatHandler echoChatter = new EchoChatHandler(userSocket,

userCnt);

userCnt);

}

}

}

Thread userChatThread = new Thread(echoChatter); System.out.println("Spawing a new chatting thread for user " +

userChatThread.start();

## Code (Handler Class):

import java.io.IOException; import java.io.InputStream; import java.io.OutputStream; import java.io.PrintWriter; import java.net.Socket; import java.util.Scanner;

class EchoChatHandler implements Runnable { private final Socket userSocket;

private int userId;

public EchoChatHandler(Socket userSocket, int userCnt){ this.userSocket = userSocket;

this.userId = userId;

}

@Override

public void run(){ try{

InputStream inS = userSocket.getInputStream(); OutputStream outS = userSocket.getOutputStream(); Scanner in = new Scanner(inS);

PrintWriter out = new PrintWriter(outS, true); out.println("Hello user!" + userId + "I am the chat server. Let's

Chat"+" Say BYE to disconnect"); boolean bye = false;

while(!bye && in.hasNextLine()){

String cMsg = in.nextLine(); out.println("Echo: " +cMsg);

if (cMsg.trim().equals("BYE")) { bye= true;}

}

userSocket.close();

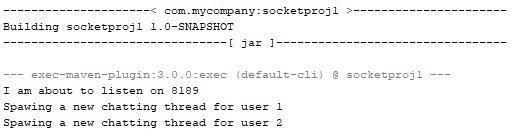
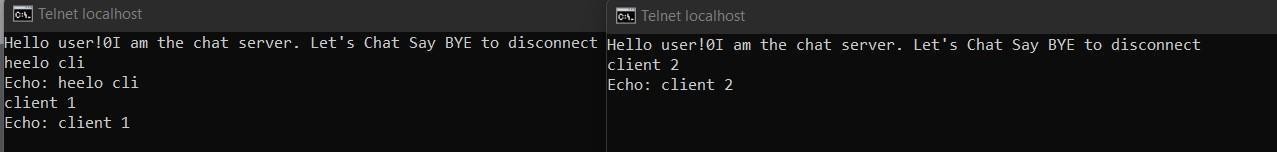
}

catch(IOException ioEX){ ioEX.printStackTrace();}

}

}

**Output:**



# INTRODUCTION TO DATAGRAM SOCKET API

## Using Datagram Socket API to Implement Number AdditionService Code (Client):

import java.io.DataInputStream; import java.io.DataOutputStream; import java.io.IOException; import java.io.InputStream; import java.io.OutputStream; import java.io.PrintWriter;

import java.net.DatagramPacket; import java.net.DatagramSocket; import java.net.InetAddress; import java.net.ServerSocket; import java.net.Socket;

import java.net.SocketException;

import java.net.UnknownHostException; import java.util.Scanner;

public class AdditionClient {

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

InetAddress addServiceIP = InetAddress.getLocalHost(); int addServicePort = 8189;

DatagramSocket clientSocket = new DatagramSocket(); Scanner sc = new Scanner(System.in); System.out.println("Please enter the list of number: "); String numberList = sc.nextLine();

DatagramPacket outDP = new DatagramPacket(numberList.getBytes (),numberList.length(),addServiceIP, addServicePort); clientSocket.send(outDP);

System.out.println("Adding the number " + numberList + " together");

byte[] buffer = new byte[256];

DatagramPacket inDP = new DatagramPacket(buffer, buffer.length); clientSocket.receive(inDP);

String servResp = new String(inDP.getData(),0,inDP.getLength()); System.out.println(servResp);

String stop = "STOP";

outDP = new DatagramPacket(stop.getBytes(),stop.length (),addServiceIP, addServicePort); clientSocket.send(outDP);

clientSocket.close();

}

catch(IOException ioEX){ System.out.println(ioEX);

}

}

}

## Code (Server):

import java.io.IOException; import java.net.DatagramPacket; import java.net.DatagramSocket; import java.net.InetAddress; import java.net.SocketException; import java.util.StringTokenizer; public class AdditionServer {

private int port = 8189;

public static void main(String[] args){

Datagramadditionserver addService = new Datagramadditionserver(); addService.start();

}

public void start(){ try{

DatagramSocket serverConn = new DatagramSocket(port); byte[] buffer = new byte[256];

DatagramPacket inDP = new DatagramPacket (buffer, buffer.length); String clientReq, serverResp;

do{

serverConn.receive(inDP);

InetAddress clientAddress = inDP.getAddress(); int clientPort = inDP.getPort();

clientReq = new String(inDP.getData(), 0, inDP.getLength()); if(clientReq != null && !clientReq.trim().equals("STOP")){

double sumResult = 0;

StringTokenizer st = new StringTokenizer(clientReq); try{

while(st.hasMoreTokens()){

Double d = new Double(st.nextToken()); sumResult += d.doubleValue();

}

serverResp = "The result is " + sumResult;

}

catch(NumberFormatException nEx){

serverResp = "Sorry, your list contains an invalid number";

}

DatagramPacket outDP = new DatagramPacket(serverResp.getBytes(),serverResp.length(),

clientAddress, clientPort); serverConn.send(outDP);

