TP1: Socket Programming in Java

ii.3502

Objective :

The aim of this TP is to show you and learn how to create a basic and simple Server / client that connect to each other using sockets over TCP using JAVA as programming language.
To use JAVA , you need to install the JAVA Development Kit (JDK) and integrated development environment (IDE) such as Netbeans,IntelliJ IDEA etc ...

Part 0: Installation and configuration

In this part you have to download JDK from its source link to start developing with JAVA (for this TP we are going to use JAVA 8 so we need to download the right version that support JAVA8 " JDK 8")

JDK download links :

- Windows : https://shorturl.at/lmqvY
- Linux : https://shorturl.at/rACGJ
- MacOS : https://shorturl.at/berD8

Download the IDE you that you want to use to perform this TP

For windows users :

- Netbeans : https://shorturl.at/KMSX4
- IntelliJ (community) : https://shorturl.at/lpwB9
- Eclipse : https://shorturl.at/lrvN6

For linux users :

- Netbeans : https://shorturl.at/ixKYZ
- IntelliJ (community) : https://shorturl.at/oHRV1
- Eclipse : https://shorturl.at/glrS3

For MacOs Users:

TP1: Socket Programming in Java

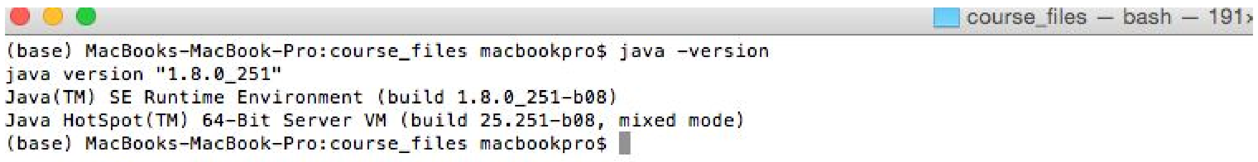
ii.3502

- Netbeans : https://shorturl.at/gO248
- IntelliJ (community) : https://shorturl.at/orzZ4
- Eclipse : https://shorturl.at/dfqwC

Follow the install instructions from official source

After installing the JDK 8 make sure you have java in you machine by trying to take the commande bellow in your terminal or ms-dos or powershell (java version should be 1.8) where 8 mean the JAVA8

"java -version"



```
(base) MacBooks-MacBook-Pro:course_files macbookpro$ java -version
java version "1.8.0_251"
Java(TM) SE Runtime Environment (build 1.8.0_251-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.251-b08, mixed mode)
(base) MacBooks-MacBook-Pro:course_files macbookpro$
```

If any error occur , make sure to update the path environment variables after setting JAVA_HOME and add it to main PATH

Part I: Simple server using TCP sockets

In this part you have to create a simple server that cad add two given numbers by following these steps:

- Creating and Running a Server
- Connecting to server using PuTTy or your terminal in linux or MacOs
- Creating a Java Client

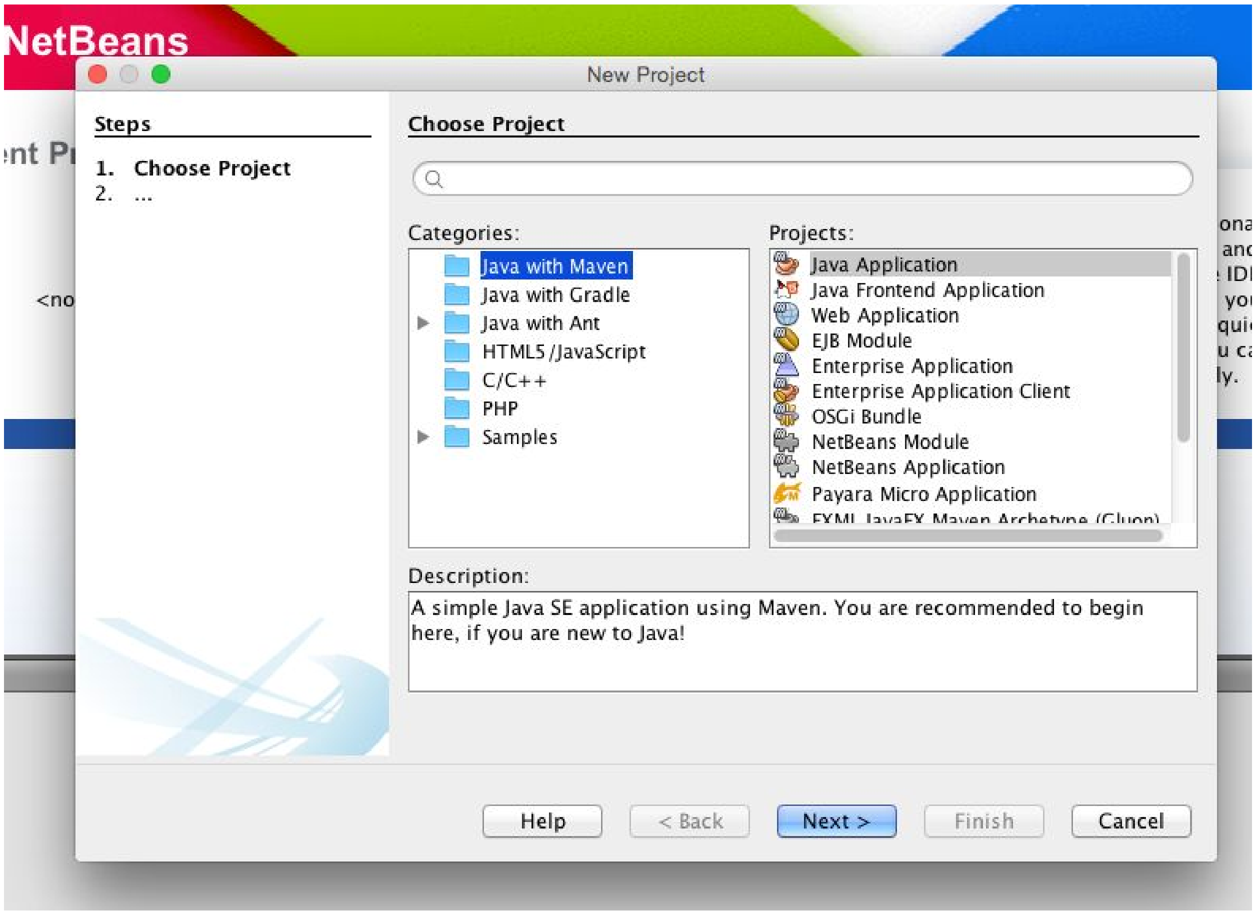
After everything is set up correctly as described in the Part 0 open your IDE and try to create a new project
Follow the instruction bellow

