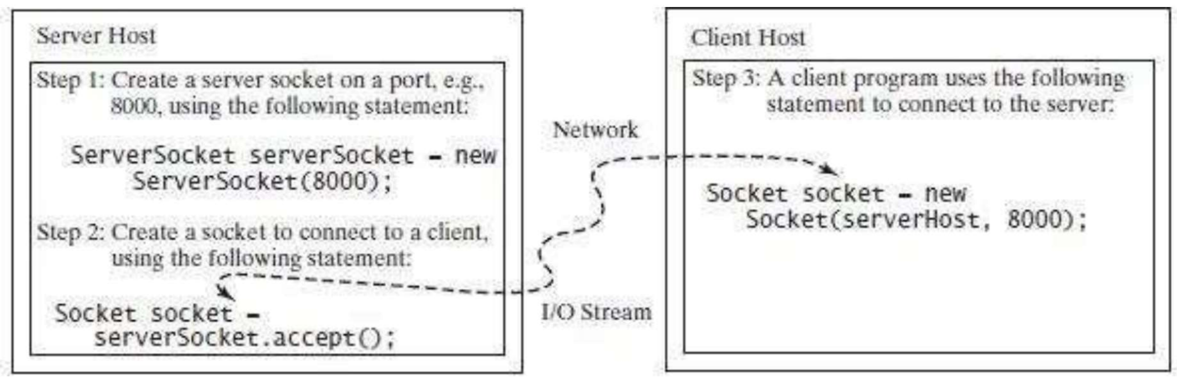
North South University
CSE 438L: Data Communication & Network Lab

Lab Class 4: Introduction to Socket Programing

Socket Programing

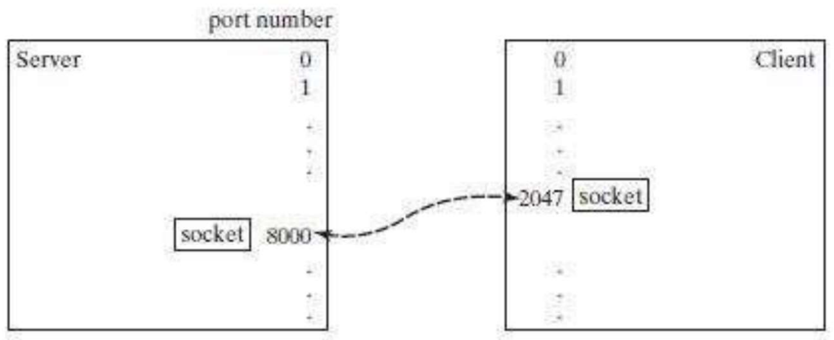
Networking is tightly integrated in Java. Java API provides the classes for creating sockets to facilitate program communications over the Internet. Sockets are the endpoints of logical connections between two hosts and can be used to send and receive data. Java treats socket communications much as it treats I/O operations; thus programs can read from or write to sockets as easily as they can read from or write to files. Java supports both TCP and UDP for communication.

To establish a server, you need to create a server socket and attach it to a port, which is where the server listens for connections.


Server Host
Step 1: Create a server socket on a port, e.g., 8000, using the following statement:
ServerSocket serverSocket = new ServerSocket(8000);
Step 2: Create a socket to connect to a client, using the following statement:
Socket socket = serverSocket.accept();
Network
I/O Stream
Client Host
Step 3: A client program uses the following statement to connect to the server:
Socket socket = new Socket(serverHost, 8000);

Note:

When you create a server socket, you have to specify a port (e.g., 8000) for the socket. When a client connects to the server (line 43 in Client.java), a socket is created on the client. This socket has its own local port. This port number (e.g., 2047) is automatically chosen by the JVM.



Exercise 1: Create a simple Server

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class Server1 {

        private static ServerSocket server = null; private static
        Socket socket = null; private static final int port =
        8080;

        public static void main(String[] args) {

                //Create IO Objects
                BufferedReader in = null;
                PrintWriter out = null;
                Scanner consoleInput = new Scanner(System.in);

                //Start Server
                try {
                        System.out.println("Server is starting ...");
                        server = new ServerSocket(port);
                        System.out.println("Server has started");
                } catch (IOException e) {
                        System.out.println("Can not listen to port: " + port); System.exit(-1);

                }
                while(true) {
                        //Create Socket
                        try {
                                socket = server.accept();
                                System.out.println("Client has been connected\n"); } catch
                        (IOException e) {
                                System.out.println("Communication Error with
client");
                                System.exit(-1);
                        }


                        try {
                                in = new BufferedReader(
                                        new InputStreamReader(
                                                socket.getInputStream()
                                        )
                                );

                                out = new PrintWriter(socket.getOutputStream(),
true);
```

```
                    out.println("NSU CSE338 LAB Server");
                    System.out.println("Client Name: " + in.readLine());

                    while(socket.isConnected()) {
                        System.out.print("Server: ");
                        out.println(consoleInput.nextLine());
                        System.out.print("Client: ");
                        System.out.println(in.readLine());
                    }

                } catch (IOException e) {
                    System.out.print("Client Left");
                    consoleInput.close();
                }
            }

        }


}
```

Exercise 2: Create a Client

```
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class Client1 {

        public static Socket socket = null;

        public static void main(String[] args) {

                try {
                        socket = new Socket("localhost", 8080);
                        System.out.println("Connected to Server\n"
                                                    + "Socket: " + socket.getInetAddress() + ":" +
socket.getPort() + "\n" );
                } catch (IOException e) {
                        System.out.print("Connection to network can not be
stablished");
                }

                BufferedReader in = null;
                PrintWriter out = null;
                Scanner consoleInput = new Scanner(System.in);

                try {
```

```
                              in = new BufferedReader( new InputStreamReader(
socket.getInputStream() ));
                              out = new PrintWriter(socket.getOutputStream(), true);

                              System.out.println("Server: " + in.readLine());
                              System.out.print("Client: your name_ ");
                              out.println(consoleInput.nextLine());

                              while(true) {
                                        System.out.print("Server: ");
                                        System.out.println(in.readLine());
                                        System.out.print("Client: ");
                                        out.println(consoleInput.nextLine());
                              }


                    } catch (IOException e) {

                              e.printStackTrace();

                    }
          }
}
```

Task:

Perform all three exercises in the class.

Assignment - 1:

Write a socket server program named RockPaperScissorsServer, which communicates with players (clients). Server firstly receives a name for the player. Let's assume 1 is Rock, 2 is Paper and 3 is Scissors. Next Server receives any of the 3 number from the user, generate a random number between 1 to 3 itself and then apply the Rock Paper Scissors game logic (which is paper beats rock, rock beats scissors, scissors beats paper), Then it will tell the player if the player or server has won. Write another client program to communicate with the server. *Bonus marks for making the server such a way the any time any player can leave and another player can join.

Reading References:

1. http://www.oracle.com/technetwork/java/socket-140484.html
2. https://docs.oracle.com/javase/tutorial/networking/sockets/
3. https://docs.oracle.com/javase/7/docs/api/java/net/ServerSocket.html