}

}

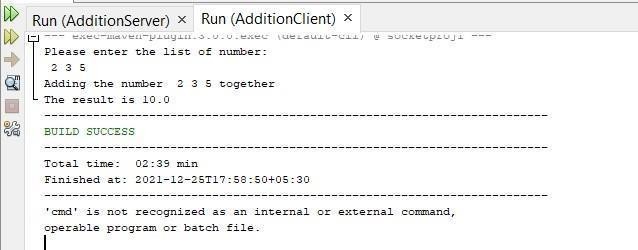
while(clientReq.trim().equals("STOP")); serverConn.close();

}catch(IOException ioex){ System.out.println(ioex);}

}

}

**Output:**



# IMPLEMENTING TOKEN-RING PROTOCOL FOR DISTRIBUTED

**MUTUAL-EXCLUSION**

## Using Datagram Socket API to Implement Token RingProtocol to Enforce Distributed Mutual Exclusion

**Code (Server):**

import java.io.IOException; import java.net.DatagramPacket; import java.net.DatagramSocket; import java.net.SocketException; public class TokenChatServer {

private static DatagramSocket ds; private static DatagramPacket dp;

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

ds = new DatagramSocket(1000);

}catch(SocketException ex){ ex.printStackTrace();} while(true){

byte buff[] = new byte[1024];

ds.receive(dp = new DatagramPacket (buff, buff.length)); String str = new String(dp.getData(),0,dp.getLength()); System.out.println("Message From: " + str);

}

}

}

## Code (Client1):

import java.io.BufferedReader; import java.io.IOException; import java.io.InputStreamReader; import java.net.DatagramPacket; import java.net.DatagramSocket; import java.net.InetAddress; import java.net.SocketException; import java.util.logging.Level; import java.util.logging.Logger; public class TokenClient1 {

private static DatagramSocket ds; private static DatagramPacket dp; private static BufferedReader br; private static int cp=100;

public static void main(String[] args) throws IOException{ boolean hasToken = true;

try{

ds= new DatagramSocket(cp);

}

catch(SocketException ex){ Logger.getLogger(TokenClient1.class.getName()).log(Level.SEVERE, null, ex);

throw ex;

}

while(true){

if(hasToken){

System.out.println("Do you want to say Something"+"(i.e.,Send Data)to Server?:"+"Type Y for Yes/N for No");

br=new BufferedReader(new InputStreamReader(System.in)); String userResp= br.readLine(); if(userResp.equalsIgnoreCase("Y")){

System.out.println("Enter what you want to send:"); String userData="Client 1===>> "+br.readLine(); System.out.println("Getting ready to send data ..."); byte buff[]=userData.getBytes(); System.out.println("Sending...");

ds.send(new DatagramPacket(buff,buff.length,

InetAddress.getLocalHost(),1000));

}else{

client 2.");

System.out.println("Data Sent.");

System.out.println("Since I am in "+"busy state ... passing token to String tokenMsg = "Token";

byte[] bf1 = new byte[1024]; bf1 = tokenMsg.getBytes(); ds.send(new

DatagramPacket(bf1,bf1.length,InetAddress.getLocalHost(),200));

hasToken= false;

is"+msgClient3);

}else{

}

System.out.println("Entering in the receiving mode ..."); byte bf[]=new byte[1024];

ds.receive(dp = new DatagramPacket(bf,bf.length));

String msgClient3= new String(dp.getData(),0,dp.getLength()); System.out.println("The data received from left neighbor :"+"client 3

if(msgClient3.equalsIgnoreCase("Token")){ hasToken = true;

}

}

}

}

}

## Code (Client2):

import java.io.BufferedReader; import java.io.IOException; import java.io.InputStreamReader;

import java.net.DatagramPacket; import java.net.DatagramSocket; import java.net.InetAddress; import java.net.SocketException; import java.util.logging.Level; import java.util.logging.Logger; public class TokenClient2 {

private static DatagramSocket ds; private static DatagramPacket dp; private static BufferedReader br; private static int cp = 200;

public static void main(String[] args) throws IOException{ boolean hasToken = false;

try{

ds = new DatagramSocket(cp);

}

catch(SocketException ex){ Logger.getLogger(TokenClient2.class.getName()).log(Level.SEVERE, null,ex); throw ex;

}

while(true){

if(hasToken){

System.out.println("Do you want to say something, " + "(i.e. Send Data) To Server?: " + "Type Y for Yes/N for No");

br = new BufferedReader(new InputStreamReader(System.in)); String userResp = br.readLine(); if(userResp.equalsIgnoreCase("Y")){

System.out.println("Enter what you want to send: "); String userData = "Client 2 ===> "+br.readLine(); System.out.println("Getting ready to send data ..."); byte buff[] = userData.getBytes(); System.out.println("Something...");

ds.send(new DatagramPacket (buff,buff.length,

InetAddress.getLocalHost(),1000));

}else{

client 3.");

System.out.println("Data Sent.");

System.out.println("Since I am in "+"busy state ... passing token to String tokenMsg = "Token";

byte[] bf1 = new byte[1024]; bf1 = tokenMsg.getBytes(); ds.send(new

DatagramPacket(bf1,bf1.length,InetAddress.getLocalHost(),300));

hasToken= false;