TP1: Socket Programming in Java

ii.3502

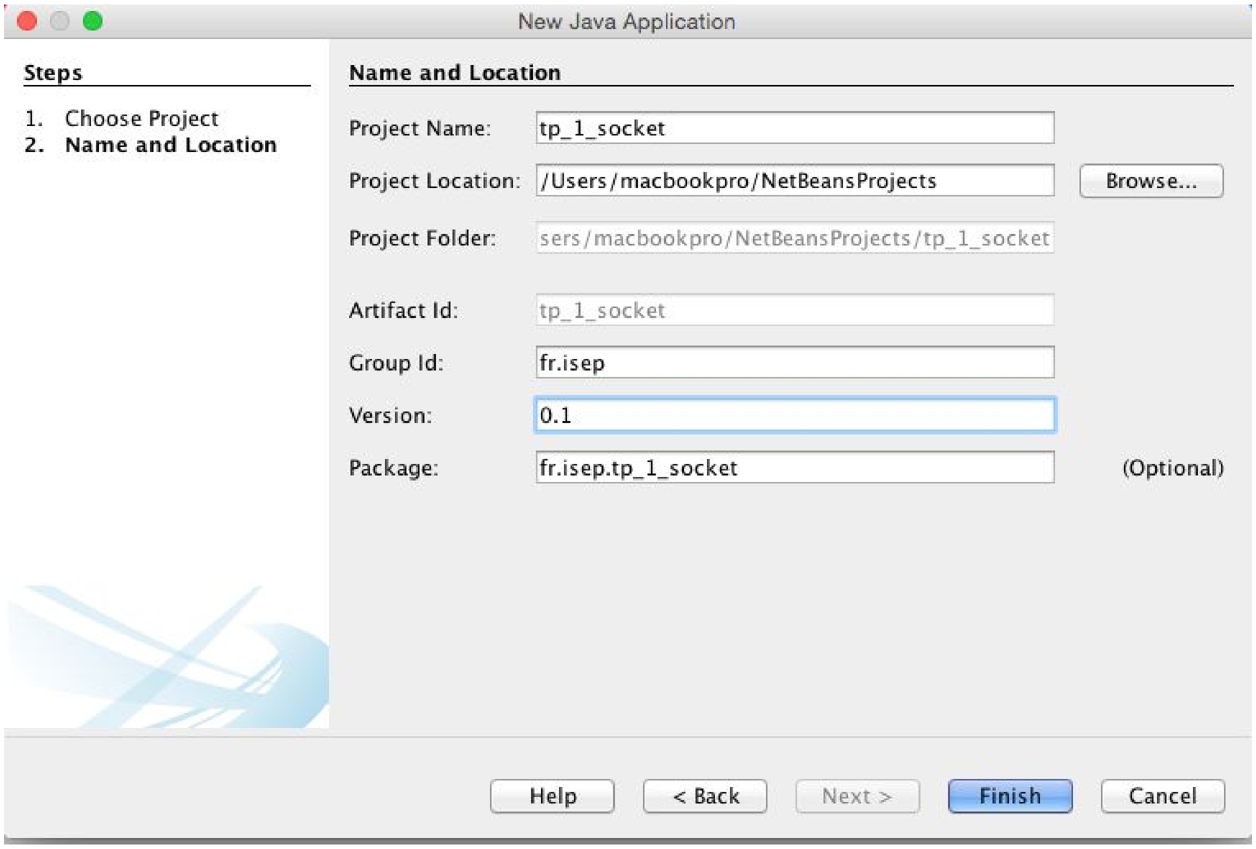
I.1. creating new project

Open IDE -> new project -> Select JAVA with Maven in catégories window -> Java Application in projects window -> click Next



