# SYSC 3303 – Assignment 2: Game Client / Server

By: Nishant Bhasin

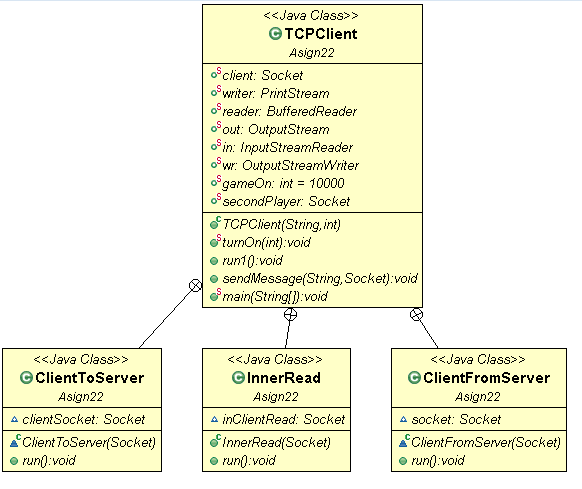
# Background

This client server application allows multiple clients to connect to a pre-threaded game server (Asign2.2 Package) and play a simple word collecting game in which a client can create a game with another client joined to the same server. The client / server is synchronized using queues and conditional variables.

The server being pre threaded means that connection to the client is limited to the number of threads initialized by the server. If any other clients tries to connect to the server while the threads are busy then it has to wait for its turn in the queue and as soon as a client exits the thread becomes free and the first client in the queue is assigned that thread (FIFO).