}

is"+msgClient3);

}else{

System.out.println("Entering in the receiving mode ..."); byte bf[]=new byte[1024];

ds.receive(dp = new DatagramPacket(bf,bf.length));

String msgClient3= new String(dp.getData(),0,dp.getLength()); System.out.println("The data received from left neighbor :"+"client 1

if(msgClient3.equalsIgnoreCase("Token")){ hasToken = true;

}

}

}

}

}

## Code (Client3):

import java.io.BufferedReader; import java.io.IOException; import java.io.InputStreamReader; import java.net.DatagramPacket; import java.net.DatagramSocket; import java.net.InetAddress; import java.net.SocketException; import java.util.logging.Level; import java.util.logging.Logger; public class TokenClient3 {

private static DatagramSocket ds; private static DatagramPacket dp; private static BufferedReader br; private static int cp = 300;

public static void main(String[] args) throws IOException{ boolean hasToken = false;

try{

ds = new DatagramSocket(cp);

}

catch(SocketException ex){ Logger.getLogger(TokenClient3.class.getName()).log(Level.SEVERE, null,ex); throw ex;

}

while(true){

if(hasToken){

System.out.println("Do you want to say something, " + "(i.e. Send Data) To Server?: " + "Type Y for Yes/N for No");

br = new BufferedReader(new InputStreamReader(System.in)); String userResp = br.readLine();

if(userResp.equalsIgnoreCase("Y")){ System.out.println("Enter what you want to send: "); String userData = "Client 3 ===> "+br.readLine(); System.out.println("Getting ready to send data ..."); byte buff[] = userData.getBytes(); System.out.println("Something...");

ds.send(new DatagramPacket (buff,buff.length,

InetAddress.getLocalHost(),1000));

}else{

client 2.");

System.out.println("Data Sent.");

System.out.println("Since I am in "+"busy state ... passing token to String tokenMsg = "Token";

byte[] bf1 = new byte[1024]; bf1 = tokenMsg.getBytes(); ds.send(new

DatagramPacket(bf1,bf1.length,InetAddress.getLocalHost(),100));

hasToken= false;

is"+msgClient3);

}else{

}

System.out.println("Entering in the receiving mode ..."); byte bf[]=new byte[1024];

ds.receive(dp = new DatagramPacket(bf,bf.length));

String msgClient3= new String(dp.getData(),0,dp.getLength()); System.out.println("The data received from left neighbor :"+"client 2

if(msgClient3.equalsIgnoreCase("Token"))

{

hasToken = true;

}

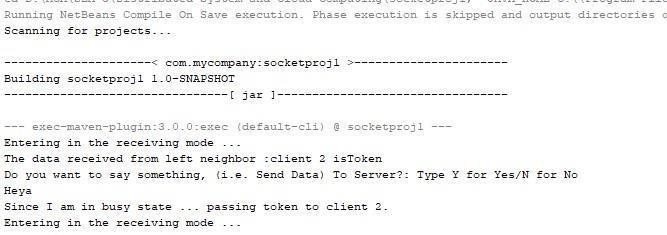
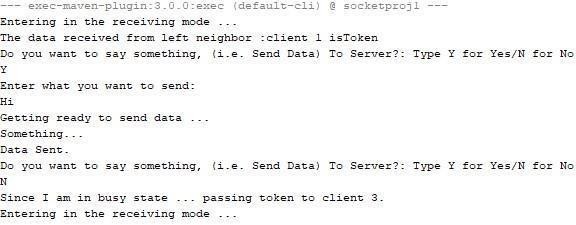
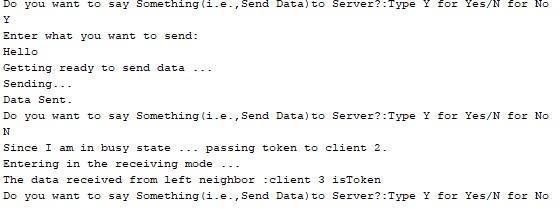
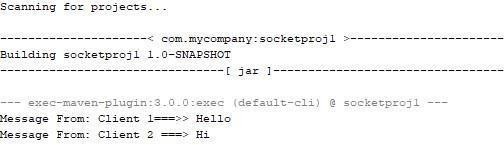
}

}

}

}

**Output:**



# Implement a simple client server chat application using Socket API

## Code (Client):

import java.io.DataInputStream; import java.io.DataOutputStream; import java.io.IOException; import static java.lang.System.out; import java.net.Socket;

import java.net.UnknownHostException; public class Client\_Q1 {

private Socket socket=null;

private DataInputStream input =null; private DataOutputStream output =null;

public Client\_Q1(String address, int port) throws IOException{ try{

socket=new Socket(address, port); System.out.println("Connected"); input=new DataInputStream(System.in);

output=new DataOutputStream(socket.getOutputStream());

}

catch(UnknownHostException u){ System.out.println(u);

}

catch(IOException i){

System.out.println(i);