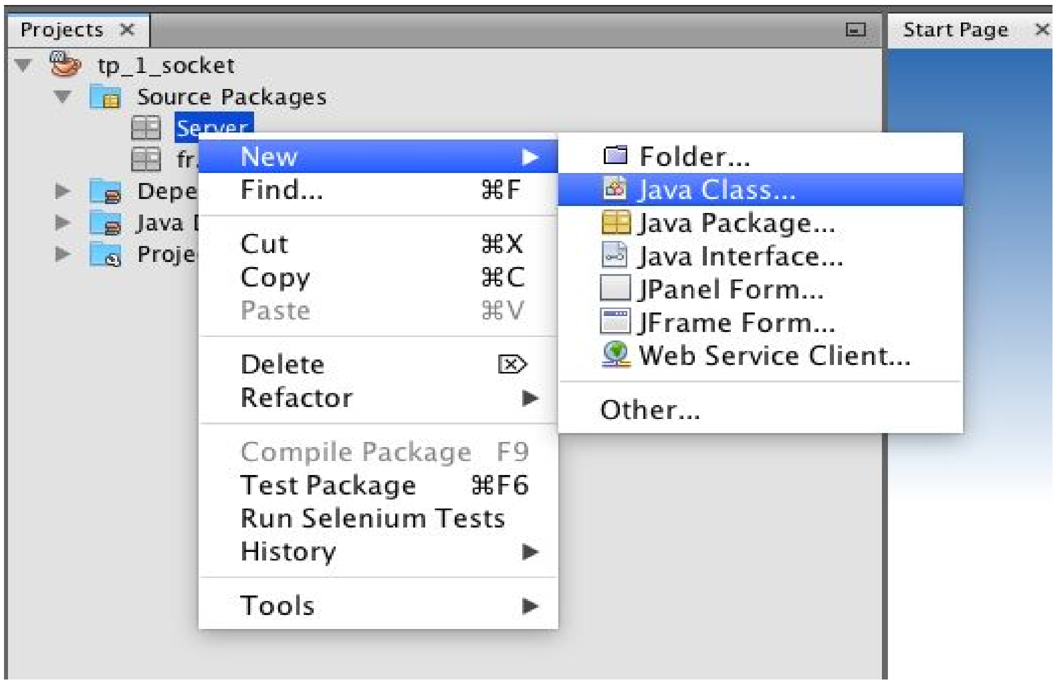
A simple Java SE application using Maven. You are recommended to begin here, if you are new to Java!

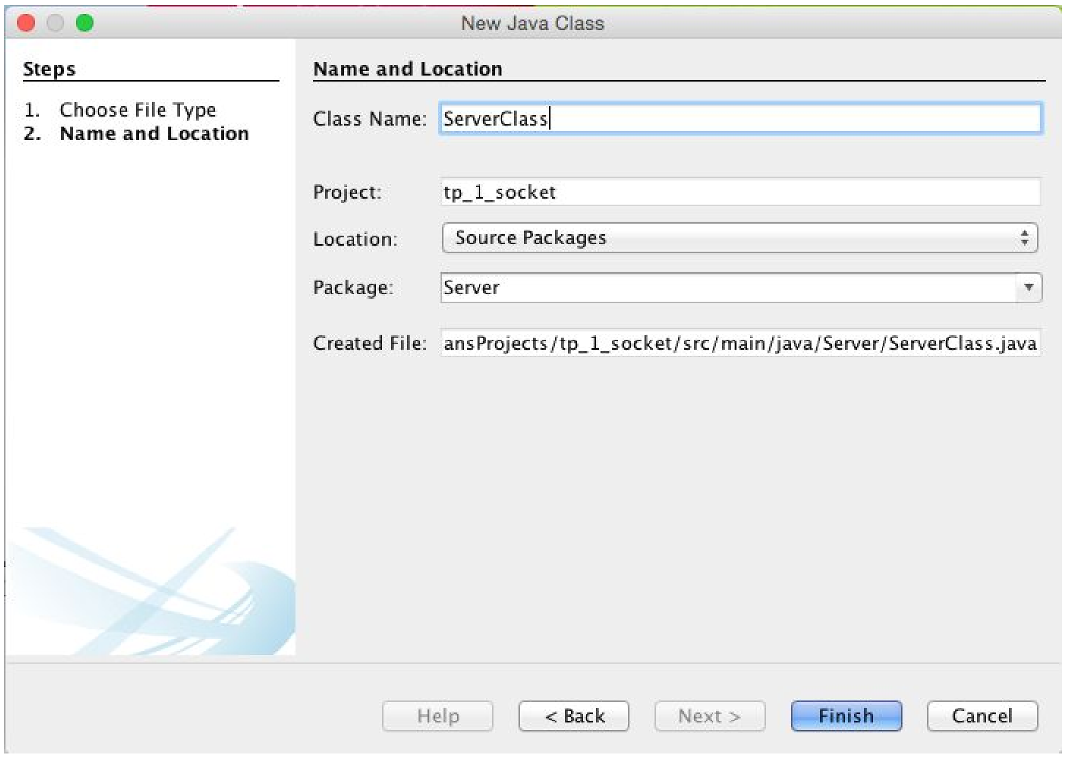
Fill the name and location as showing in the picture bellow :