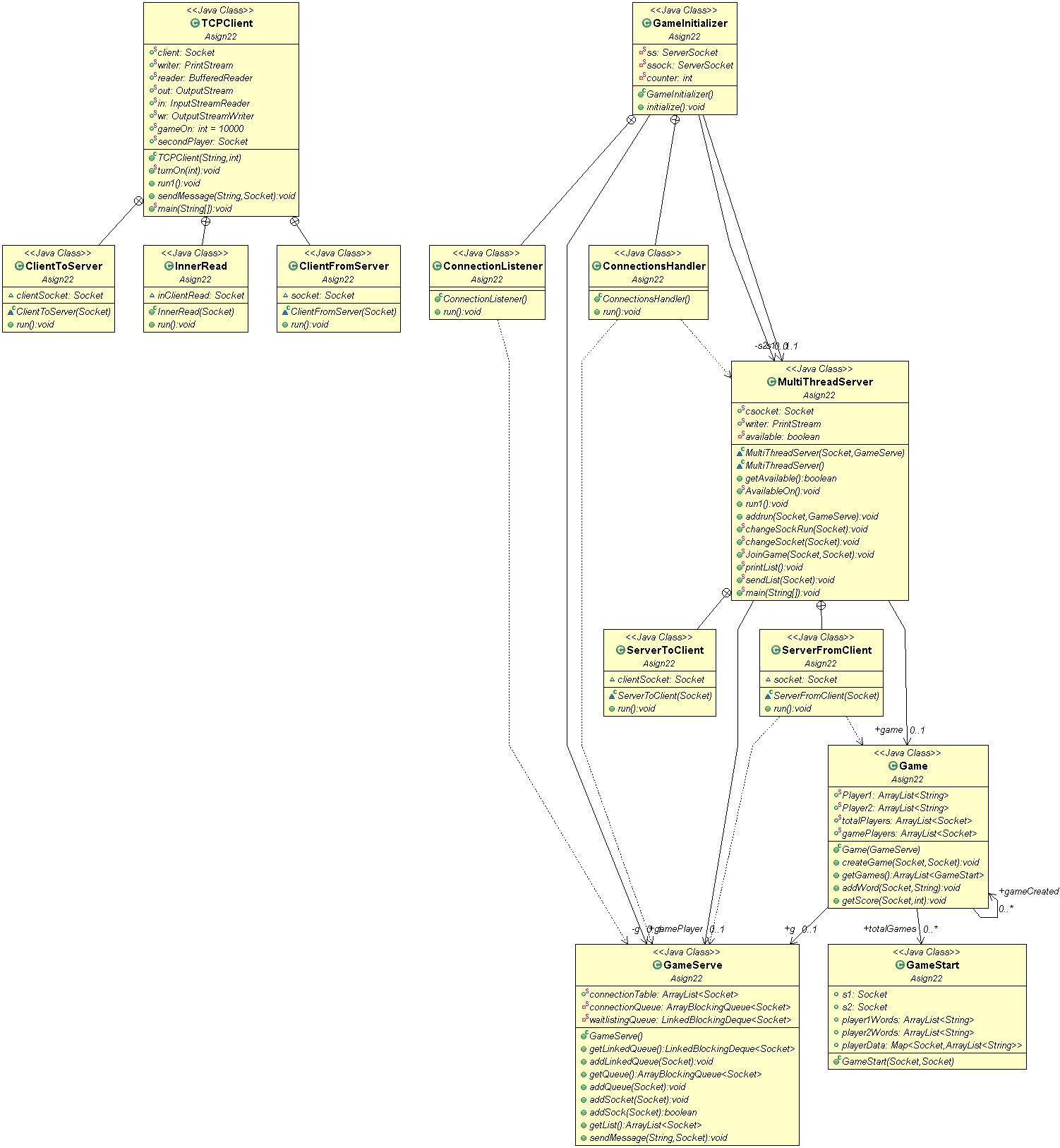
# The UML Diagram – Code Description

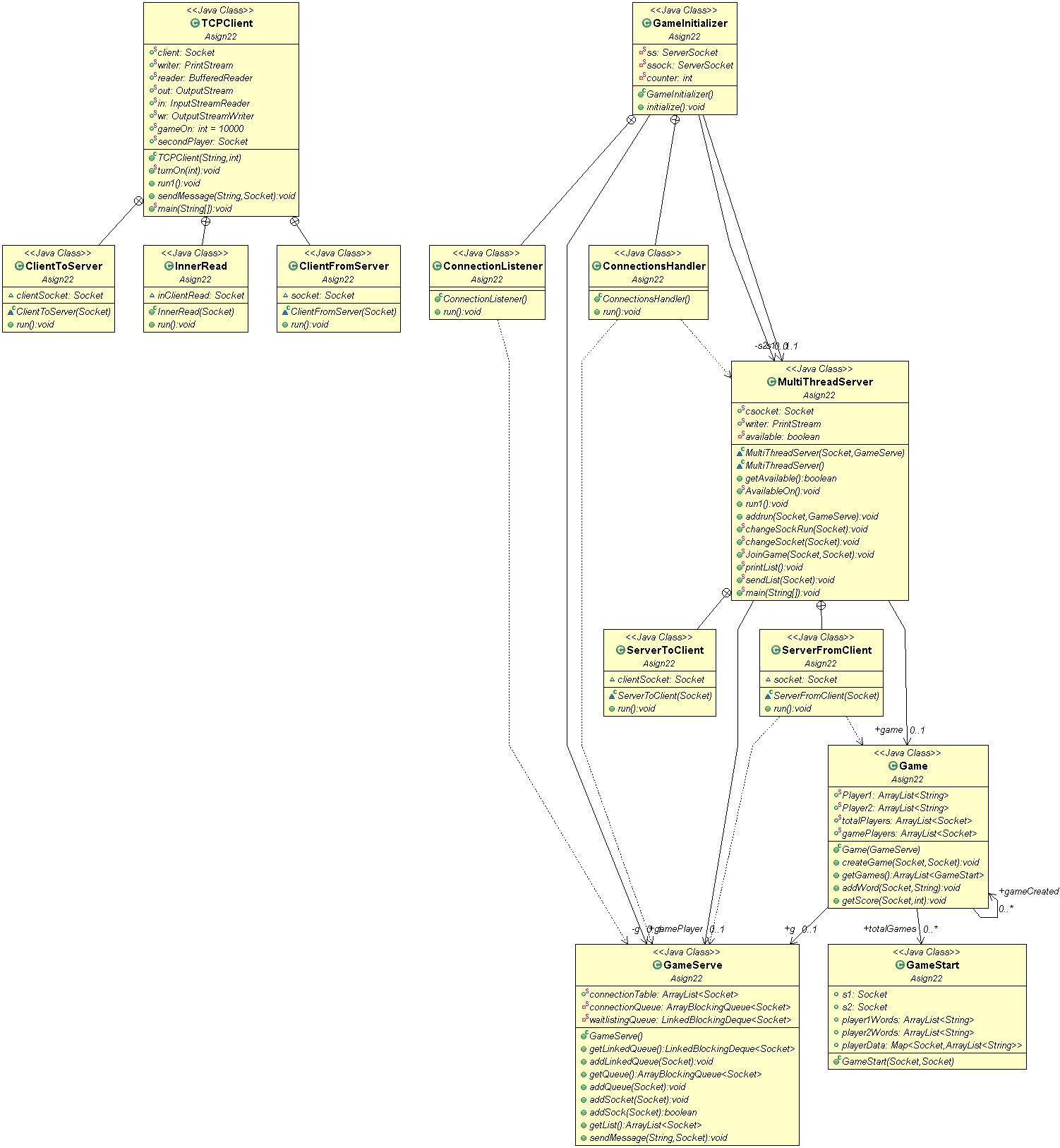