}

String line=""; while(!line.equals("Over")){

try{

line=input.readLine(); output.writeUTF(line);

}

}

try{

}

catch(IOException i){ System.out.println(i);

}

input.close(); output.close(); socket.close();

catch(IOException i){

System.out.println(i);

}

}

public static void main(String args[]) throws IOException{ Client\_Q1 client=new Client\_Q1("127.0.0.1",5000);

}

}

## Code (Server):

import com.sun.corba.se.spi.activation.Server; import java.io.BufferedInputStream;

import java.io.DataInputStream; import java.io.IOException; import java.net.ServerSocket; import java.net.Socket;

public class Server\_Q1 {

private Socket socket = null; private ServerSocket server = null; private DataInputStream in = null;

public Server\_Q1(int port) throws IOException{ try{

server = new ServerSocket(port); System.out.println("Server started"); System.out.println("waiting for client"); socket = server.accept(); System.out.println("Client Accepted");

in = new DataInputStream(new BufferedInputStream(socket.getInputStream())); String line = "";

while(!line.equals("Over")){ try{

line = in.readUTF(); System.out.println(line);

}

catch(IOException i){

System.out.println(i);

}

}

System.out.println("Closing connection"); socket.close();

in.close();

}

catch(IOException i){

System.out.println(i);

}

}

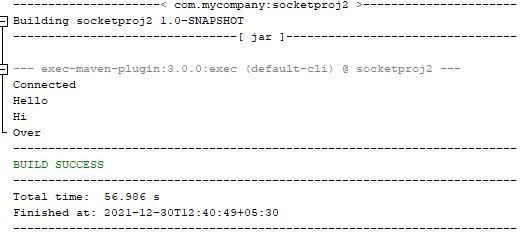
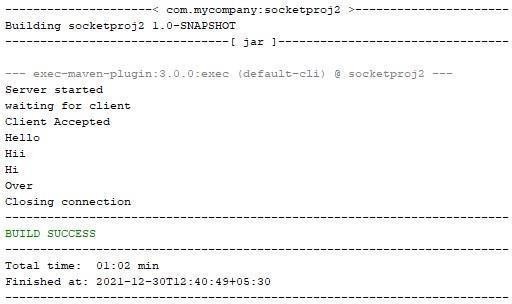
public static void main(String args[]) throws IOException{

Server\_Q1 server = new Server\_Q1(5000);

}

}

**Output:**



# To develop a program for multi-client chat server.

**Code (Client):** import java.io.\*; import java.net.\*;

import java.util.Scanner; public class MultiChatClient

{

final static int ServerPort = 1234;

public static void main(String args[]) throws UnknownHostException, IOException

{

Scanner scn = new Scanner(System.in);

// getting localhost ip

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

// establish the connection

Socket s = new Socket(ip, ServerPort);

// obtaining input and out streams

DataInputStream dis = new DataInputStream(s.getInputStream()); DataOutputStream dos = new DataOutputStream(s.getOutputStream());

// sendMessage thread

Thread sendMessage = new Thread(new Runnable()

{

@Override

public void run() {

while (true) {

// read the message to deliver. String msg = scn.nextLine(); try {

dos.writeUTF(msg);

// write on the output stream

}

catch (IOException e) {

e.printStackTrace();

}

}

}

});

// readMessage thread

Thread readMessage = new Thread(new Runnable()

{

@Override

public void run() {

while (true) {

try {

// read the message sent to this client String msg = dis.readUTF(); System.out.println(msg);

}

}

}

});

}

sendMessage.start(); readMessage.start();

}

}

catch (IOException e) {

e.printStackTrace();

## Code (Server):

import java.io.\*; import java.util.\*; import java.net.\*;

public class MultiChatServer

{

// Vector to store active clients

static Vector<ClientHandler> ar = new Vector<>();

// counter for clients static int i = 0;

public static void main(String[] args) throws IOException

{

// server is listening on port 1234 ServerSocket ss = new ServerSocket(1234); Socket s;

// running infinite loop for getting

// client request while (true)

{

// Accept the incoming request s = ss.accept();

System.out.println("New client request received : " + s);

// obtain input and output streams

DataInputStream dis = new DataInputStream(s.getInputStream()); DataOutputStream dos = new DataOutputStream(s.getOutputStream()); System.out.println("Creating a new handler for this client...");

// Create a new handler object for handling this request. ClientHandler mtch = new ClientHandler(s,"client " + i, dis, dos);

// Create a new Thread with this object. Thread t = new Thread(mtch);

System.out.println("Adding this client to active client list");

// add this client to active clients list ar.add(mtch);

// start the thread.

t.start();

// increment i for new client.

// i is used for naming only, and can be replaced

// by any naming scheme i++;

}

}

}

// ClientHandler class

class ClientHandler implements Runnable { Scanner scn = new Scanner(System.in); private String name;

final DataInputStream dis; final DataOutputStream dos; Socket s;

boolean isloggedin;

// constructor

public ClientHandler(Socket s, String name,DataInputStream dis,DataOutputStream dos) { this.dis = dis;

this.dos = dos; this.name = name; this.s = s; this.isloggedin=true;

}

@Override

public void run() {

String received; while (true)

{

try

{

// receive the string received = dis.readUTF();

System.out.println(received); if(received.equals("logout")){

this.isloggedin=false; this.s.close();

break;

}

// break the string into message and recipient part StringTokenizer st = new StringTokenizer(received, "#"); String MsgToSend = st.nextToken();

String recipient = st.nextToken();

// search for the recipient in the connected devices list.

