**Assignment Report**

**Module Name**: Advanced Programming

**Module Code:** CE301

**Student ID:** 1805945

Functionality Table

|  |  |
| --- | --- |
| **Function** |  |
| Client establishes a connection with the server | Yes |
| Client is assigned a unique ID when joining the game | Yes |
| Client displays up-to-date information about the game state | Yes |
| Client allows passing the ball to another player | Yes |
| Server manages multiple client connections | Yes |
| Server accepts connections during the game | Yes |
| Server correctly handles clients leaving the game | Yes |

Protocol Outline

|  |  |
| --- | --- |
| **Client Request** | **Server Response (on success)** |
| (On connection) | “You have joined the lobby.” |
| ID | <player ID> |
| LOBBY | <list of players> |
| HOLDER | <ball holder ID> |
| PASS | (Change ball holder)  (Add pass acknowledgement message to each player event queue)  <pass acknowledgement> |
| EVENT | (Remove entry from player event queue)  <next game event> |
| (On disconnection) | (Add player disconnected message to each player event queue) |

Client Threads

* **Main Thread** – created when the player program is run and presents the user interface to the player. Delegates functionality for communicating with the server to the **Player** class. This thread terminates when the player program is closed.

Server Threads

* **Main Thread** – created when the server program is run and listens for incoming player connections. A new thread is instantiated for each connected player. This thread terminates when the server program is closed.
* **Player Handler Thread** – created when the server program establishes a connection with a player. Processes requests from that player and sends back the appropriate responses. Some requests will also change the game’s state. This thread terminates when the corresponding player program is closed.

Project Review

* ***How did it go?***

Overall, I would say it was successful. Although I spent a lot more time on it than I expected, I gained more knowledge on Java APIs.

* ***Which parts were easy, and which were challenging?***

Implementing the server-side so that it works with PuTTY was straightforward. However, having the client programs communicate with each other so they can display the current status of the game became extremely challenging.

* ***Are there any features in particular I am proud of?***

I am most proud of the methods and data structures used that allows players to broadcast events such as joining or leaving the game to every other player. I am also happy the quality of the user interface.

* ***Are there any problems with the quality of the program***?

None that I can think of.

* ***Is there anything I would do differently next time?***

When the list of connected players becomes large, it should be printed on separate columns rather than on a single one.