Jordan University
Department of Computer Engineering
Networks and Internet Programming
Course Project
Eng. Asma Abdel Karim

In this course project, you are required to write a network application that implements a **client/server gaming environment** using TCP API in Java.

The environment runs games between clients such that each game consists of two players. The game consists of asking the two players 5 multiple-choice questions. The player that sends the correct answer first gets the question points. The player that gets more points wins the game. More than one game can be run at the same time.
Attached with this project are two files:

1. *Question.java* which includes the definition of the *Question* class. The *Question* class represents the questions that will be used in our games. It consists of a *String* that represents the question, an array of *Strings* that represents the choices, an *integer* that represents the number of the correct answer, and an *integer* which represents the number of points assigned for this question.
2. *questions.out* which includes a serialized *ArrayList* that consist of a number of *Question* objects.

- At the server, the code takes each two clients connections and starts a game for them as follows:
    1. When the server application is first started, the *ArrayList* of questions must be read from the file *questions.out*.
    2. When a player is connected to the server, he should enter his username which will represent him in the game.
    3. If a client (player) connects to the server and there is no other player to start a game with, the server waits for another client to connect to start the game. The connected client must be notified that the server is waiting for another player to connect. Once another player connects, both players must be notified that the game has started. This means that if a player connects, and there is already a player connected waiting, he will be notified directly that the game has started.

4. When the game starts, the server must select a question randomly from the read *ArrayList* of questions, and send it to both players. The player that responds first with the correct answer takes the question points. The correct answer along with the name of the player who answered first is sent to both players. Each player can only send one answer. If neither of the two players provides a correct answer, then the correct answer must be sent to both players along with a statement that the question points do not go to any of the players (both provided wrong answers).
5. This process in step 4 is repeated for the 5 questions, making sure that no question is selected twice and 5 different questions are asked to the players in each game. Make sure to introduce a slight delay between each question and the next.
6. The server application must compute the points of each player by summing the points of the questions he earned. Note that each question has a number of points assigned to it.
7. As the game ends, the server must send to both players the results which include the name of the winner and the number of points he earned vs. the number of points for the loser.
8. The server application must be able to run more than one game at the same time, and must keep on taking clients connections and starting games between players.

- A new client that connects to the server should be assigned to a game. The details are as follows:
  1. When the client first connects, he should enter his name which will represent him in the game.
  2. If there is already a client waiting, the new client will be assigned to a new game with the waiting client. If there is no client already waiting, this client will be waiting for a client to connect. In either way, the client receives a message from the server indicating the condition and it should notify the user with one of the following messages "Waiting for another player to connect" or "Game has started, prepare for question 1".
  3. Once the game starts, the client must start receiving questions from the server, display the question to the user, read the answer number and send it to the server. Then, it must receive the result of the question which includes the correct answer and who got the question points.
  4. Before displaying each question, the client code should display the message: "Prepare for question number ….".
  5. At the end of the game the client application must receive the result and display it to the client console

Your submission must include the following:

1. Source code (.java files)
   Your code should include good modularization, coding style, and an appropriate amount of comments.
2. A simple report that includes the UML diagram of your application and a brief description of the most important methods in your code.