Faculty of Engineering, University of Jaffna

Department of Computer Engineering

EC5080: Software Construction

Lab 09: Concurrency and network clients

Task01:

You are given a single thread client-server program.

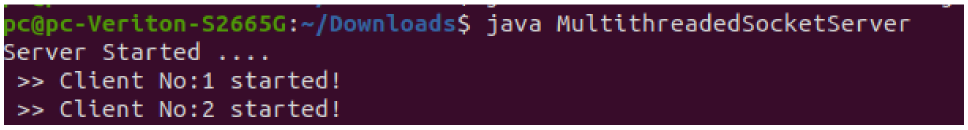
```
import java.net.*;

import java.io.*;

public class TCPServer {

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

    try{

      ServerSocket server=new ServerSocket(8888);

      Socket serverClient=server.accept();

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

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

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

      String clientMessage="", serverMessage="";

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

        clientMessage=inStream.readUTF();

        System.out.println("From Client: "+clientMessage);

        serverMessage=reader.readLine();

        outStream.writeUTF(serverMessage);

        outStream.flush();

      }

      inStream.close();

      outStream.close();
      serverClient.close();
      server.close();
    }catch(Exception e){
      System.out.println(e);
    }
  }
}
```

```
import java.io.*;
import java.net.*;
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")){
      clientMessage=br.readLine();
      outStream.writeUTF(clientMessage);
      outStream.flush();
      serverMessage=inStream.readUTF();
      System.out.println("From Server: "+serverMessage);
    }
    outStream.close();
    outStream.close();
    socket.close();
  }catch(Exception e){
    System.out.println(e);
  }
  }
}
```

Run the programs on two consoles. Explain the purpose of each line of code as comments next to each.

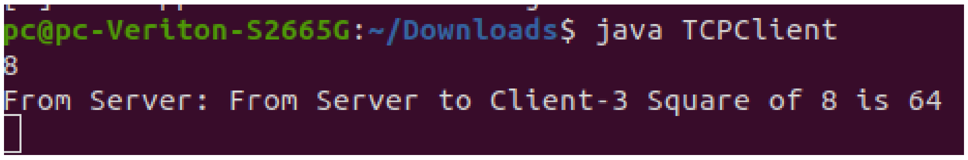
Task02:

Write a multithreaded client (multi)-server program in Java. The multithreaded server will be start running at first. The figure shows the example server console.



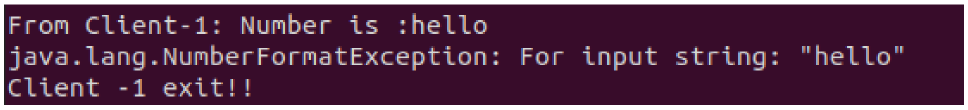
```
pc@pc-Veriton-S2665G:~/Downloads$ java MultithreadedSocketServer
Server Started ....
 >> Client No:1 started!
 >> Client No:2 started!
```

When running the given TCPCLient.java you will get the output as follows.



```
pc@pc-Veriton-S2665G:~/Downloads$ java TCPClient
8
From Server: From Server to Client-3 Square of 8 is 64
```

When the input from a client is not a number, that particular client should exit. The figure shows the example (a part) server console.



```
From Client-1: Number is :hello
java.lang.NumberFormatException: For input string: "hello"
Client -1 exit!!
```

Help: SCThread xx= new SCThread (acceptance, counter); You can write a Java class to send a request to a separate thread where the counter helps to calculate the requesting client's number. And it outstream the server's message to the client.

Upload both tasks' source code and output screenshots of task2 as PDF file in a zip folder named Lab09_RegNo.