

CIS5003 – Software Carpentry

Parastoo Porhonar st20136791

Assessment Title: Implementation of an interactive multiplayer quiz program using client-server architecture

# Description

This project is the demonstrating the design and implementation of IMQQuiz an interactive application using a client-server architecture simulating a quiz program.

The program has a server side that listens for clients; once a client connects and is authenticated, it is provided with some options to answer the question, choosing an option sends a response back to the server.

The server software component is designed to listen to a given port and can serve requests from multiple clients. The expected functionality of the server and client components are given below.

Server:

- The server is initiated by establishing a server socket on a given port number.

o The port number can be given as a command line argument

- The server should have the facility to read and parse a given XML game script. The filename should be given as a command line argument (more details on XML game script given below)

- The number of players accepted for the game (N) is indicated in the game script

- The server should facilitate up to N client connections. i.e., The server should accept connections from N number of players who will be able to play the game simultaneously.

o Hint: Each client connection to the server should treated as a separate thread.

- Once a client is connected, the server performs the following:

o Greets the client with a welcome message

o Collects the details of the player (e.g., first name, last name, age etc)

 The server could send messages to the client requesting data (first name, age) and the client can reply with user inputs

- The server should wait until all N clients are connected before the game starts

- Once all N clients are connected, the server initiates the quiz program.

Loop: For each question given in the XML game script:

o The server sends the question and MCQ answers given in the XML game script to all connected clients

o The server gives clients T number of seconds to answer the question.

o If the client answers before the time T, the server checks the answer, computes the score and moves on to the next question

o If the client fails to answer, the server sends a suitable message to the client and moves on to the next question

End Loop

- This process repeats until all questions given in the XML game script are presented to the client.

- Once all the questions are presented and answered by the client, the server computes the total scores for each client and communicates them along with the result (whether the client has won or lost the game)

- Finally, the server closes the game and disconnects all the clients

Client:

- Once the server is initiated, clients can connect to the server using the server’s host IP address and port number.

o These should be provided as command line arguments when initiating a client.

- Once connected to the server, each client gets a greeting message from the server, followed by requests to provide player details (i.e., name, age etc.)

- Once all clients are connected, the server will start sending the questions with multiple choice answers.

o Questions are displayed in the terminal standard output.

- Clients will then answer the questions within the given time window (T seconds).

o Answers are given as user inputs through the terminal

# Design

The overall design of the program was split up into two components, one for the server and one for the client. These two components interact with each other using a protocol. The UML diagram in the figure below shows how the main components of the software interact.

