DS A-1 Report

By Tianchen FAN 1166401

To run the program:
Use openjdk 20.0.2;
For server, parameters are: port(int number - port to listenon), time in seconds (int number (optional, with default 60), time that the cookie expire);
For Client, parameters are: IP address (String, address to set up communication), port (int number, port to set up communication).

The problem context in which the assignment has been given.

In this assessment, we are required to develop a multithreaded dictionary server and the corresponding client which allows clients to query(check)/add/update/remove word(s) in the dictionary through the server.
In order to reach the goal, I develop two softwares, server and client. The client will communicate with the server via **TCP** to send requests to the server and get responses from the server. The server is multithreaded, which means that it could handle more than one request parallelly at the same time.

A brief description of the components of the system.

The whole system consists of two parts, the server and the client. They would be introduced separately.

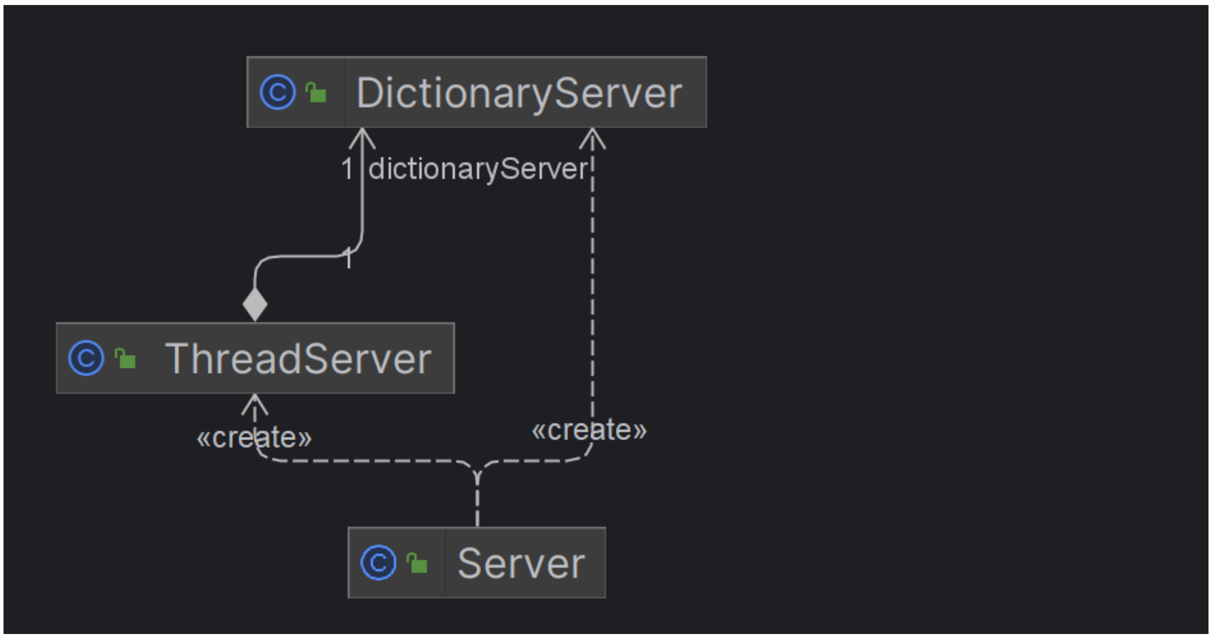

Figure 1. Components of the server.
The server part has three main components, including Server, ThreadServer and DictionaryServer.
Server, the entry point of the server part. It would accept up to two numbers when it's launched. The first number is the port the server would try to listen on. The second number is optional and stands for the time in seconds that the login of the user would expire. The threading architecture of the server is **thread per connect**. So, when a new connection is coming in, Server would create a new thread and ThreadServer to handle the connection.
ThreadServer, the component that handles that request. It would check the type of the request and then make the corresponding request to the DictionaryServer and structure the returned result and send it back to the client.
DictionaryServer would interact with the dictionary file. It is shared by all ThreadServers to make sure that the change made by ThreadServer a could be reached byThreadServer b.

