Programming project1
Due date : July 19, 2011

TCP Client/Server (Calculator Service)

In this project you will implement a multi-thread server which is capable of processing multiple simultaneous requests from one or more than one client. Each client sends a list of numbers to the server, and then it sends requests to calculate summation (SUM) , maximum (MAX) or minimum (MIN) of those numbers. The server processes the client request and sends the result to the client. Each client can send more than one request and upon receiving each request the server sends the calculation result to the client.

TCPClient:

Since the server is implemented as a multi-thread server, the Client can be implemented a persistent connection. The connection between the client and server is a TCP connection on port 6789. The client creates one socket and each request is sent through this socket. Write the client program in a way to accept the server_host_name as input argument when executing it in command line.

The Client shows a menu to the user containing

(1) Enter the list of numbers
(2) Summation
(3) Maximum
(4) Minimum
(5) Exit

Through choose (1) the user enters the list of numbers. She/He enters 0 to finish the list (assume that the list of numbers doesn't contain 0). Through choices 2-4 , the user can choose one of the operands for the list of numbers.

Here's a sample sequence of actions:
1- The user chooses option (1)
2- The client sends a request "LIST the list of numbers"
3- The user entered 2,3 and -5,
4- The client send "LIST 2 3 -5" to the server.
5- The user enters one of the operands (by choosing options 2-5).
6- The client sends "SUM","MAX", "MIN" and "EXIT" for the options (2) to (5) respectively.
7- The client waits to receive the result from server
8- The client shows the result to the user,
9- The client shows the menu to the user and this loop continues until the user chooses Exit (i.e. option (5)).

The structure of the client will be as follows:

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

public final class CalClient
{
        public static void main(String argv[]) throws Exception
        {
                . . .
        }
}
```

TCPServer:

The server will be multi-threaded, in which the processing of each incoming request will take place inside a separate thread of execution. This allows the server to service multiple clients in parallel, or to process multiple requests from a client. Once we create a new thread of execution, we need to pass an instance of the class that implements the `Runnable` interface to the Thread's constructor. This is the reason that we define a separate class called `CalRequest`. The structure of the server will be as follows:

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

public final class CalServer
{
        public static void main(String argv[]) throws Exception
        {
                . . .
        }
}

final class CalRequest implements Runnable
{
        . . .
}
```

When a connection request is received, we create an `CalRequest` object, passing a reference to the `Socket` object that represents our established connection with the client to its constructor.

```
// Construct an object to process the Calculation request message.

CalRequest request = new CalRequest( ? );

// Create a new thread to process the request.
```

```
Thread thread = new Thread(request);

// Start the thread.
thread.start();
```

In order to have the `CalRequest` object handle the incoming calculation service request in a separate thread, we first create a new `Thread` object, passing a reference to the `CalRequest` object to its constructor, and then call the thread's `start()` method.

The structure of the CalRequest class is shown below:

```
final class CalRequest implements Runnable
{

        Socket socket;

        // Constructor
        public CalRequest(Socket socket) throws Exception
        {
                this.socket = socket;
        }

        // Implement the run() method of the Runnable interface.
        public void run()
        {
                . . .
        }

        private void processRequest() throws Exception
        {
                . . .
        }
}
```

Submission:

1- Create a directory X where X is your surname. In the directory X put three files: CalServer.java, TCPClient.java and CalRequest.java

2- Compress your directory X to create a .zip file.

3- Submit your compressed file online at https://courses.cs.sfu.ca/ .