CSCD50 Computer Networks - Winter 2016

Programming Assignment 1

Socket Programming

Due: February 24th, 2016, 5:00PM.

In this assignment, you will implement a socket to send and receive files.

Throughout this assignment, the *server* is the receiver, and *client* is the sender of the files.

This assignment is in 2 parts.

Part 1 – Client Side Implementation (40%)

In files *A1Listener.java* and *A1Receiver.java*, You are given the server side component of the socket implementation. In this part, you will implement *A1Sender.java*, which is the client side of the implementation.

A1Sender.java is supposed to do the following:

- Contain a Java *main*() method to run *A1Sender.java* as an application and a *sendFiles*() method. Your main method is as follows:

```
public static void main(String [] args)    {
        sendFiles(3);
}
```

  The signature of sendFiles() is as follows:

```
private static void sendFiles(int k);
```

  sendfiles() should execute correctly on any value of $k>0$ to transmit k files as specified in the next bullet.

  You can code other methods in *A1Sender* if you think necessary as long as they comply with the code given above. Do not add other classes into your code.

- Read from the default directory the files *this_0.txt, this_1.txt*, … *this_(k-1).txt* and send them one after the other, in increasing order (*this_0.txt* first, *this_1.txt* next …) to the server through the socket connection. See the sample files given for test: *this_0.txt*, *this_1.txt* and *this_2.txt*. Once *A1Sender* sends these 3 files, the resulting output file at the server side should look like *StreamReceived.txt*:

- The dedicated server port for this connection is 7777. Your client should connect to the server on "localhost" at port 7777.

- Send an INITIAL_GREETING to the server which is "/<IP_address>:<port#>". Use *Socket.getLocalSocketAddress*() for this message. Also search "INITIAL_GREETING" and see the relevant line comment in *A1Receiver.java*.
- The socket connection between the client and the server is torn down when the client side closes its socket. (The server is keeping the connection open unless there is an error to exit—as given in *A1Receiver.java).* After sending all files successfully, the client should close its socket.
- *A1Sender.java* should execute correctly without any changes to the given *A1Receiver.java* and *A1Listener.java*.

Part 2 – Transfer Files (60%)

Rewrite or modify the given *A1Receiver.java* and your *A1Sender.java* respectively into A1ReceiverFiles.java and A1SenderFiles.java to transmit the files separately each. Your code should do the following:

- *A1SenderFiles* should contain a Java main() method and a *sendTheseFiles*() method. Your main method is as follows:

```
public static void main(String [] args)    {
        sendTheseFiles(3);
}
```

  The signature of *sendTheseFiles*() is as follows:

```
private static void sendTheseFiles(int k);
```

  sendTheseFiles() should execute correctly on any value of k>0 to send k files. You can add other methods to *A1SenderFiles* if you think necessary—as long as they comply with the above specifications. Do not add other classes into your code.

- The server side should receive each file sent by the client and save it to an existing "Test" subdirectory under the default directory, by the same file name as the client side. Eg.: after your implementation is run to send *this_0.txt*, *this_1.txt* and *this_2.txt*, these 3 files should appear in the "Test" folder without any changes in their format with their respective names.

  A suggested implementation is, before sending each one of the *k* files, send the file name and length to the server. The server then can receive as many bytes as specified and store to disk under the file name given.

  You can also make the client send a special sequence of bytes to serve as a delimiter to mark the end of a file.

Running The Code:

To run your assignment for both parts, do the following:

- Create two projects on Eclipse: *ClientSide* and *ServerSide*.
- Put the classes *A1Listener*, *A1Receiver* and *A1ReceiverFiles* into project *ServerSide*.
- Put the classes *A1Sender* and *A1SenderFiles* into project *ClientSide*.
- First, run *A1Listener* as a Java project. Then run A1Sender for Part 1, and A1SenderFiles for part 2 as a Java project.
- For running Part 2, replace the single occurrence of "A1Receiver" in *A1Listener* with "A1ReceiverFiles".

What to Submit:

The submission deadline is February 24th, 2016, **5:00PM sharp**.

Submit your source codes on BlackBoard in 3 files as mentioned above by their names: *A1Sender.java*, *A1SenderFiles.java*, *A1ReceiverFiles.java*.

In each file, write the full name and StudentID of each team member within a comment at the top of the file.

Make sure your codes compile correctly. Classes with compile time errors won't be marked.

Ensure the quality of your code. Your code should be readable and well written by Java standards.