- UML IMQ use case diagram

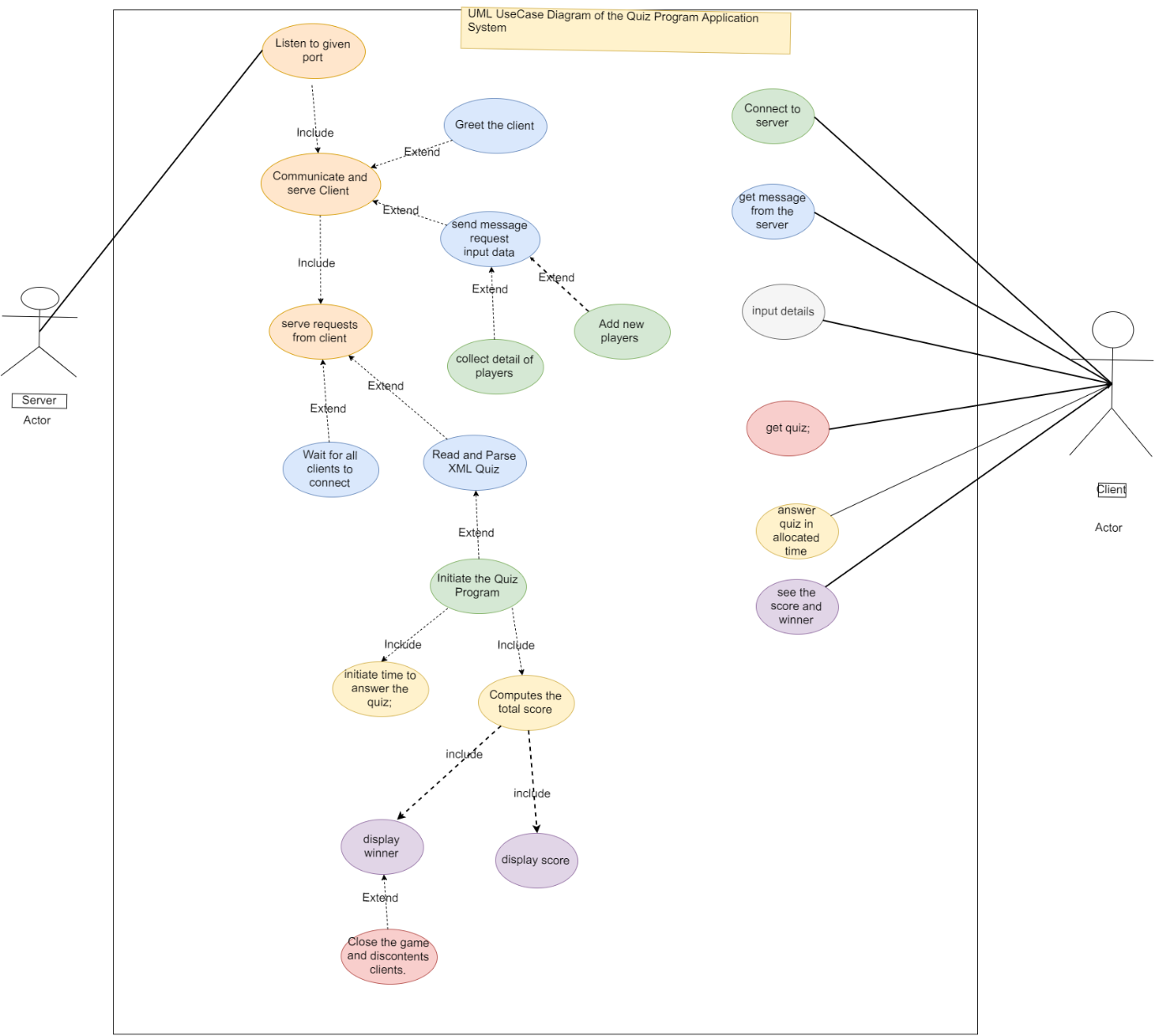
The server actor initiate communication by establishing a server socket on a given port number.

## Server-side Use cases

* Listen to given port
* read and parse a given *XML game script*.
* Serve the client and able to facilitate up to *N* client connections and accept connections from *N* number of players to play the game simultaneously.
* Greets the client with a welcome message
* Collects the details of the player (e.g., first name, last name, age etc)
* initiates the quiz program.
* computes the total scores for each client
* display score
* display the winner
* close the game and disconnects all the clients

Client-Side Use cases

* Clients can connect to the server
* Gets a greeting message from the server
* Provides player details (i.e., name, age etc.)
* Get Quiz Questions
* Answer the questions within the given time window (*T* seconds).
* Get the score and winner
* Disconnect the server

