Network Programming
Fall 2020\2021
Course Project:
Client/Server Gaming Environment

Omar Mesheh
0174034

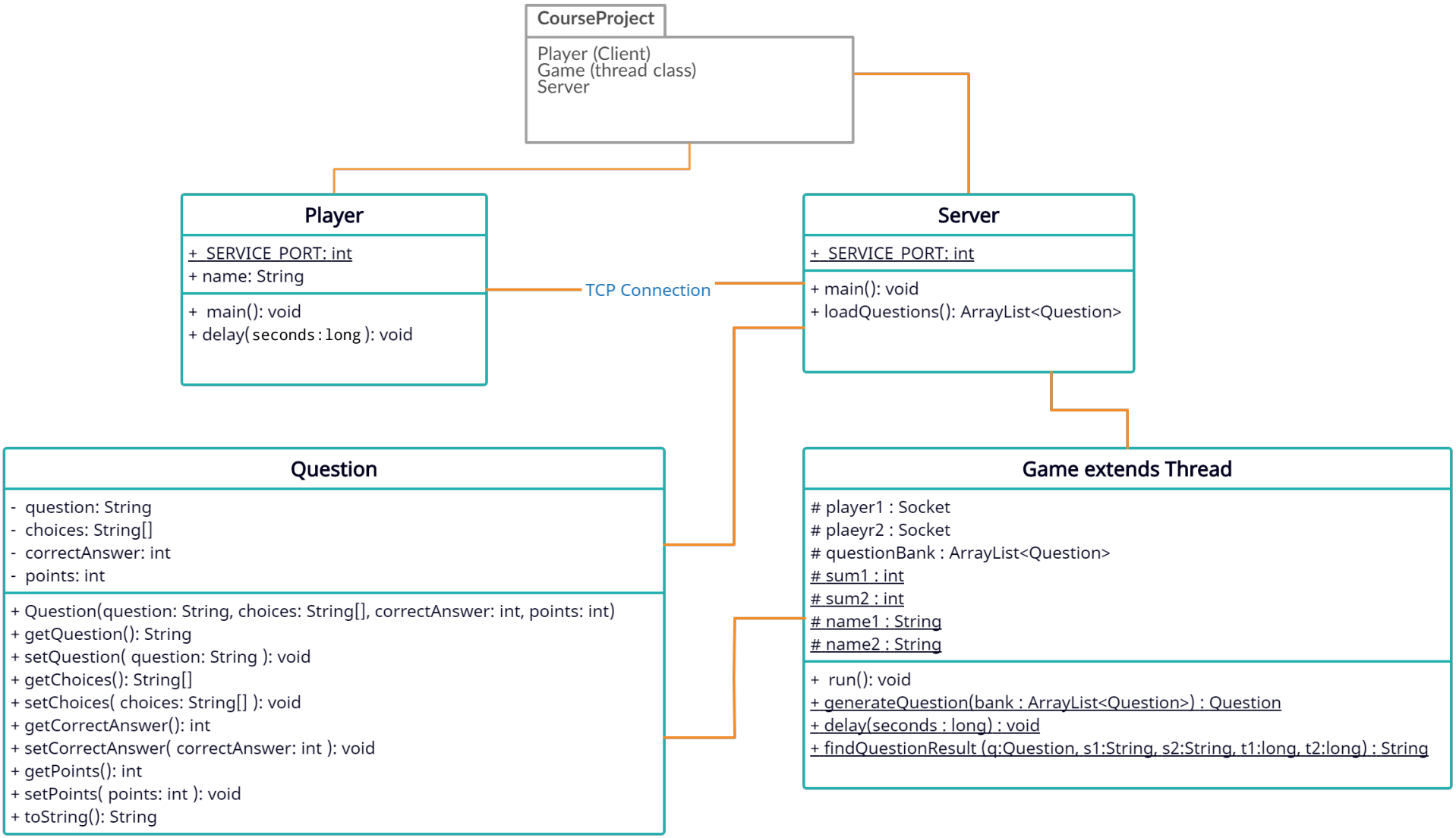

Figure 1: The UML diagram of the TCP/IP application.

• Introduction

In this course project, we were asked to build a network application that simulates a Client/Server gaming environment using TCP API in Java. The project consists of a client that is represented by a player, this player is connected to the server via **port 3000**. When two players are connected, the server will assign them to a new game and start serving them from this game as a new thread.

Now, inside each class I implemented some methods as follows:

I. in Server.java:

• ArrayList<Question> loadQuestions()

This method is responsible of loading the questions bank that is given inside the question.out file. This method is invoked one time only when the server is started.

• the main method:

In the main method of the Server class, a new ServerSocket is created, then, the server will load the questions bank from the ‘question.out‘ file using the loadQuestion() method. Now, the main method will start accepting connections from new players and assigning each two players to a new game. It will notify each player with an appropriate message based on the state of how many players are connected.

II. In Game.java:

• Question generateQuestion (ArrayList<Question> bank)

This method will choose a random question from a set of given questions called ‘bank‘, note that in order to ensure that a question is not picked twice, this method will remove the chosen question from the given bank. **Important note:** The original bank loaded from question.out file will not be affected by the removing process, since I am passing a copy of it to each created thread.

• void delay(long seconds)

This method will generate time delays in seconds. I used it to reduce the try-catch blocks in my code. Multiple delays with different periods is used to give the players a good experience while playing the game.

- **String findQuestionResult (Question q, String s1, String s2, long t1, long t2)**

This method will select the player who answered the question ‘q‘ first, the answers are passed as ‘s1‘ and ‘s2‘, and the timestamps of these answers are passed as ‘t1‘ and ‘t2‘ respectively. It will also add the points of that question to the player‘s overall score. If both players provided wrong answers, no points is added.

- **void run()**

The overloaded method of the Thread class. First, it will read the names of the two players, then, it will send the questions to both players as a Question object (to get advantage of the toString method). After this, it will receive the answers of both players packed with their timestamps of their responses. Now, the result of each question is generated using the ‘findQuestionResult‘ method and then sent to both players.

At the end of the fifth question, it will compare the overall points of both players and send the final results to them with how many points are collected by each player.

III. In Player.java

No special method is implemented here, but the main method will execute as follows:

It will first connect to the server at port 3000. If the player is connected successfully, he will be asked to enter his name. Now, this player will wait until another player is connected. The two players now will start receiving questions, answering them through the console and sending their answers with the timestamp of their answers to the server side through their sockets. **Important Note:** I am sending the timestamps in the same line of each answer because no priorities is set here, so errors could occur if we sent the answer and its time in separate lines.

At the end of the fifth question, the player code will receive the final result from the server and prints it to the console of each player.