Project Name: tp_1_socket
Project Location: /Users/macbookpro/NetBeansProjects
Project Folder: sers/macbookpro/NetBeansProjects/tp_1_socket
Artifact Id: tp_1_socket
Group Id: fr.isep
Version: 0.1
Package: fr.isep.tp_1_socket

Under your project create a folder name it “ server” and within this folder you have to create the Class with a name of : “ ServerClass” as shown in the screenshots bellow:





Name and Location

Class Name: ServerClass

Project: tp_1_socket

Location: Source Packages

Package: Server

Created File: ansProjects/tp_1_socket/src/main/java/Server/ServerClass.java

Write below the code for the Server that does the Addition operation. Copy the following code into the Server class that you had just created.

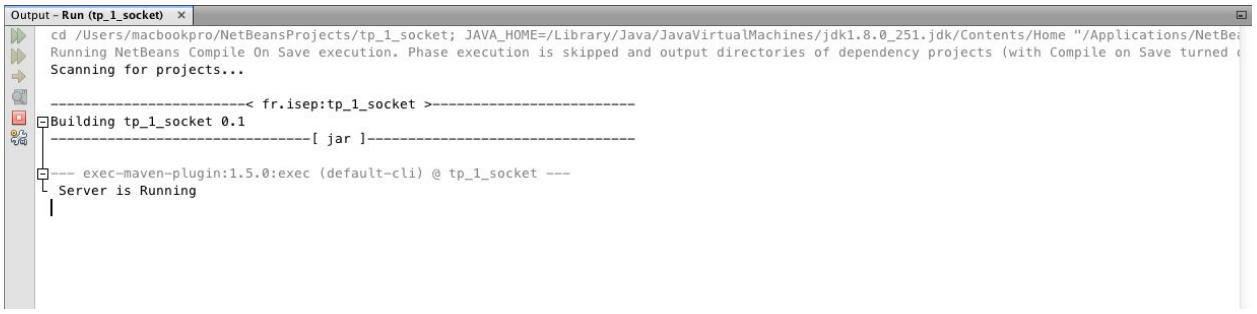
```
package Server;

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

/**
 *
 * @author macbookpro
 */
public class ServerClass {
    public static void main(String argv[]) throws Exception
             {

 System.out.println(" Server is Running " );
 ServerSocket mysocket = new ServerSocket(5555);
    while(true)
       {
           Socket connectionSocket = mysocket.accept();

           BufferedReader reader = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
           BufferedWriter writer;
     writer = new BufferedWriter(new
OutputStreamWriter(connectionSocket.getOutputStream()));
           writer.write("*** Welcome to the Calculation Server (Addition Only) ***\r\n");
           writer.write("*** Please type in the first number and press Enter : \n");
           writer.flush();
           String data1 = reader.readLine().trim();
           writer.write("*** Please type in the second number and press Enter :\n");
           writer.flush();
           String data2 = reader.readLine().trim();
```

```
            int num1=Integer.parseInt(data1);
            int num2=Integer.parseInt(data2);
            int result=num1+num2;
            System.out.println("Addition operation done " );
            writer.write("\r\n=== Result is : \n"+result+"\n" );
            writer.flush();
            connectionSocket.close();

         }
               }

}
```

Compile and Run the Application now

Expected result:

As output in your IDE you should see “ SERVER is Running”



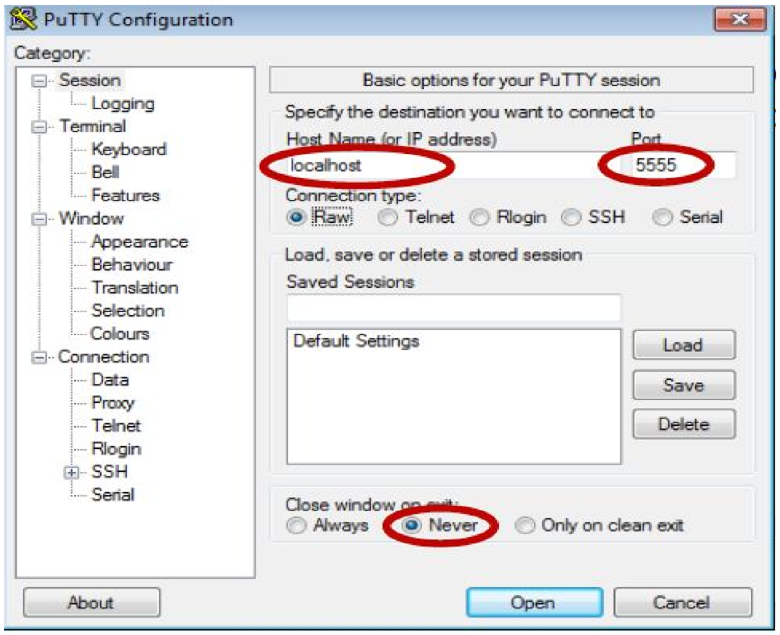
Output - Run (tp_1_socket)

```
cd /Users/macbookpro/NetBeansProjects/tp_1_socket; JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_251.jdk/Contents/Home "/Applications/NetBe
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned
Scanning for projects...

------------------------< fr.isep:tp_1_socket >-------------------------
Building tp_1_socket 0.1
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:1.5.0:exec (default-cli) @ tp_1_socket ---
 Server is Running
```