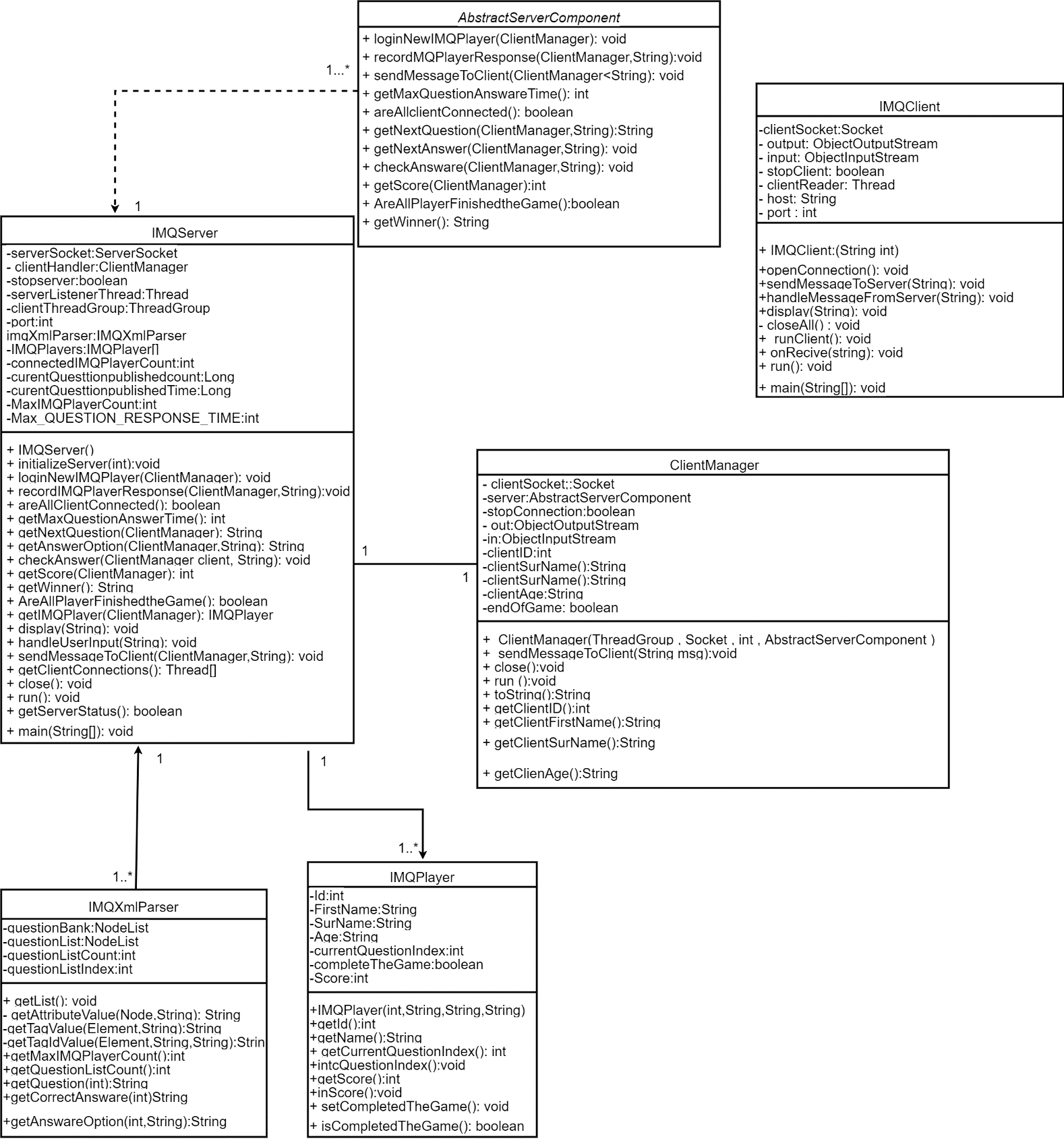


While the server initiates the quiz client can start to play

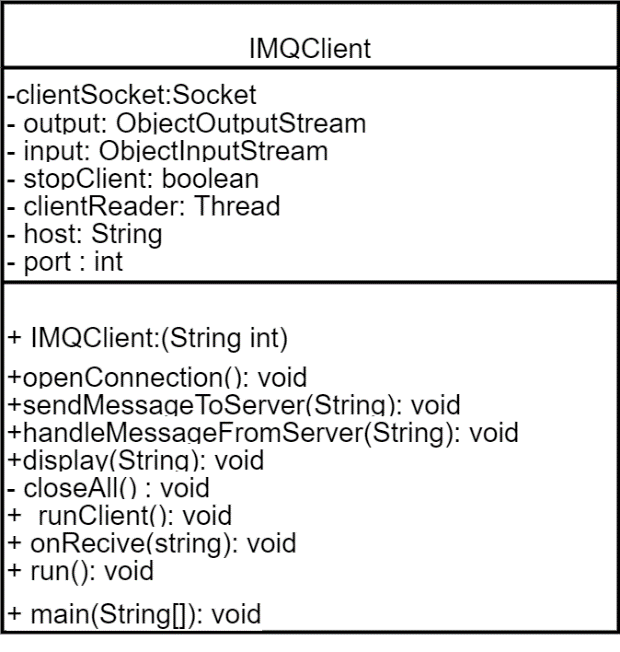
When client and server are connected client receive welcome msg

