Selected Topics in SE 1 (Software Engineering for Distributed Systems)

Assignment 1: Hangman Game (Single & Multiplayer)

Objective

The aim of this assignment is to create the classic hangman game which utilizes client-server architecture. You're required to implement it **using Socket programming and multithreading.**

The server should allow a single client (user) to play the game alone, and also allow multiple clients to play together (multiplayers). The server will request the login information from the user at the beginning before the game starts. The application should provide the following list of features:

1. **Supported Clients:** The application should support multiple users playing concurrently **(at least 4)**.

2. **Logging into the application:** To use the application, existing users would need to login, while new users need to register first.
    - To login, the user needs to send his username and password.
    - To register, the user needs to send his name, username, and password.
    - Invalid login/registration scenarios should be handled as follows:
        - if password is wrong, 401 error should appear (unauthorized)
        - if username is not found, 404 error should appear (not found)
        - if a username cannot be used to register a new user because it is already reserved, some custom error should appear.

3. **Game Setup:** The server should load the below files on startup:
    - The users' login credentials
    - Score history for each user
    - The game configuration (the number of incorrect guesses (attempts) needed to lose the game, the maximum and minimum number of players per team or game room, … etc.)
    - The lookup file containing the phrases to be guessed. Each line in the file contains one phrase (one or more words).

    After the files are loaded, the server waits for the players to login and then the game starts based on the options which the user chooses. Users can choose to quit the game anytime by pressing '-' character.

4. **Game Options:** Once logged in, the user can see the menu provided by the server, which allows him to choose whether he wants to play as a single player or to play with multiplayers (multiple clients). If the user chooses the multiplayer option, he can then play within specific teams (users he already knows).

5. **Teams:** Users can choose the option to play in the form of teams with the users they want to play with. In this case, two teams can play against each other. The name of the teams should be unique. The server should check that the number of players in both teams are equal before starting the game. Otherwise, it should display an error message. You should specify the team formation criteria on your own.

6. **The Game:** Once the game starts, the phrase to be guessed is shown to all the users within this game in the form of underscores to simulate the hidden letters (e.g., _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _). Word guessing will be case insensitive (i.e., if a user enters a lower-case letter and the word contains an upper-case letter, it should be counted as a correct guess if they are both equivalent to the same letter). The game will be in turns. The server should display the user's name who should play in each turn. When a user guesses a correct character, it should be shown to all the users in the game and the user's score should be updated and displayed to him.

7. **Number of attempts:** The game ends once the users are out of attempts. The server should end the game even if the phrase has missing characters to be guessed. The server should display the correct phrase at the end of the game. Also, the number of attempts should get updated and displayed for each user whenever he makes an incorrect guess.

8. **Scores:** Each user should have a score history for his last single/multiplayer games. You should specify the score calculation criteria on your own. This calculation should hold for all supported types of games (i.e., for games of students working in teams 2 **and** for games of students working in teams of 3).

9. **Additional Feature #1 (For students working in teams of 3):** You will implement a server application for the lookup of the phrases to be guessed, so that two servers will be there; one of them is the game server and the other one is the lookup server. The game server will call the lookup server to retrieve the phrases based on certain criteria (selection of phrases with a specific number or range of characters). Multiple simultaneous calls to the lookup server should be supported.

10. **Additional Feature #2 (For students working in teams of 3):** If the user chooses the multiplayer option, he will have an option to join a game room, where he can play

with other users (selected randomly by the server). Game rooms are not previously loaded by the server. They are created and used only during the server runtime and get deleted when the server shuts down. The game rooms should have a minimum and a maximum number of users (previously loaded from the game config file).

No graphical user interface is needed. The whole assignment can be run through the command line.

P.S. You must handle any exceptions that may occur. For example, you should handle the issues which may occur if a client disconnects at the middle of the game for any reason.

Guidelines and Submission Details:

1- The project must:
   a. Be developed in the Java Programming Language only.
   b. Compiles against JDK 8 at least.
   c. Can be imported directly into NetBeans IDE for Java.
2- The assignment will be solved in a group of **2 or 3 students** from the same lab. In case you're forming a group of 3, then you MUST implement the additional features #1 and #2 (Points 9 and 10).
3- If more than 2 team members submit only the features for a team of two, all team members will get zero.
4- If more than 3 team members submit the assignment, all team members will get zero.
5- You should submit your assignment as **ONE zip file** with the below naming convention:
   Assign1_GroupNumber_ID1_ID2 (example: A1ssign_S1_20116001_20116002)
6- You should submit **your source code along with a document** explaining any decisions or assumptions that you made and illustrate how you implemented the teams' formation and the scores criteria.
7- You SHOULD NOT copy any code from the internet or from your colleagues. It will be detected and considered as a cheating case.
8- Submission of the assignment will be on Google Classroom through a Google form link that will be shared later.
9- Deadline for the submission is **Saturday 25th of March 2023**