Ones the server is Running we need to connect to this server using remote PuTTy or your terminal

I.2. Connecting to the server using PuTTY or Terminal or MS-DOS

PuTTY is a simple terminal program that we use to connect to other machines usually through SSH, Telnet and so on. (1) Download PuTTY. (2) Double Click on PuTTY to start it. Fill it as shown below. The address is usually localhost for your computer. Choose RAW for the connection type. The port is 5555 as shown in the code above. Choose NEVER for the closing of the window.



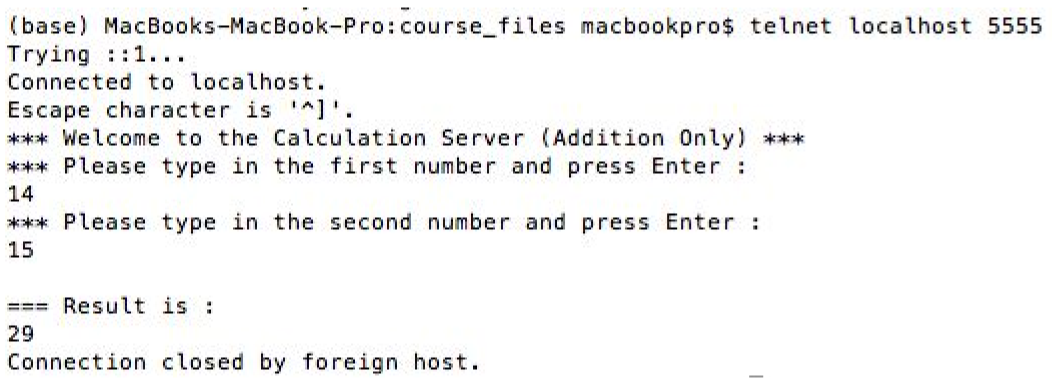
Specify the destination you want to connect to
Host Name (or IP address): localhost
Port: 5555
Connection type: [✓ Raw] Telnet Rlogin SSH Serial
Close window on exit: Always [✓ Never] Only on clean exit

Once you have filled all the required information, click on Open. A black window should shown up asking for the first number…

Entre the two requested number and tape entre and see the results

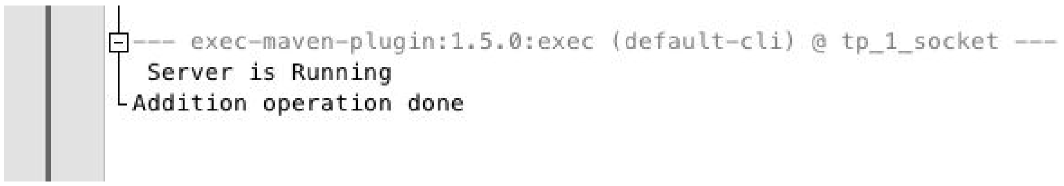
In case you use your terminal or your MS-DOS you have to telnet the server with command bellow:

```
telnet localhost 5555
```



```
(base) MacBooks-MacBook-Pro:course_files macbookpro$ telnet localhost 5555
Trying ::1...
Connected to localhost.
Escape character is '^]'.
*** Welcome to the Calculation Server (Addition Only) ***
*** Please type in the first number and press Enter :
14
*** Please type in the second number and press Enter :
15

=== Result is :
29
Connection closed by foreign host.
```

After the operation is done in the server side you should see a message " Addition operation done"

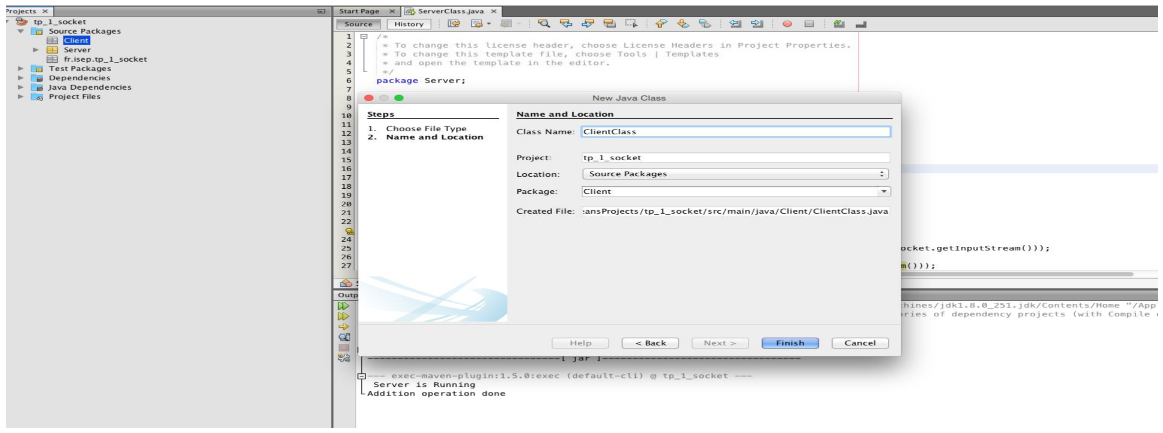


```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ tp_1_socket ---
 Server is Running
Addition operation done
```