## - UML IMQ class diagram

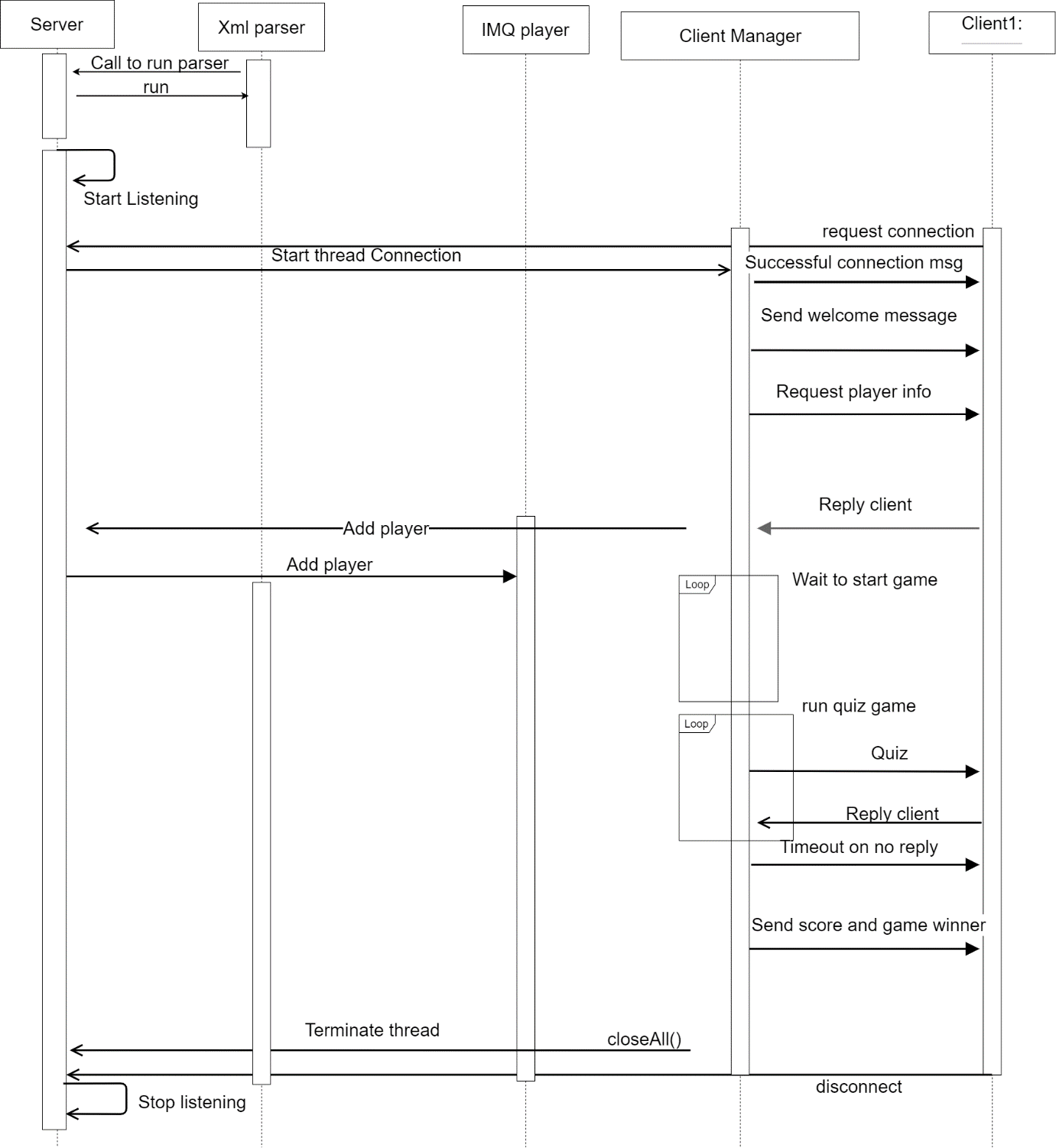
**Server-side class diagram**



**Client-side class diagram**



**-** UML sequence diagram(s)



**Server-Side Architecture**

Server component consists of these sub-components; the main sub-component that provides required classes running the server and listening and accepting connections,

### Server

The server side has a main class named Server; it handles the processing and error checking of the input parameters. This class must be provided the port number on which the server should be run. Additionally, the list of players in the class passes to the Client Manager class. It will get thee questions and answers from pare class and run it.

The Client Manager class is responsible for handling the clients inputs from the user; in this case the server.

When a client tries to connect to the server on the specified port, a new instance of a Thread is created, which can accept a connection from a client, start a thread for that communication, and continue listening for requests from other clients a multi-threaded server program.

