Programming in Java

Assignment 3 — A Quiz System

2013-14

1 The problem description

You are asked to design and implement a simple on-line quiz game using Java RMI. This assignment is to be carried out individually.

2 The quiz game

The quiz game system should consist of a central quiz server and two separate client programs: a *set-up client* and a *player client*.

The *set-up client* program should enable a user who wishes to organise an electronic quiz game to create a new quiz (i.e., a set of questions) on the server, with a given name for the quiz, and a set of possible answers for each question. This will return a quiz game **id**. At some point in the future, this client should be able to close the quiz game, quoting the game id. The outcome will be a notification of the winner together with full player details (which should be persisted on the server).

The *player client* program should enable players to play one of quizzes available on the server.

- Firstly, the program should enable players to see a list of current quizzes and select one.
- The players should then be able to answer each question of the quiz by choosing their answer amongst those suggested (i.e., multiple-choice).
- At the end of the quiz the quiz server should return the score of the player for this quiz.

The *quiz game server* should deal with (potentially concurrent) requests from the two client programs and maintain the state of the various ongoing quizzes in a consistent manner. The server should make use of persistence to store the quiz questions and the results of the various user attempts.
Please note: you are **not** expected to provide a sophisticated user interface for the two client programs (a simple text based interface will suffice).

Your grading will depend on the functionality of the system you provide; the above is a *barebones* description of the system and you should add additional functionaility as you feel appropriate.

3 The deliverables

The assignment must be pushed to a project called `Quiz` on your GitHub account. It will be automatically cloned on the due date. You should upload your classes, documentation, and unit tests, in your submission. You should include full javadoc for your classes, and a `README` file detailing how to run your system.

You are encouraged to leave everything ready well in advance – both the programming and pushing it to your GitHub account – to avoid last-minute problems (e.g., GitHub may be down for maintenance). The assignment will be graded according to its ability to fulfil the above scenario; the simplicity, clarity, and generality of the code (including succint but illustrative comments and JavaDoc); and the compliance with good practices of coding, and version control as outlined during the module (e.g., commiting often and in small pieces, use of descriptive commit messages, committing only source code and not binary or class files).

Regardless of the times you choose to push your changes to GitHub, you should commit early and often. In case of suspected plagiarism, your version control history will be used as additional evidence to judge the case. It is in your best interest to commit very often (and to use adequate commit messages) to make it clear that the process of creation is entirely your own.

4 Credits

KLM can't remember where the basis of this coursework came from but hopefully he will shortly!