I.3. Creating a Java Client

Instead of using PuTTY or any other third party client like terminal or MS-DOS for the Addition Server, we can write our own Java Client through the following steps.

1) Within your IDE , Create NEW Java Folder a new project with class name : "ClientClass" as done before for the "ServerClass"



```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Server;
```

Name and Location
Class Name: ClientClass
Project: tp_1_socket
Location: Source Packages
Package: Client
Created File: :ansProjects/tp_1_socket/src/main/java/Client/ClientClass.java

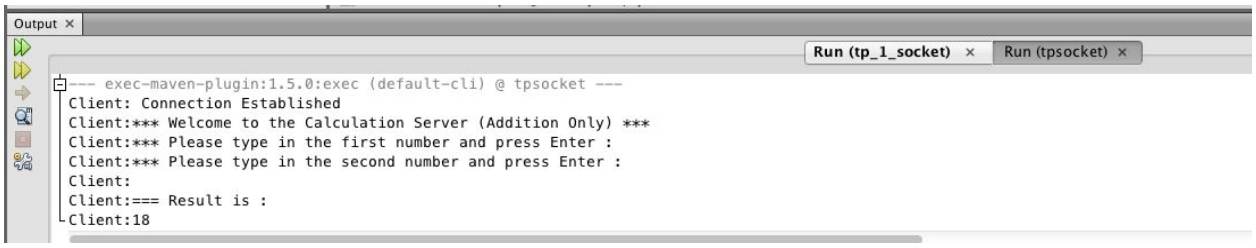
```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ tp_1_socket ---
 Server is Running
Addition operation done
```

Copy the following code into the Client class “ ClientClass”. The Client is programmed to send two numbers: 8 and 10 to the server for the Addition operation.

```
package Client;
import java.io.*;
import java.net.*;
import java.util.*;

/**
 *
 * @author macbookpro
 */
public class ClientClass {
    public static void main(String argv[])
{
try{
 Socket socketClient= new Socket("localhost",5555);
 System.out.println("Client: "+"Connection Established");

 BufferedReader reader = new BufferedReader(new
InputStreamReader(socketClient.getInputStream()));
 BufferedWriter writer= new BufferedWriter(new
OutputStreamWriter(socketClient.getOutputStream()));
 String serverMsg;
 writer.write("8\r\n");
 writer.write("10\r\n");
 writer.flush();
 while((serverMsg = reader.readLine())!= null) {
                       System.out.println("Client:" + serverMsg);
               }
       }catch(Exception e){e.printStackTrace();}
       }
   }
```

Make sure that the server is running!
Run the client now. You will get the following into the console:

Expected result :



```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ tpsocket ---
Client: Connection Established
Client:*** Welcome to the Calculation Server (Addition Only) ***
Client:*** Please type in the first number and press Enter :
Client:*** Please type in the second number and press Enter :
Client:
Client:=== Result is :
Client:18
```

I.4. Server/Client File transfer :

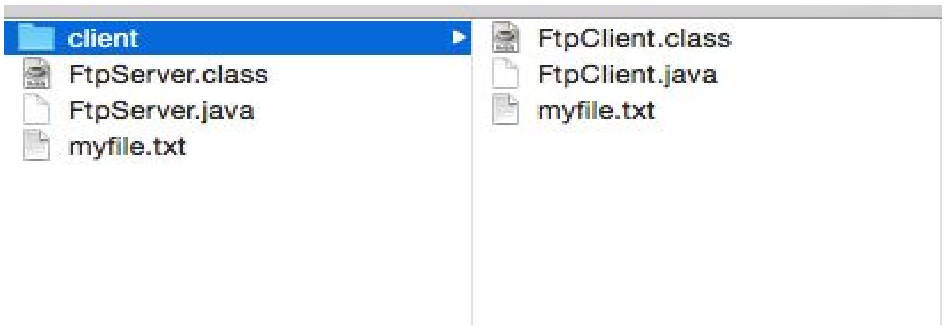
In this part we are going to create a Client/server file transfer using socket:

1. You have to create a Client that read a txt file and transfer it to the server
2. In the server side you need to be sure that the server read the file and save it to a specifique location.