Server is listening and waiting for the clients to connect When the client attempts to connect, server accepts the connection. thread listens to the port and creates client connections. Each connection is treated as a separate thread and each client is associated with the Thread Group.

### Client

There is a main class named Client that handles the creation of the client.

The main thread of the client will listen for messages\* from the server. The first message will be a "WELCOME"\*. then requesting for information of the player such as first name ,sure name and age.

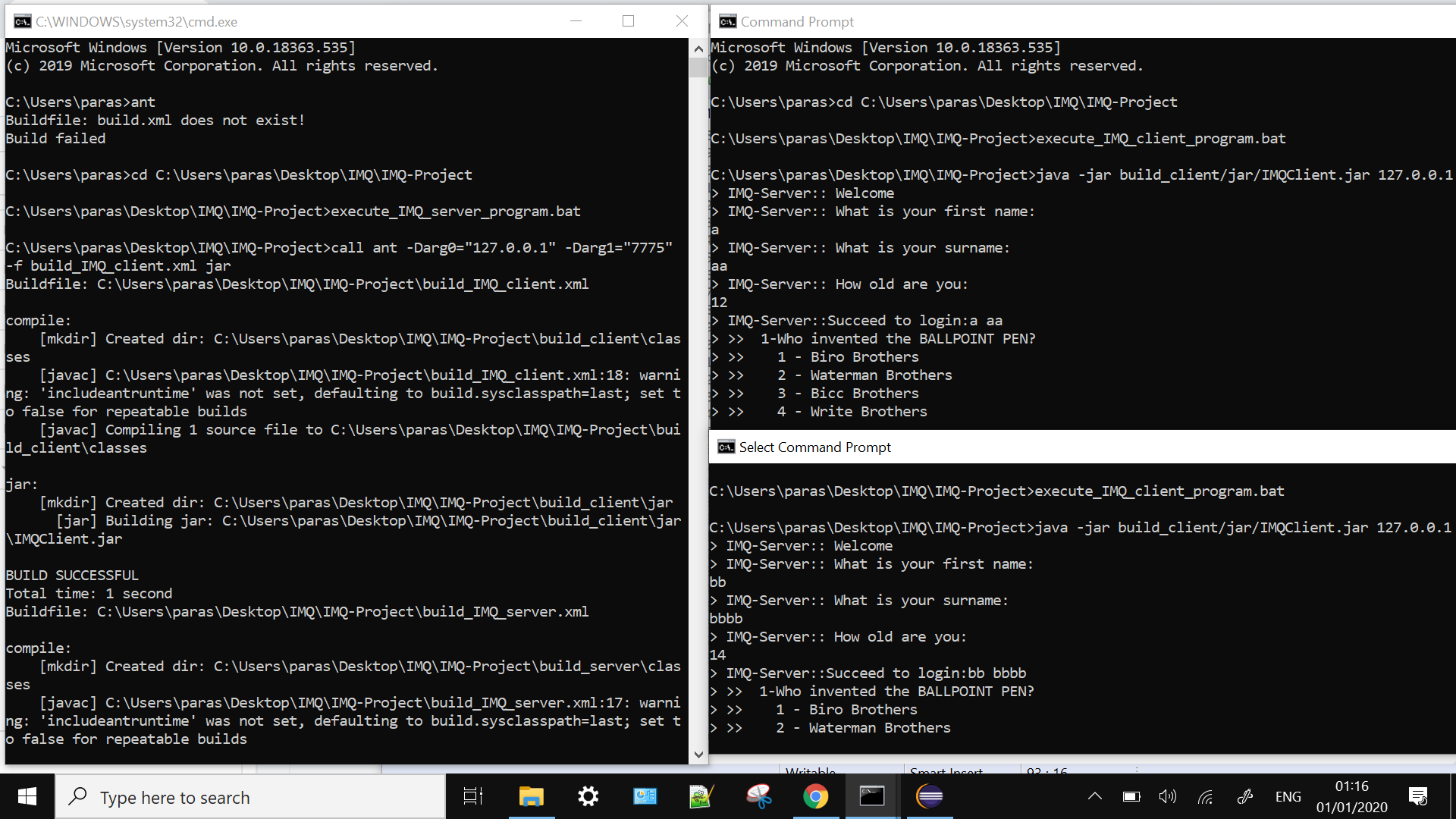
Then we go into a\* loop listening for question.