// ar is the vector storing client of active users

for (ClientHandler mc : MultiChatServer.ar)

{

// if the recipient is found, write on its

// output stream

if (mc.name.equals(recipient) && mc.isloggedin==true)

{

mc.dos.writeUTF(this.name+" : "+MsgToSend); break;

}

}

}

catch (IOException e) {

e.printStackTrace();

}

}

try

{

}

// closing resources this.dis.close(); this.dos.close();

catch(IOException e){

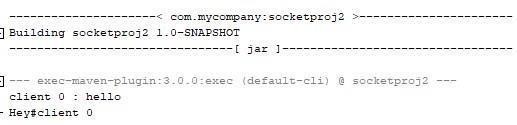
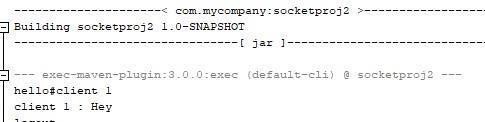
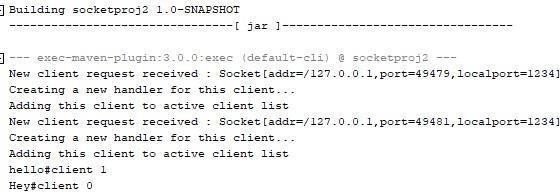
e.printStackTrace();

}

}

}

**Output:**



# To implement a Server calculator using RPC concept. (Make use of datagram)

## Code (Client):

import java.io.IOException; import java.net.DatagramPacket; import java.net.DatagramSocket; import java.net.InetAddress; import java.util.Scanner;

public class Client {

public static void main(String[] args)throws IOException { Scanner sc = new Scanner(System.in); DatagramSocket ds= new DatagramSocket(); InetAddress ip=InetAddress.getLocalHost();

byte buf[]=null; while(true){

System.out.println("Enter the equaion in the format:"); System.out.println("'Operand1 And operand2'"); String inp=sc.nextLine();

buf=new byte[65535]; buf=inp.getBytes();

DatagramPacket DpSend = new DatagramPacket(buf,buf.length, ip, 1234); ds.send(DpSend);

if(inp.equals("Exit")) break;

buf=new byte[65535];

DatagramPacket DpReceive = new DatagramPacket(buf, buf.length); ds.receive(DpReceive);

System.out.println("Answer="+ new String(buf,0, buf.length));

}

}

}

## Code (Server):

import java.io.IOException; import java.net.DatagramPacket; import java.net.DatagramSocket; import java.net.InetAddress; import java.util.StringTokenizer; public class Server {

public static void main(String args[]) throws IOException{ DatagramSocket ds=new DatagramSocket(1234); byte buf[]=null ;

DatagramPacket DpSend=null; DatagramPacket DpReceive=null; while(true){

buf=new byte[65535];

DpReceive=new DatagramPacket(buf,buf.length);

ds.receive(DpReceive);

String inp=new String(buf, 0, buf.length); inp=inp.trim();

System.out.println("Equaion Received:-"+inp); if(inp.equals("Exit")){

System.out.println("client Exiting"); break;

}

int result;

StringTokenizer st=new StringTokenizer(inp); int op1=Integer.parseInt(st.nextToken()); String operation=st.nextToken();

int op2=Integer.parseInt(st.nextToken()); if(operation.equals("+"))

result=op1+op2;

else if(operation.equals("-")) result=op1-op2;

else if(operation.equals("\*")) result=op1\*op2;

else

result=op1/op2;

System.out.println("Sending the result..."); String res=Integer.toString(result); buf=res.getBytes();

int port=DpReceive.getPort();

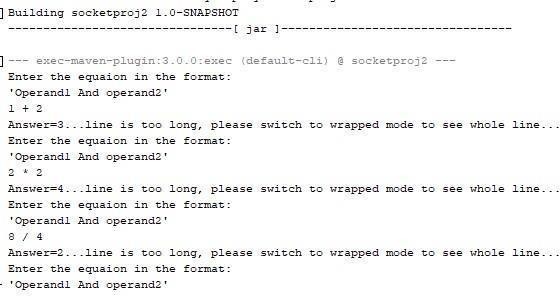
DpSend =new DatagramPacket(buf, buf.length, InetAddress.getLocalHost(),port); ds.send(DpSend);

}

}

}

## Output:



**Code (Client):**

# To implement a Date Time Server using RPC concept.

import java.io.IOException; import java.net.DatagramPacket; import java.net.DatagramSocket; import java.net.InetAddress; import java.util.\*;

public class UDP\_Client {

public static void main(String[] args) throws IOException { Scanner sc = new Scanner(System.in); DatagramSocket ds = new DatagramSocket(); InetAddress ip= InetAddress.getLocalHost();

byte buf[] = null; while(true){

System.out.println("What do you want to know? Date / Time"); String inp = sc.nextLine();

buf = new byte[65535]; buf = inp.getBytes();