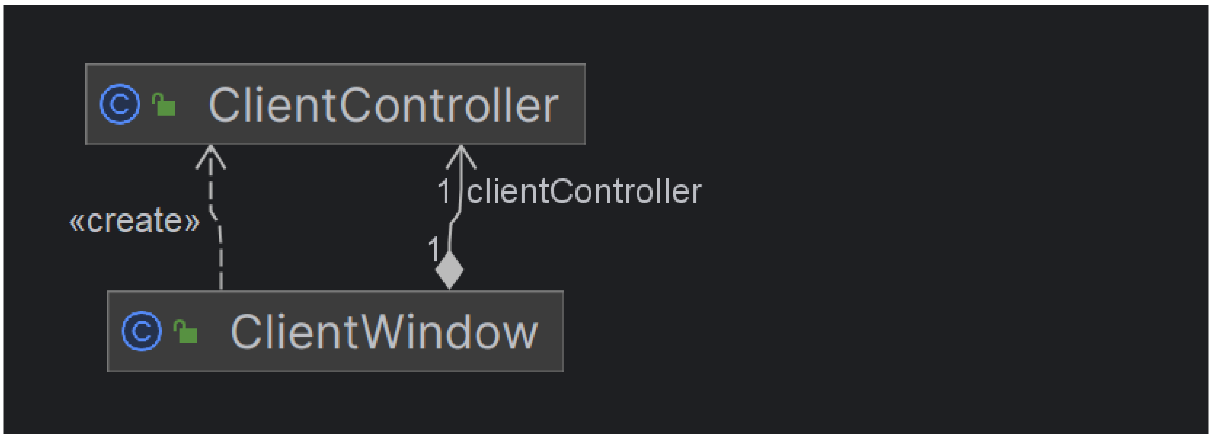

Figure 2. Components of the client.
ClientWindows is the entry point of the client program. It is the component that decides the showing content in the UI. It would check the input from the user and send it to ClientController to send to the server. It would get the result sent back by the ClientController to display them to the user.
ClientController would get the text from the the ClientWindows and set up the communication with the server.

An overall class design and an interaction diagram.

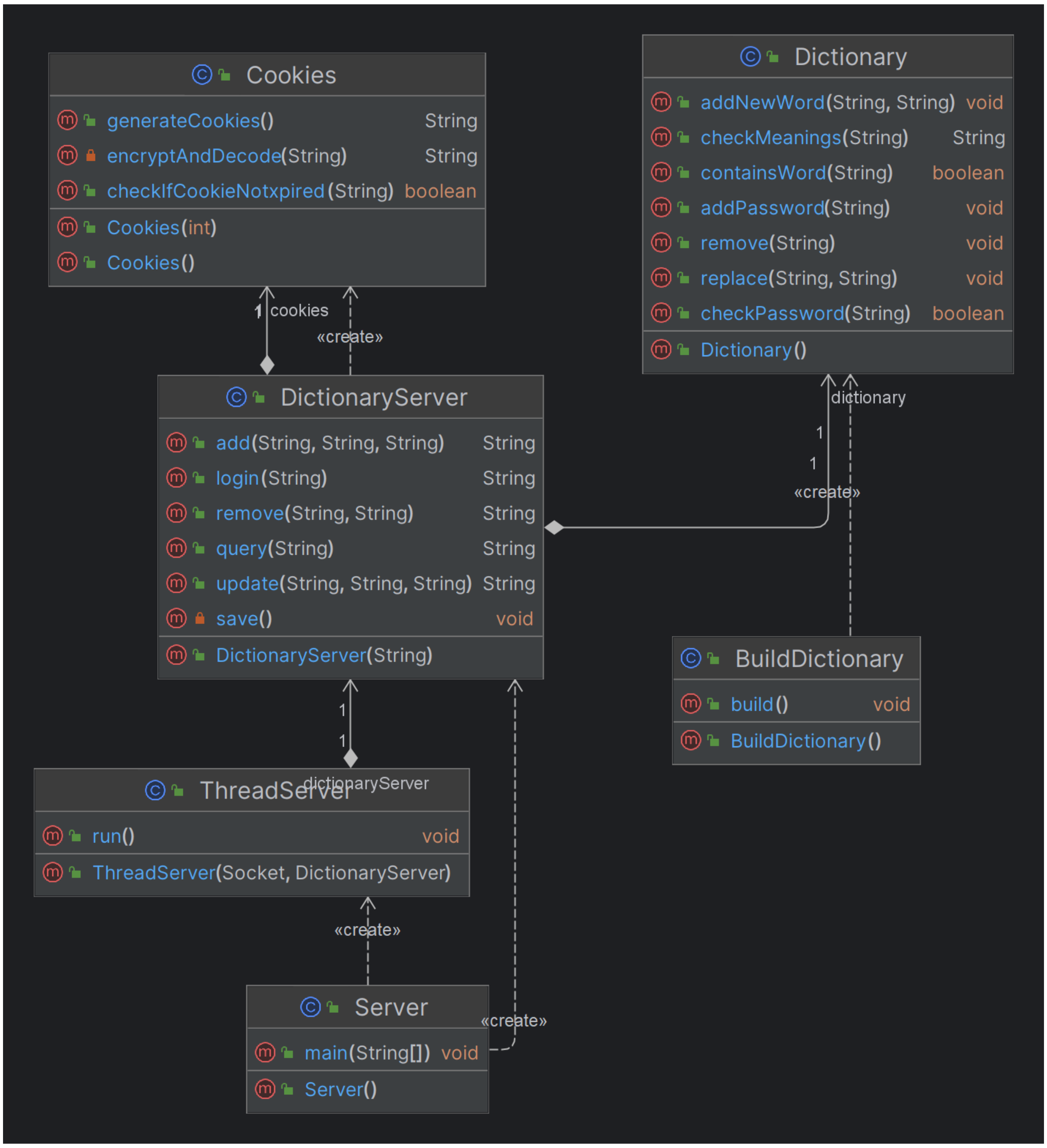

Figure 3. Class design of the server part.