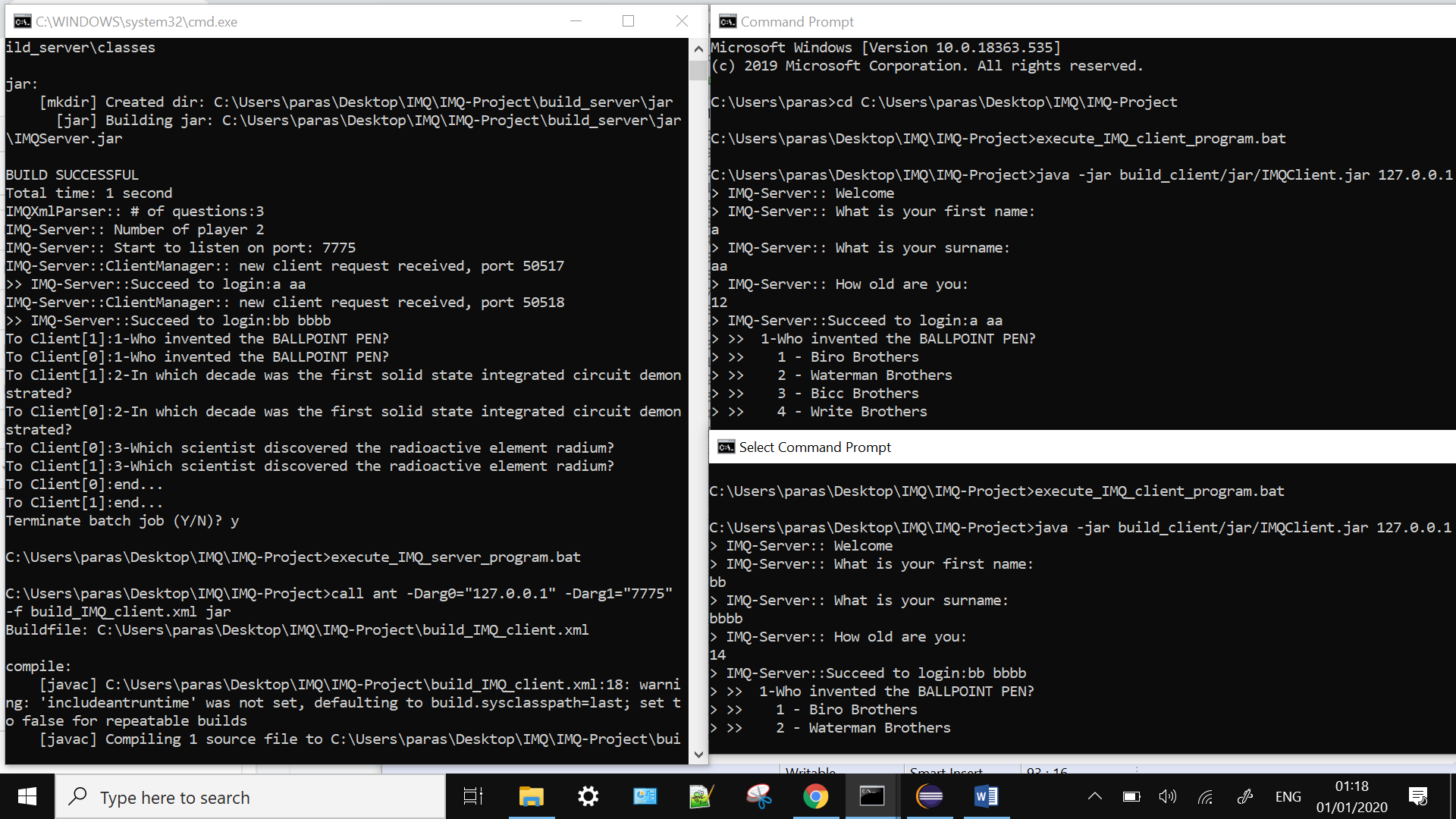
Client-Side Program

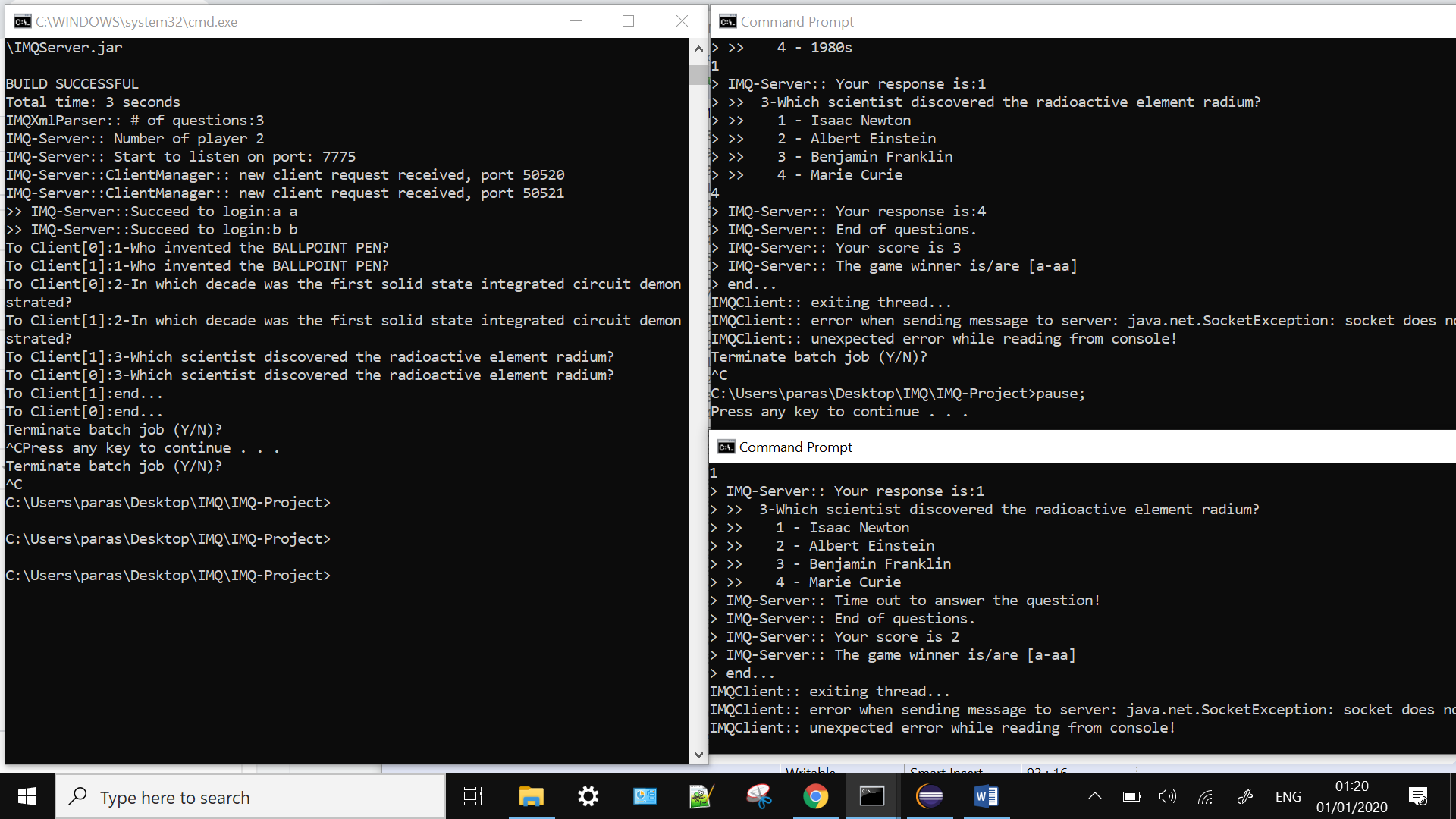
The client program establishes a connection to the server program on a particular host and port number in its Socket method and thread for connection.

The Socket method first creates a Socket object with the port number (7777) where the server program is listening for client connection requests. It also creates a Buffered Reader object to read the text sent by the server back to the client.

Below is the screen print of the running program.







# Usage

The [server jar file](https://github.com/farhan3/java-clicker-quiz/blob/master/server) and [client jar file](https://github.com/farhan3/java-clicker-quiz/blob/master/client) can be found in their respective directories.

Start the server by running the following command:

ava -jar build\_server2/jar/SimpleServer.jar 7775

Start the client by running the following command:

java -jar build\_client/jar/SimpleClient.jar 127.0.0.1 7775

**Git repository on Bitbucket**

git clone https://Paras2P@bitbucket.org/Paras2P/imqprogram.git