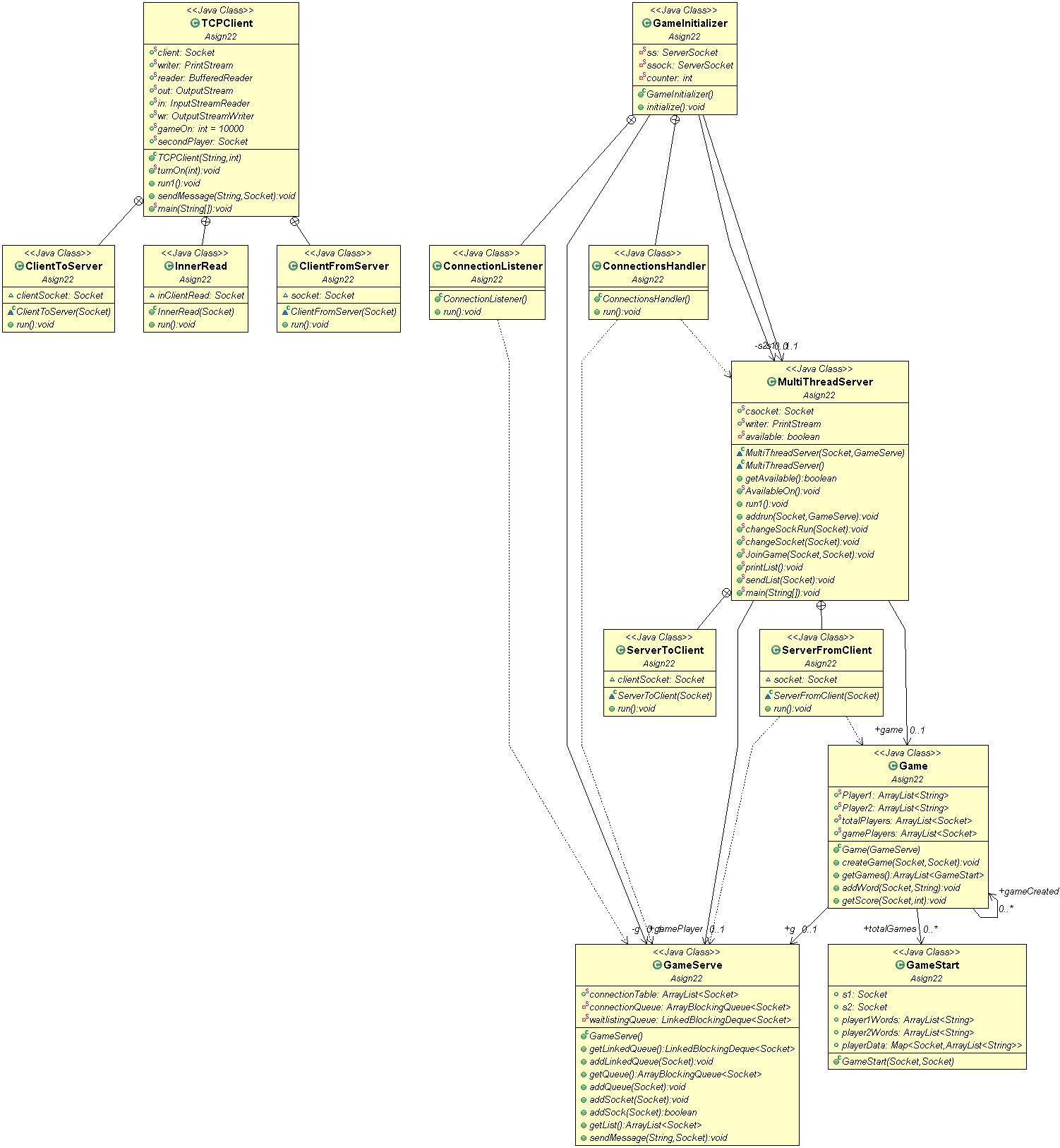