DatagramPacket DpSend = new DatagramPacket(buf, buf.length, ip, 1234); ds.send(DpSend);

if(inp.equals("BYE")) break;

buf = new byte[65535];

DatagramPacket DpReceive = new DatagramPacket(buf,0, buf.length); ds.receive(DpReceive);

System.out.println(inp+ ":" +new String(buf,0, buf.length));

}

}

}

## Code (Server):

import java.io.IOException; import java.net.DatagramPacket; import java.net.DatagramSocket; import java.net.InetAddress; import java.util.\*;

import java.text.\*;

public class UDP\_Server {

public static void main(String[] args) throws IOException { DatagramSocket ds = new DatagramSocket(1234);

System.out.println("the chat server is Listenting on the port 1234. "); DateFormat forDate = new SimpleDateFormat("yyyy/mm/dd"); DateFormat forTime = new SimpleDateFormat("hh:mm:ss");

byte buf[]= null; DatagramPacket DpSend = null;

DatagramPacket DpReceive = null; while(true){

String toReturn = ""; buf = new byte[65535];

DpReceive = new DatagramPacket(buf, buf.length); ds.receive(DpReceive);

String inp = new String (buf,0 ,buf.length); inp = inp.trim();

if(inp.equals("BYE")){

System.out.println("Client is saying Bye... exiting"); break;

}

Date date = new Date(); if(inp.equals("Date"))

toReturn = forDate.format(date); else if(inp.equals("Time"))

toReturn = forTime.format(date); System.out.println("Sending result ..."); buf = toReturn.getBytes();

int port = DpReceive.getPort();

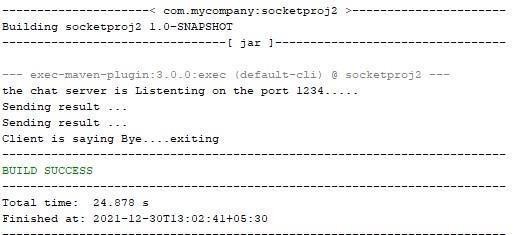
DpSend = new DatagramPacket(buf, buf.length, InetAddress.getLocalHost(),port); ds.send(DpSend);

}

}

}

**Output:**



# Using MySQL create Library database. Create table Book (Book\_id,

**Book\_name, Book\_author) and retrieve the Book information from Library database using Remote Object Communication concept.**

## Code (Client):

import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry; import java.util.\*;

public class Client {

private Client() {}

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

// Getting the registry

Registry registry = LocateRegistry.getRegistry(null);

// Looking up the registry for the remote object Hello stub = (Hello) registry.lookup("Hello");

// Calling the remote method using the obtained object @SuppressWarnings("unchecked")

List<Student> list = (List)stub.getStudents(); for (Student s:list) {

// System.out.println("bc "+s.getBranch()); System.out.println("ID: " + s.getId()); System.out.println("name: " + s.getName()); System.out.println("branch: " + s.getBranch()); System.out.println("percent: " + s.getPercent()); System.out.println("email: " + s.getEmail());

}

// System.out.println(list);

} catch (Exception e) {

System.err.println("Client exception: " + e.toString()); e.printStackTrace();

}

}

}

## Code (Server):

import java.rmi.registry.Registry; import java.rmi.registry.LocateRegistry; import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject; public class Server extends ImplExample {

public Server() {}

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

object to the stub)

// Instantiating the implementation class ImplExample obj = new ImplExample();

// Exporting the object of implementation class (here we are exporting the remote

Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);

// Binding the remote object (stub) in the registry Registry registry = LocateRegistry.getRegistry(); registry.bind("Hello", stub); System.err.println("Server ready");

} catch (Exception e) {

System.err.println("Server exception: " + e.toString()); e.printStackTrace();

}

}

}

## Code (Hello):

import java.rmi.Remote;

import java.rmi.RemoteException; import java.util.\*;

// Creating Remote interface for our application public interface Hello extends Remote {

public List<Student> getStudents() throws Exception; }

## Code (ImplExample):

import java.sql.\*; import java.util.\*;

// Implementing the remote interface

public class ImplExample implements Hello {

// Implementing the interface method

`public List<Student> getStudents() throws Exception { List<Student> list = new ArrayList<Student>();

// JDBC driver name and database URL

String JDBC\_DRIVER = "com.mysql.jdbc.Driver"; String DB\_URL = "jdbc:mysql://localhost:3306/details";

// Database credentials String USER = "root"; String PASS = "admin";

Connection conn = null; Statement stmt = null;

//Register JDBC driver Class.forName("com.mysql.jdbc.Driver");

//Open a connection

System.out.println("Connecting to a selected database...");

conn = DriverManager.getConnection(DB\_URL, USER, PASS); System.out.println("Connected database successfully...");

//Execute a query System.out.println("Creating statement..."); stmt = conn.createStatement();

String sql = "SELECT \* FROM student\_data"; ResultSet rs = stmt.executeQuery(sql);