Hints :

- For server :
    - Create serversocket (handle exception)
    - Create a file input (FileInputStream)
    - Read all lines from input stream
    - Write output into file
- For client :
    - Create socket connection to your server
    - Create FileOutputStream
    - Read all lines from your outputStream

Expected result:
The myfile.txt should be transferred to the server side from the client as showing bellow



TP1: Socket Programming in Java

ii.3502

Part II: Threaded Server using TCP Socket

In part I, we have created a simple server using TCP sockets. Because of the limitation of accepting only a single client at a time, we will show in this part how to create a multithreaded server that can handle multiple client connection at the same time.

1) Create a new project named ThreadedServer. Then, create a class named: Server.

2) Copy the code (Server.java Part I) in the Server Class.

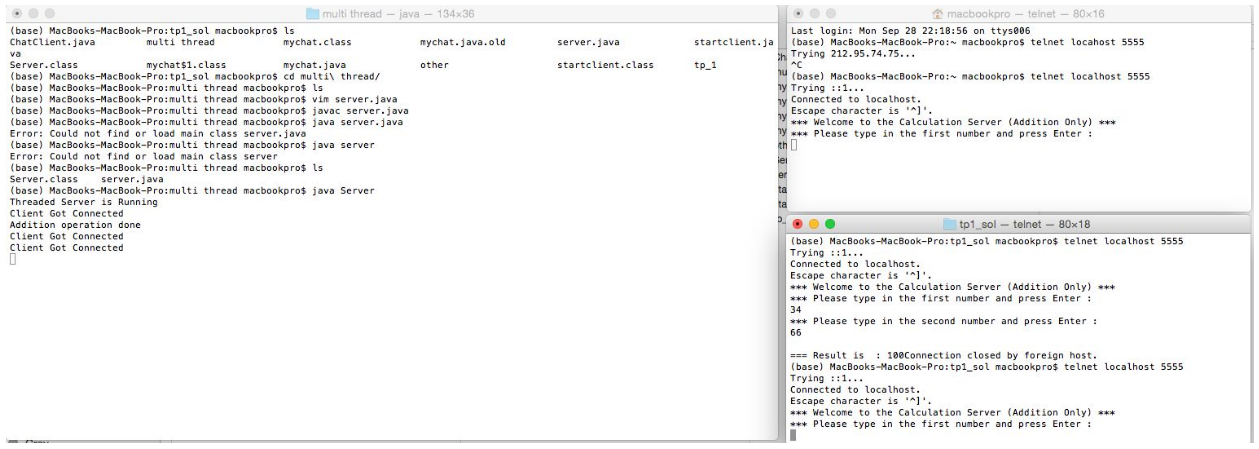
3) Modify the code in order to support Multi-threading. Multi-threading is a method of writing code for executing tasks in parallel. (For help) Consider the following code segments:

> **class Server implements Runnable** To implement a thread, the class needs to implement the Runnable interface OR Extends the Thread class.
>
> **public Server(Socket s){** Creating a constructor for the server class which takes the object Socket as an argument
>
> **public void run(){** This is the golden method of any thread. When making a thread, the core code to be executed needs to be placed within the run method.
>
> **Socket sock = mysocket.accept();** The server is waiting for client connection to be accepted. Once a successful connection is made, the socket object is created.
>
> **Thread serverThread=new Thread(server);** Because we chose to implement the Runnable interface, we need to create a Thread object. The thread object usually takes as argument any class that implements the Runnable interface.
>
> **serverThread.start();**This method (start()) would trigger the execution of the code within the run() method.

4) Run the Server now From your IDE (in my case it's NetBeans) or terminal.

5) Open TWO Putty terminal windows and connect to the server. The address is usually localhost for your computer. Choose RAW for the connection type. The port is 5555 as shown in the code above. Choose NEVER for the closing of the window. Once you have filled all the required information, click on Open. A black window should show up asking for the first number and so on…

Expected results:



```
(base) MacBooks-MacBook-Pro:tp1_sol macbookpro$ ls
ChatClient.java         multi thread            mychat.class            mychat.java.old         server.java             startclient.ja
va
Server.class            mychat$1.class          mychat.java             other                   startclient.class       tp_1
(base) MacBooks-MacBook-Pro:tp1_sol macbookpro$ cd multi\ thread/
(base) MacBooks-MacBook-Pro:multi thread macbookpro$ ls
(base) MacBooks-MacBook-Pro:multi thread macbookpro$ vim server.java
(base) MacBooks-MacBook-Pro:multi thread macbookpro$ javac server.java
(base) MacBooks-MacBook-Pro:multi thread macbookpro$ java server.java
Error: Could not find or load main class server.java
(base) MacBooks-MacBook-Pro:multi thread macbookpro$ java server
Error: Could not find or load main class server
(base) MacBooks-MacBook-Pro:multi thread macbookpro$ ls
Server.class    server.java
(base) MacBooks-MacBook-Pro:multi thread macbookpro$ java Server
Threaded Server is Running
Client Got Connected
Addition operation done
Client Got Connected
Client Got Connected
```

```
Last login: Mon Sep 28 22:18:56 on ttys006
(base) MacBooks-MacBook-Pro:~ macbookpro$ telnet locahost 5555
Trying 212.95.74.75...
^C
(base) MacBooks-MacBook-Pro:~ macbookpro$ telnet localhost 5555
Trying ::1...
Connected to localhost.
Escape character is '^]'.
*** Welcome to the Calculation Server (Addition Only) ***
*** Please type in the first number and press Enter :
```

```
(base) MacBooks-MacBook-Pro:tp1_sol macbookpro$ telnet localhost 5555
Trying ::1...
Connected to localhost.
Escape character is '^]'.
*** Welcome to the Calculation Server (Addition Only) ***
*** Please type in the first number and press Enter :
34
*** Please type in the second number and press Enter :
66

=== Result is  : 100Connection closed by foreign host.
(base) MacBooks-MacBook-Pro:tp1_sol macbookpro$ telnet localhost 5555
Trying ::1...
Connected to localhost.
Escape character is '^]'.
*** Welcome to the Calculation Server (Addition Only) ***
*** Please type in the first number and press Enter :
```