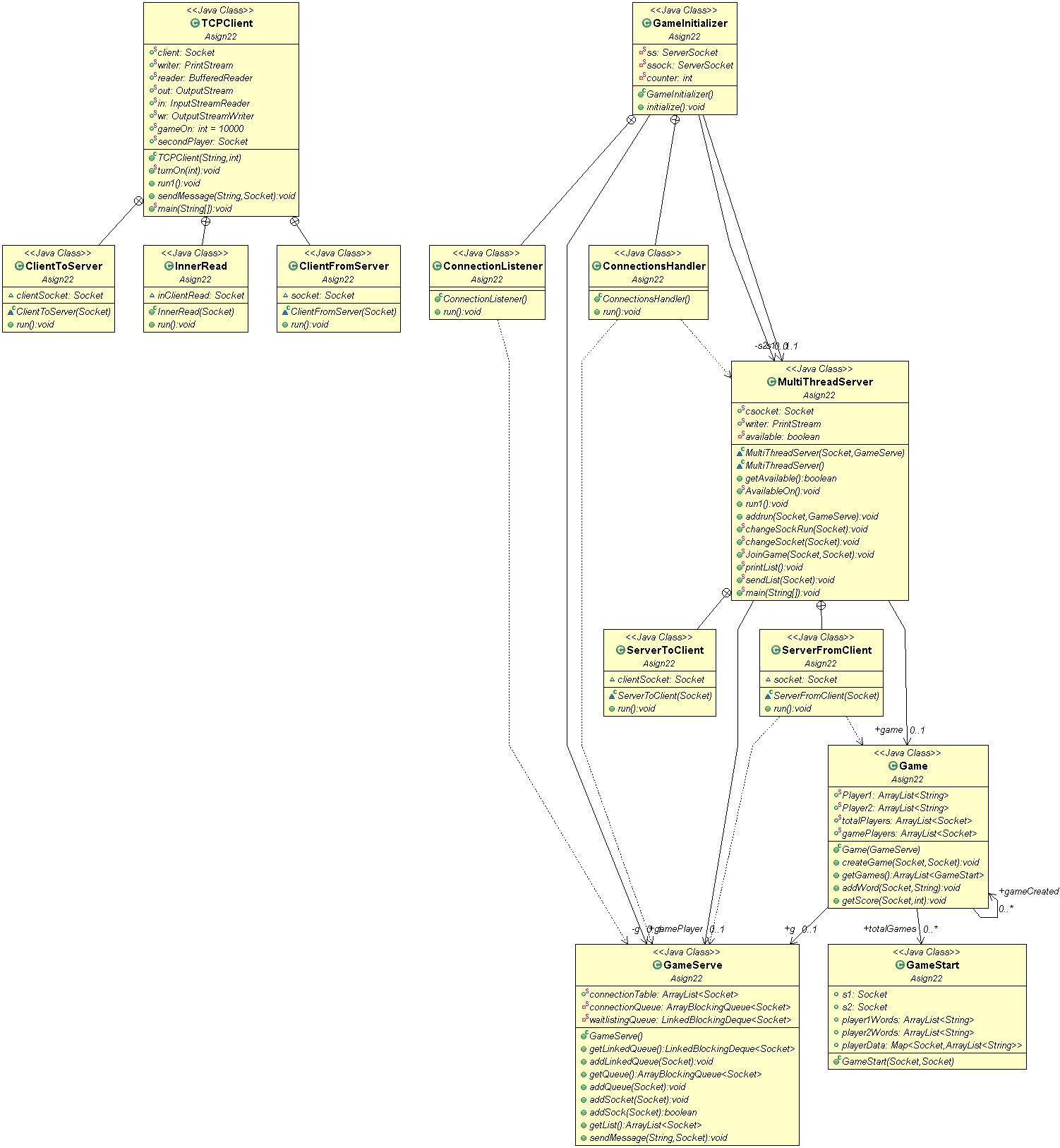
**3.1 The TCPClient**

