Project 5
CS 342 Trivia Game

Team #18
12:30 PM Section

William FitzGerald	**wfitzg2@uic.edu**
Christian Cardenas	**ccarde8@uic.edu**
Ugnius Rumsevicius	**urumse2@uic.edu**

Project Description

This project consists of an implementation of a 4-player CS 342 (Software Design) trivia game! A random question pertaining to topics discussed in CS 342 will be given to each client every round and the clients will each have the same multiple choice selections displayed in their GUI. Questions and answer choices can managed in the server window. These questions will be served to all connected clients.

The game logic will keep track of who correctly and incorrectly answered and their points will be reflected to the clients GUI as well. When 10* rounds has been played, the players will be ranked based off points and the server will notify all the players the rankings.

*Actually number or rounds to be determined

Server GUI:
This will allow you to manage all questions. You can add new questions and create four choices to be presented to clients. You must also choose which option is correct.

Client GUI:
Once you connect and a game has been started, you will be served a question with four choices. Once all clients have answered, you will be served the next question. The game continues until all question have been asked. At this point, a ranking of all clients will be displayed to show how well you did compared to other players.

Networking:
We plan on reusing the network base from project 3, but simplifying it down since clients don't need to send information to other clients, just to the server.

Server will wait for connections indefinitely, adding client threads to an arraylist, communicating and updating them with the Observer design pattern, just like in project 3.

Clients will send network objects to the server that contain information such as: individual player score and what the users choice was for the round.

The server will send back information to each client such as: all the individual scores, whether or not a client's answer was correct, end of round, and so on through either a similar network object, the update method, a mix between the two, or an unforeseen workaround.

We will use Java to implement the game logic and JavaFX to create the GUI.