//Extract data from result set while(rs.next()) {

// Retrieve by column name int id = rs.getInt("id");

String name = rs.getString("name"); String branch = rs.getString("branch"); int percent = rs.getInt("percentage"); String email = rs.getString("email");

// Setting the values

Student student = new Student(); student.setID(id); student.setName(name); student.setBranch(branch); student.setPercent(percent); student.setEmail(email); list.add(student);

}

rs.close(); return list;

}

}

## Code (Student):

public class Student implements java.io.Serializable { private int id, percent;

private String name, branch, email; public int getId() {

return id;

}

public String getName() { return name;

}

public String getBranch() { return branch;

}

public int getPercent() {

return percent;

}

public String getEmail() { return email;

}

public void setID(int id) { this.id = id;

}

public void setName(String name) { this.name = name;

}

public void setBranch(String branch) { this.branch = branch;

}

public void setPercent(int percent) { this.percent = percent;

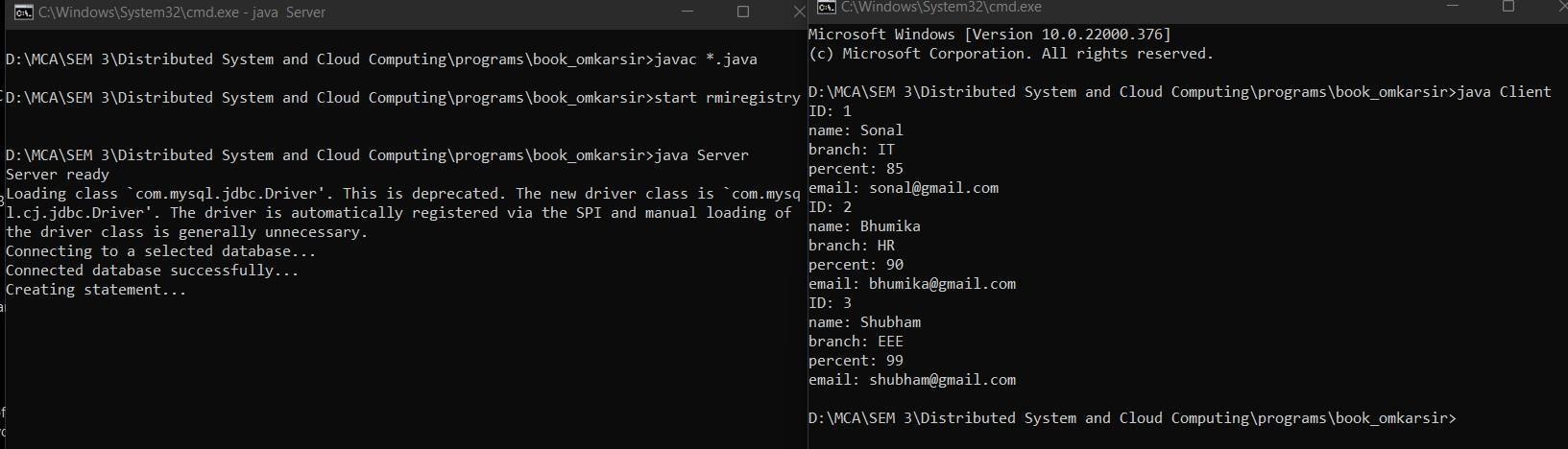
}

public void setEmail(String email) { this.email = email;

}

}

**Output:**



# Using MySQL create Electric\_Bill database. Create tableBill (consumer\_name, bill\_due\_date, bill\_amount) and retrieve the Bill information from the Elecrtic\_Bill database using Remote Object

**Communication concept.**

## Code (Client):

import java.rmi.registry.LocateRegistry; import java.rmi.registry.Registry; import java.util.\*;

public class Client {

private Client() {}

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

// Getting the registry

Registry registry = LocateRegistry.getRegistry(null);

// Looking up the registry for the remote object Hello stub = (Hello) registry.lookup("Hello");

// Calling the remote method using the obtained object @SuppressWarnings("unchecked")

List<Student> list = (List)stub.getStudents(); for (Student s:list) {

// System.out.println("bc "+s.getBranch()); System.out.println("ID: " + s.getId()); System.out.println("name: " + s.getName()); System.out.println("branch: " + s.getBranch());

//System.out.println("percent: " + s.getPercent());

//System.out.println("email: " + s.getEmail());

}

// System.out.println(list);

} catch (Exception e) {

System.err.println("Client exception: " + e.toString()); e.printStackTrace();

}

}

}

## Code (Server):

import java.rmi.registry.Registry; import java.rmi.registry.LocateRegistry; import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject; public class Server extends ImplExample {

public Server() {}

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

object to the stub)

// Instantiating the implementation class ImplExample obj = new ImplExample();

// Exporting the object of implementation class (here we are exporting the remote

Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);

// Binding the remote object (stub) in the registry Registry registry = LocateRegistry.getRegistry(); registry.bind("Hello", stub); System.err.println("Server ready");

} catch (Exception e) {

System.err.println("Server exception: " + e.toString()); e.printStackTrace();