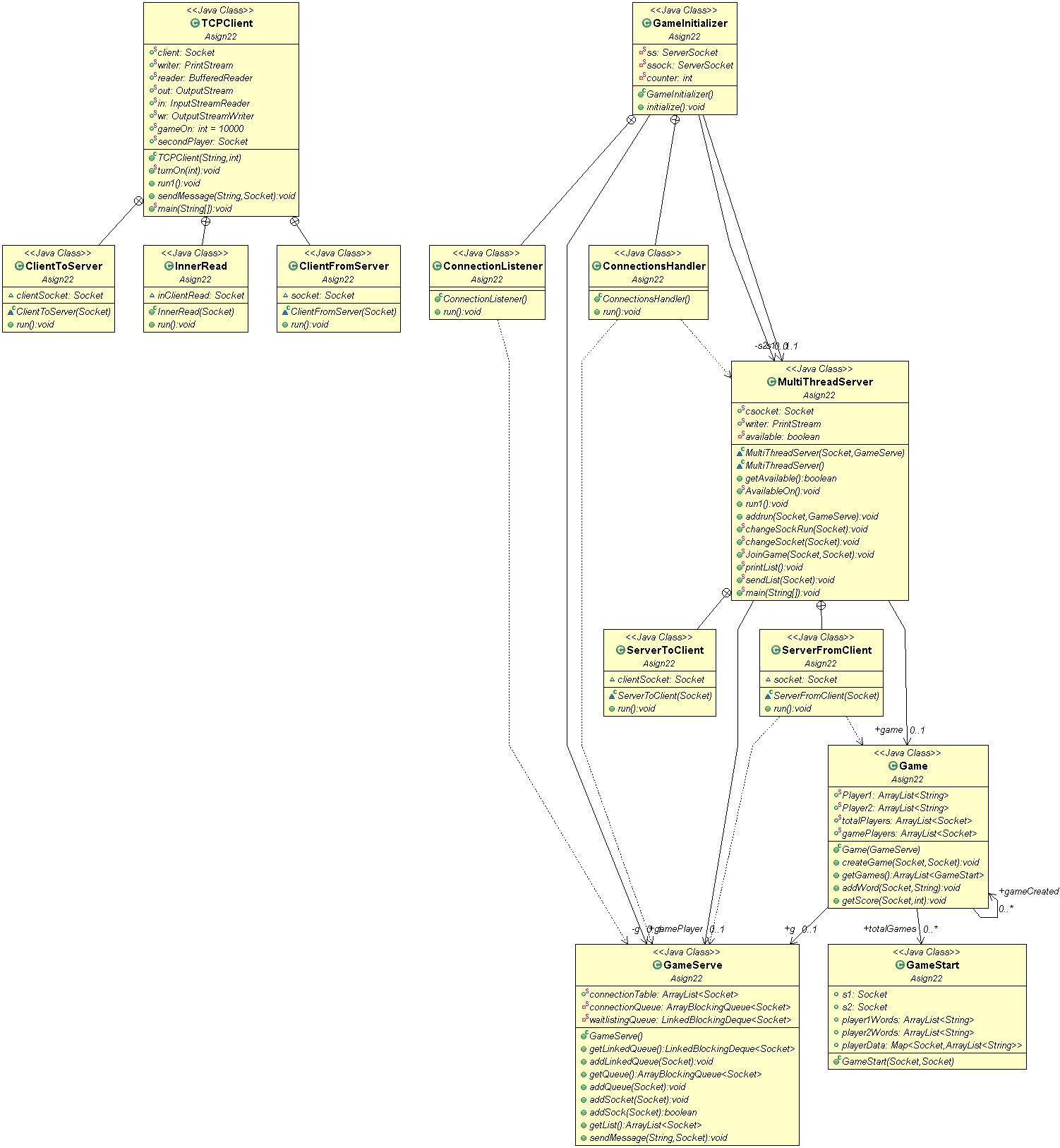


The TCPClient connects to the multithreaded server, creates a game / joins a game and adds words to the server for storage.

The three inner classes ClientToServer, InnerRead and ClientFromServer extends TCPclient which extends thread and run simultaneously. The reason they need to run at the same time is because if you are trying to send a message to the server then you cannot get a message until your message is delivered to the server but with simultaneous running ClientToServer and ClientFromServer we only need to pass the socket to these classes and they take care of all the outgoing and incoming messages between server and client. InnerRead was made but never fully implemented in the design.

**3.2 Server / Other Classes**

****The server contains a socket csocket, writer and available: Boolean (condition variable). This condition variable tells if the server is free to accept another connection from client or not. Once the connection is accepted and the available variable becomes false i.e. no more connection will be accepted by the server. If the connection comes while variable is false it goes into either connectionQueue or waitlistingQueue in GameServer class which keeps track of all the incoming connection from the clients. The server also has a ServerToClient and ServerFromClient inner class which extends thread and run simultaneously. Its done for the same reason as discussed in previous section that we need uninterrupted communication between client and server.

****

The GameStart class keeps track of two players playing game and adding words to the buffer i.e. player1Words and player2Words which are bother ArrayList of type string. The game initializer on the other hand creates a new instance of the game using connectionsHandler and connectionsListener. The connectionsHandler handles what to do with the new incoming connections from the client i.e. whether to send it to the queue or send it to the server. If the server is free (available = true) the connection is send to the server otherwise its send to GameServer.connectonQueue where it waits for its turn. The connectionListener accepts incoming connections from the clients.

# Testing and Database implementation

**4.1 Database**

Under the package 2.3 you will find that I have implemented a database. The main difference between package 2.3 and 2.2 is that 2.2 does not contain text database and uses conditional variables for synchronization whereas for package 2.3 the server creates new thread per connection hence there is no need for synchronization. To get a better idea of how the database will be implemented kindly run package in section 2.3 with two clients and refer to the database text file created by it.

**4.2 Testing**

There is a testclass in package 2.3 which talks about the testing that will be done in assignment 3.

TEST PLAN :-

1. TYPES OF TESTING :-

- Stress Test

The stress test is a series of test that will try to fill the buffer as much as possible by having multiple clients connected to the server and filling sending words to it and eventually try to crash the program.

- Concurrency Test

This series of tests deals with maximum number of games that can be played with out system and will test if the system is capable of handling simultaneous running games of its flawed.

- Database Test

This test involves filling in the buffer and see when it gets emptied by the server to print to the database.

- Misc. Test

Series of test to see if the components of the server, gamestart etc. are working properly or not. Checking the pre condition and post condition of various methods in different class.

- Maximum Connection Test

This series of test will deal with how many connections can the server handle. If the waitlist queue is working properly and what happens if the connection goes offline in the middle of game. Can our server handle that kind of condition or not.

- Buffer Empty Test

Empty the buffer while clients are trying to fill in the buffer.

- Buffer Filling Test

The maximum amount of words which you can add in the buffer and while clients try to empty the buffer.

All these test mentioned above will be done in assignment 3.

# References

* [1] Assign2.gif – UML for package 2.2
* [2] MultiThreaded.gif – UML for package 2.3