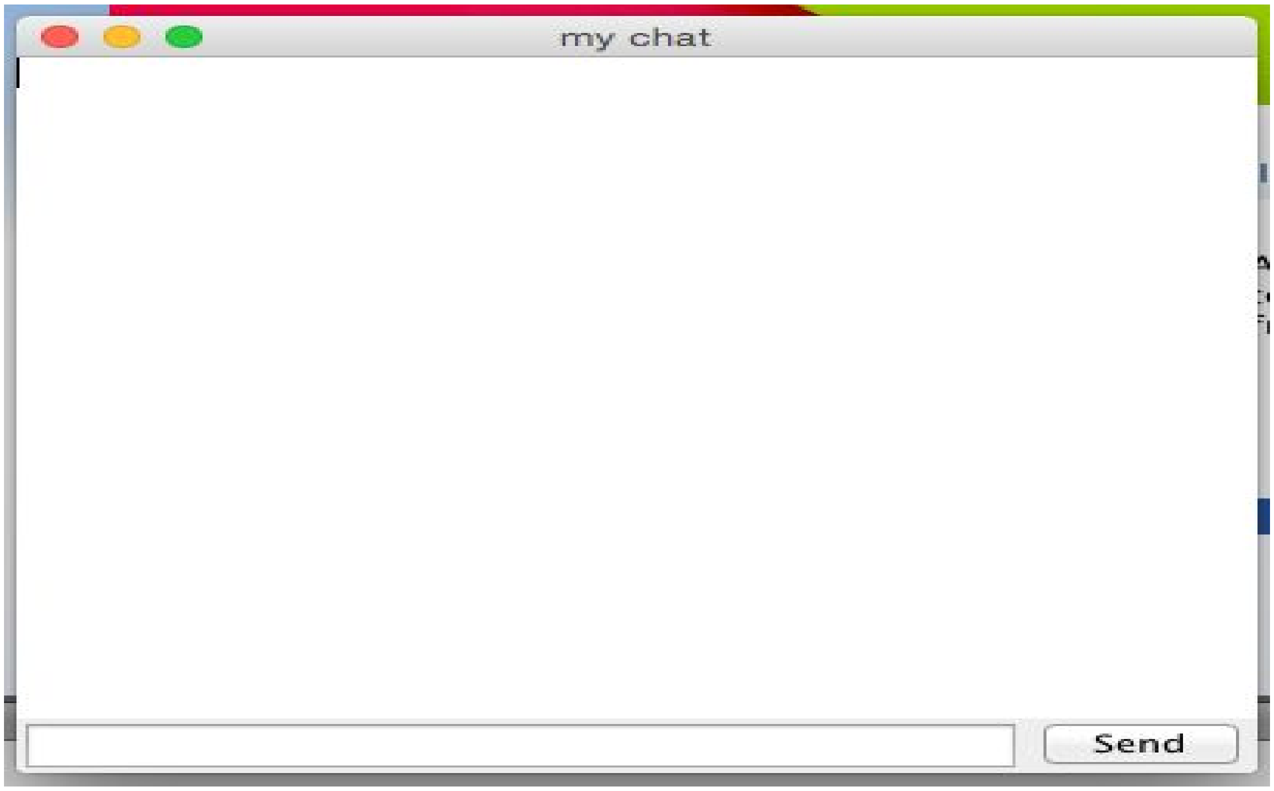
Part III: Group chat system using low-level sockets

In order to create the group chat system using low-level sockets, you will be having three simple steps.

- 1 - Creating the user interface
- 2 - Creating the Chat Server
- 3 - Messages to All Clients (every client should receive the broadcasted message)

III.1. Creating a Java Client

1) Create a simple graphical interface for the client to send and receive the chat messages (“chat.java”). Then, create an instance of the interface and start the client. Once you run the code, you shall get the following user interface.



You have now to complete the “mychat.java” file.

3) Make a simple event handler for sending a message from the client to the server when the user clicks the send button after typing some text. You must define the instance variable for a BufferedWriter.

4) Create the writer object which writes to the server. Within the constructor, we add the following code. The localhost address can be changed to reflect the server address. 5555 is the chosen port number.

5) Add the action listener to the button (Send).