}

}

}

## Code (Hello):

import java.rmi.Remote;

import java.rmi.RemoteException; import java.util.\*;

// Creating Remote interface for our application public interface Hello extends Remote {

public List<Student> getStudents() throws Exception; }

## Code (Student):

public class Student implements java.io.Serializable { private int id, percent;

private String name, branch, email; public int getId() {

return id;

}

public String getName() { return name;

}

public String getBranch() { return branch;

}

public int getPercent() {

return percent;

}

public String getEmail() { return email;

}

public void setID(int id) { this.id = id;

}

public void setName(String name) { this.name = name;

}

public void setBranch(String branch) { this.branch = branch;

}

public void setPercent(int percent) { this.percent = percent;

}

public void setEmail(String email) { this.email = email;

}

}

## Code (ImplExample):

import java.sql.\*; import java.util.\*;

// Implementing the remote interface

public class ImplExample implements Hello {

// Implementing the interface method

public List<Student> getStudents() throws Exception { List<Student> list = new ArrayList<Student>();

// JDBC driver name and database URL

String JDBC\_DRIVER = "com.mysql.jdbc.Driver"; String DB\_URL = "jdbc:mysql://localhost:3306/details1";

// Database credentials String USER = "root"; String PASS = "admin";

Connection conn = null; Statement stmt = null;

//Register JDBC driver Class.forName("com.mysql.jdbc.Driver");

//Open a connection

System.out.println("Connecting to a selected database...");

conn = DriverManager.getConnection(DB\_URL, USER, PASS); System.out.println("Connected database successfully...");

//Execute a query System.out.println("Creating statement...");

stmt = conn.createStatement();

String sql = "SELECT \* FROM bill"; ResultSet rs = stmt.executeQuery(sql);

//Extract data from result set while(rs.next()) {

// Retrieve by column name

int id = rs.getInt("bill\_amount");

String name = rs.getString("consumer\_name"); String branch = rs.getString("bill\_due\_date");

//int percent = rs.getInt("percentage");

//String email = rs.getString("email");

// Setting the values

Student student = new Student(); student.setID(id); student.setName(name); student.setBranch(branch);

//student.setPercent(percent);

//student.setEmail(email); list.add(student);

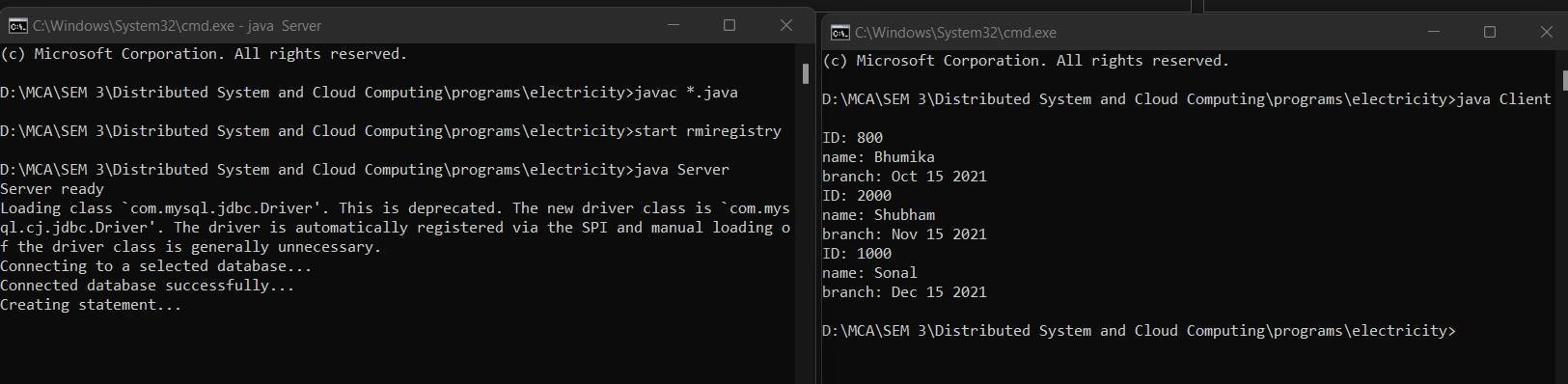
}

rs.close(); return list;

}

}

**Output:**



# To develop applications using Google App Engine by using Eclipse IDE.

## Steps:

1. Open any browser of your choice, go to cloud.google.com, login to your google account.
2. At the Top Right hand side click on Console, Click on Current Project -> Click on New Project.
3. Under Project Name -> Enter name of application -> Click on Create.
4. When the project is created, At top Right There is a icon of “Activate Cloud Shell” ->Click on it.
5. When the terminal is opened, Enter command “mkdir symca”.
6. Click on Open Editor -> Click on New File - > Enter file name “demo.java” -> Then select our directory “symca” -> Click on OK.
7. When the file is created -> Enter the following code : public class demo{

public static void main(String[] args) { System.out.println("Atharva Kale C22059 NMITD");

}

}

1. Save the file “demo.java” -> Then click on Open Terminal
2. Enter command “cd symca” and press Enter
3. Enter command “javac demo.java” and press enter
4. Enter command “java demo” and press enter

## Output:

