Cairo University
Faculty of Engineering
CMP(N) 306
Advanced Programming Techniques

Socket Programming in Java

Requirement (1):
In this requirement, we will implement a network with socket programming using server-client architecture. We have a single server that can handle requests from two different types of clients **Admin Client and Member Client.**

1. The server has two groups (i.e. two chat groups). The communication between clients in any group is handled by the server.

2. The server offers two services: **SUBSCRIBE** (for Member Clients only) and **BROADCAST** (for Admin Clients only).

3. A Member Client can subscribe to one of the two groups. If a Member Client subscribes to a group, then they will receive all messages sent in this group. However, they can not receive any messages sent in the other group.

4. A Member Client is **NOT allowed** to send any messages in the group. They just listen to the received announcements.

5. An Admin Client is the only client that can broadcast messages to a group.

6. If an Admin Client broadcasts a message to a group, then all the Member Clients in this group **ONLY** must receive this message.

7. An Admin Client does not subscribe to a group. They just broadcast their messages to one of the two groups. Therefore, if an Admin Client decides to broadcast a message to all Member Clients subscribed to a group, the admin will not receive the message himself.

8. The server must handle multiple requests from Member Clients at the same time. For example, the server can serve 3 Member Clients in Group 1 and 2 Member Clients in Group 2 at the same time.

9. The server must handle mutliple requests from Admin Clients at the same time. For example, the server can serve 2 Admin Clients broadcasting to Group 1 and one admin client broadcasting to Group 2.

10. If two admins are broadcasting to the same group at the same time, then the Member Clients of this group will receive the messages from both admins. However, the two admins will not see each others' messages.

11. The server provides the service SUBSRIBE at port 3000, and provides the service BROADCAST at port 3001.

12. **Details of the SUBSCRIBE Service:**

A) **Communication between Server and Member Client (MC):**

A message is displayed to the MC: "Welcome, Which group do you want to join? (1) Section-1 or (2) Section-2"	
The MC chooses either 1 or 2 and sends this reply to the server. (let's say we chose 1)	The server waits for the reply of the MC and adds them to this group (if possible)
	The server replies back to the MC with a message of either: SUCCESS: **"*You are successfully added to the group Section-1 with ID = 6000".*** FAILURE: "***Sorry, the group reached its maximum count".***
The MC waits for the message from the server. The MC should parse this message to know its **ID** (the ID is a unique number generated from the server-side). If the failure message is received, then the client displays the message and terminates.	
After either message, the MC closes the socket and ends the connection with the server.	

B) **Notes:**

1. The above communication should be CONNECTION-ORIENTED.
2. If the success message is sent by the server to the MC, then the MC will not terminate but it wait endlessly for any incoming messages from the admin clients.
3. Define the unique ID generated to be equal to (6000 + the total number of member clients that have established any connected with the server at that moment).

13. **Details of the BROADCAST Service:**

A) **Communication between the Server (S), Admin Client (AC) and Member Client (MC):**

AC	Server	MC
A message is displayed to the AC: “Welcome, Which group do you want to broadcast your message to? (1) Section-1 or (2) Section-2”		
The AC chooses either 1 or 2 and sends this reply to the server. (let’s say we chose 1)	The server waits for the reply of the AC to know which group they will be broadcasting their message.	
	The server replies back to the MC with a message. **What is the message to be broadcast?**	
The AC replies back to the Server by typing any message to be broadcast.		
	The server receives the message to be broadcast.	
	The server sends this message to all MCs joined in the group.	
		Each MC in the group will receive the broadcast message and display it.
The AC sends another broadcast message to the server.		
	The server receives the message to be broadcast.	
	The server sends this message to all MCs joined in the group.	
		Each MC in the group will receive the broadcast message and display it.
The AC sends “end” to the server and terminates.		
	The server receives the message from the AC and closes the connection with the AC.	

B) **Notes:**

a) The above communication between the admin client and the server should be CONNECTION-ORIENTED.

b) The broadcasting communication between the server and the member clients should be CONNECTIONLESS. (How will the server keep track of the MC ports in order to send the packets through them? Ports should be **unique** for every client because we will be running our code from the same machine, and accordingly each client must use a different port in that machine). If we were going to have actual multiple machines, we could use the same port number without a problem.

c) The AC continues to send messages to be broadcast one after the other, until the AC sends "end" to the server. At this moment, the AC should terminate and the server should close connections with the AC.

d) The MCs and the AC communicate via the server. They don't communicate directly.

14. **Hints:**

a) **You will need three classes Server, Admin Client and Multiple Client.**
b) **The design of these classes is left for you.**
c) **Your code should be object oriented.**
d) **A software proposed solution is given as follows:**

Class Server:

1. **Class Member (Inner class defined within Server class):**
   a) This class should be **public**, **static** and **final**.
   b) This class has only data member ID (integer).
   c) The constructor takes as a parameter an integer variable, and sets *ID* with this integer parameter.
   d) It has only public method **getID()** that returns the ID.

2. **Class Group (Inner class defined within Server class):**
   a) This class should be **public**, **static** and **final**.
   b) This class has three data members:
      i. Name of the group (String)
      ii. List of Members in the group (ArrayList).
      iii. Maximum number of members in the group (a final variable = 3).
   c) The constructor initializes the name of the group as well as the member list.
   d) It has only public method **addMember().** This methods takes as a parameter a Member object and adds it to the list. The function returns true if the member is successfully added to the list, and false otherwise. (When should it return false?).

3. **Data Members of Class Server:**
   a) **groups:** ArrayList of the total groups operated by the server.
   b) **totalClients:** Number of all clients that have connected to the server at any point in time (even if some client has closed the connection some time later).

Example: If three clients were connected with the server, and then one of them disconnected, and then another client connected, the total clients will be 4 (we count all the clients that have connected with the server).

Note: You may need other inner classes.

e) **Class AdminClient & Class MemberClient are left for your design.**
f) You can refer to the output video attached in the lab folder to have an idea of how the output should be and how you should test your work.

Requirement (2):

- You are required to write a client-server program for determining the type of triangles.
    - Equilateral triangle: If the three sides of the triangle are equal.
    - Isosceles triangle: If only two sides are equal to each other.
    - Scalene triangle: If all the sides are not equal to each other.

- The client side reads a file containing the x-y coordinates of N triangles. The file format is as follows, for example:
  2
  Triangle 1:
  2 3
  3 0
  0 1
  Triangle 2:
  1 4
  0 0
  -1 -1
  This file contains two triangles, each triangle consists of three coordinates.

- Hint: You will need to define two classes: **Point and Triangle.**
- The client should read N triangles into a list of triangle objects, and then begins to **send each triangle object in the list** one by one to the server.
- Question: **How will the client send the triangle object to the server?**
- The server receives the triangle object and decides if this is an equilateral, isosceles, or scalene. It sends the answer back to the client. And the client prints the answer to the user.
- The user is prompted to press any key to continue before sending the next triangle object.
- Once the list is exhausted of triangles (no more triangle objects to be sent to the server), the client will terminate.
- The program should handle **multiple clients** at the same time.
- **Note:** After you calculate the distance between two points, round the result to the nearest integer.
- There is a sample file attached with test cases. The results should be (Isosceles, Isosceles, Scalene, Equilateral).
- You can refer to the output video attached in the lab folder to have an idea of how the output should be and how you should test your work.