# Search.java

# import java.rmi.\*;

# public interface Search extends Remote {

# // Declaring the method prototype

# public String query(String search) throws

# RemoteException;

# }

# SearchQuery.java

# import java.rmi.\*;

# import java.rmi.server.\*;

# public class SearchQuery extends RemoteObject

# implements Search

# {

# // Implementation of the query interface

# public String query(String search)

# throws RemoteException

# {

# String result;

# if (search.equals("Reflection in Java"))

# result = "Found";

# else

# result = "Not Found";

# return result; } }

# Server.java

# import java.net.\*;

# import java.io.\*;

# public class Server

# {

# //initialize socket and input stream

# private Socket socket = null;

# private ServerSocket server = null;

# private DataInputStream in = null;

# // constructor with port

# public Server(int port)

# {

# // starts server and waits for a connection

# try

# {

# server = new ServerSocket(port);

# System.out.println("Server started");

# System.out.println("Waiting for a client ...");

# socket = server.accept();

# System.out.println("Client accepted");

# // takes input from the client socket

# in = new DataInputStream(

# new BufferedInputStream(socket.getInputStream()));

# String line = "";

# // reads message from client until "Over" is sent

# while (!line.equals("Over"))

# {

# try

# {

# line = in.readUTF();

# System.out.println(line);

# }

# catch(IOException i)

# {

# System.out.println(i);

# }

# }

# System.out.println("Closing connection");

# // close connection

# socket.close();

# in.close();

# }

# catch(IOException i)

# {

# System.out.println(i);

# }

# }

# public static void main(String args[])

# {

# Server server = new Server(5000);

# }

# }

# Client.java

# import java.io.\*;

# import java.net.\*;

# public class Client {

# // initialize socket and input output streams

# private Socket socket = null;

# private BufferedReader d = null;

# private InputStream input = null;

# private DataOutputStream out = null;

# // constructor to put ip address and port

# public Client(String address, int port)

# {

# // establish a connection

# try {

# socket = new Socket(address, port);

# System.out.println("Connected");

# System.out.println("Done with 1st program of DS");

# // takes input from terminal

# d = new BufferedReader(new InputStreamReader(System.in));

# // sends output to the socket

# out = new DataOutputStream(

# socket.getOutputStream());

# }

# catch (UnknownHostException u) {

# System.out.println(u);

# return;

# }

# catch (IOException i) {

# System.out.println(i);

# return;

# }

# // string to read message from input

# String line = "";

# // keep reading until "Over" is input

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

# try {

# line = d.readLine();

# out.writeUTF(line);

# }

# catch (IOException i) {

# System.out.println(i);

# }

# }

# // close the connection

# try {

# input.close();

# out.close();

# socket.close();

# }

# catch (IOException i) {

# System.out.println(i);

# }

# }

# public static void main(String args[])

# {

# Client client = new Client("127.0.0.1", 5000);

# }

# }

# Step 7: Compile and execute application programs

# On console-1:

# #javac Search.java

# #javac SearchQuery.java

# #rmic SearchQuery

# #rmiregistry on console

# On console2:

# Compile Server Application:

# #javac Server.java

# #java Server

# On console-3:

# Compile Client Application:

# #javac Client.java

# #java Client