Haoyang Ren z5183825

COMP3331 Assignment Report

Instruction:

- Assumption:
  - In order to compile the Server.java file, we need to put the “credentials.txt” the the same directory of Server.java file
  - For the input arguments, we assume the block_duration and timeout would be in milliseconds. For example:
    - If we want to set up a server with port number 8888, 5 minutes timeout and 1 minutes block_duration, we should run the command firstly like:
      - java Server 8888 60000 300000
    - Similarly, If we want to set up a client with port number 8888 and localhost, we should run the command secondly like:
      - java Client localhost 8888
  - After this, if you see the “Please enter username: ” from the client command, which means the server and client had been set up successfully.
  - There are actually for classes in the Server.java, which are Server.class, User.class, Packet.class and Task.class. Similarly, the Client.java also contains two class: Client.class and Packet.class. After user successfully compiling the code, they should be on the same directory with server and client.

Software Design:

- Packet Transformation:
  - The Packet.class is associated with both server and client, which is the only object that being delivered between these two classes. The Packet includes five attributes: auth, username, password, request and message. Since we using TCP protocol, the server and client could only accept bytes, So we define all the attributes as String in order to simply serialise to be byte array. Furthermore, we include the method the construct the string in to packet object and another method to tear down the packet object into string.

- User Information:
  - The User.class has an Aggregation relationship with Server. Once the client login to the system, it would be stored into an object of user, who have multiple attributes, such as username, password, blackList, socket, ObjectInputStream and ObjectOutputStream. The pending list is used to store the offline message that the other clients delivered.
  - There is also a users array list inside the Server.class, which could be retrieve users' information when it's necessary.
- Server design:
  - While the server is running, it would accept the welcome socket and generate a connection socket with multiple threads. Once the user login the system, it would create a new thread and assign a new connection socket and IOPutStream for each client.
  - After the client is activated, the server would keep listening the user's request and analyse the packet in the sequence of different request command. The server would match each user case with "request" and "auth" into every available case. The "auth" attribute is actually a boolean value indicate of the authorisation flag. For the worst case, if the server can not detect any of the available command matched with the request, it would send to unknown message to the client.
  - Finally, after the client logged out or automatically timeout, The server would clean up the previous clients information includes socket. After the period of timeout duration. The server would reestablish the connection with client and ask for username and password again.
- Client design:
  - While the server starting to running first, the client would attempt to connect with server with the given IP Address and port number. If the user do not provide the correct username and password over three times, it would start the timer task to block the user within the block_duration.
  - If the user provide the correct username and password, the client would start two thread, one is for respond for the current user input command, such as whoelse, whoelsesince, block, unblock and logout. The other thread is for listening other users' delivered content like message and broadcast.
  - User could also enter the startprivate to server to make a p2p connection.

Function behaviour:

- Once the user successfully connected with the system, they would see a welcome information printed on the screen. Simultaneously, the sever will notify all the rest of available users that the new user has logged in.
- After login, the user could press “Enter” to check the available command to interact with other users, like sending messages and broadcasting to all the available user. However, if the user exist in others’ black list, the message could not be delivered successfully and the current user could also get the feedback.

Design tradeoff:

- Originally, the users list was design as HashMap to store all the user information. However, the HasMap could not provide one-many type relationship, so I have to switch it to seperate class and aggregated in the server class, which also provide more efficiency with the user information retrieval.
- Furthermore, each user also given an independent ObjectInputStream and ObjectOutputStream. If we only use the single OutputStream in the server to deliver message, the client part would be packet corruption exception. The situation make me stuck for a long time.

Improvement:

- For the user login part, if the user enter the correct username and invalid password, the system would catch the Invalid password and notify the user on the screen. However, the system should record the username and only provide the password entrance in the second chance. The misbehaviour has not fixed yet due to the time lankness, which should be solved in less one day in the future.
- Last but not least, I have attempted to finished the last p2p part. I almost finished the client conversation but it still behaviour extremely unstable during the test. In order to keep the stability of the first part, I have to delete all the p2p part in the client and only keep startprivate method in the server. Please consider the mark for the p2p part with the blocks of code in the server. Thanks a lot!