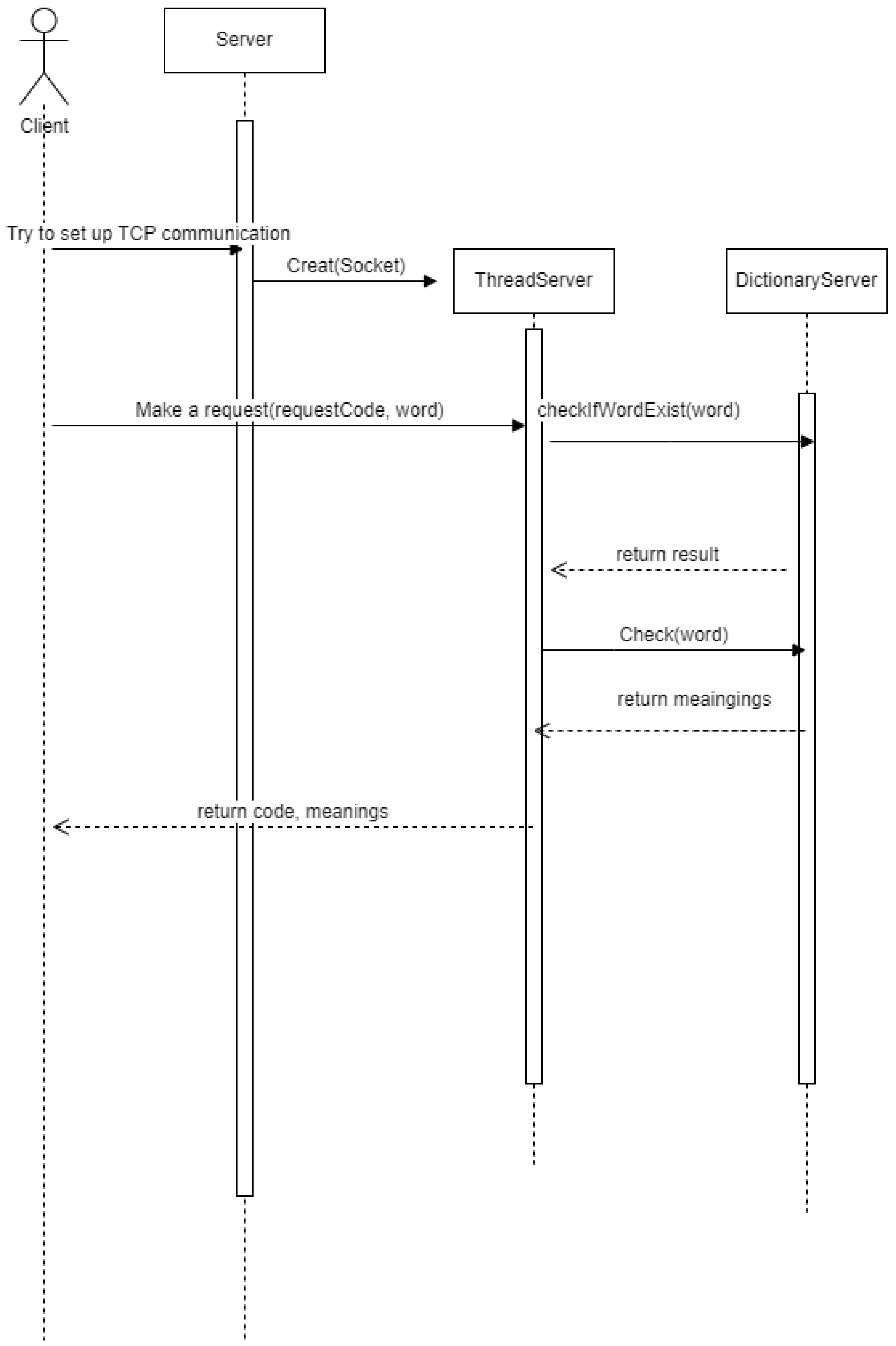

Figure 4. interaction diagram of query in the server side.
It is the success scenario that the server handles the query request from the client. An alternative scenario is that DictionaryServer replies that there is no such word in the

dictionary. Then ThreadServer would directly send the client the message that the word is not found.

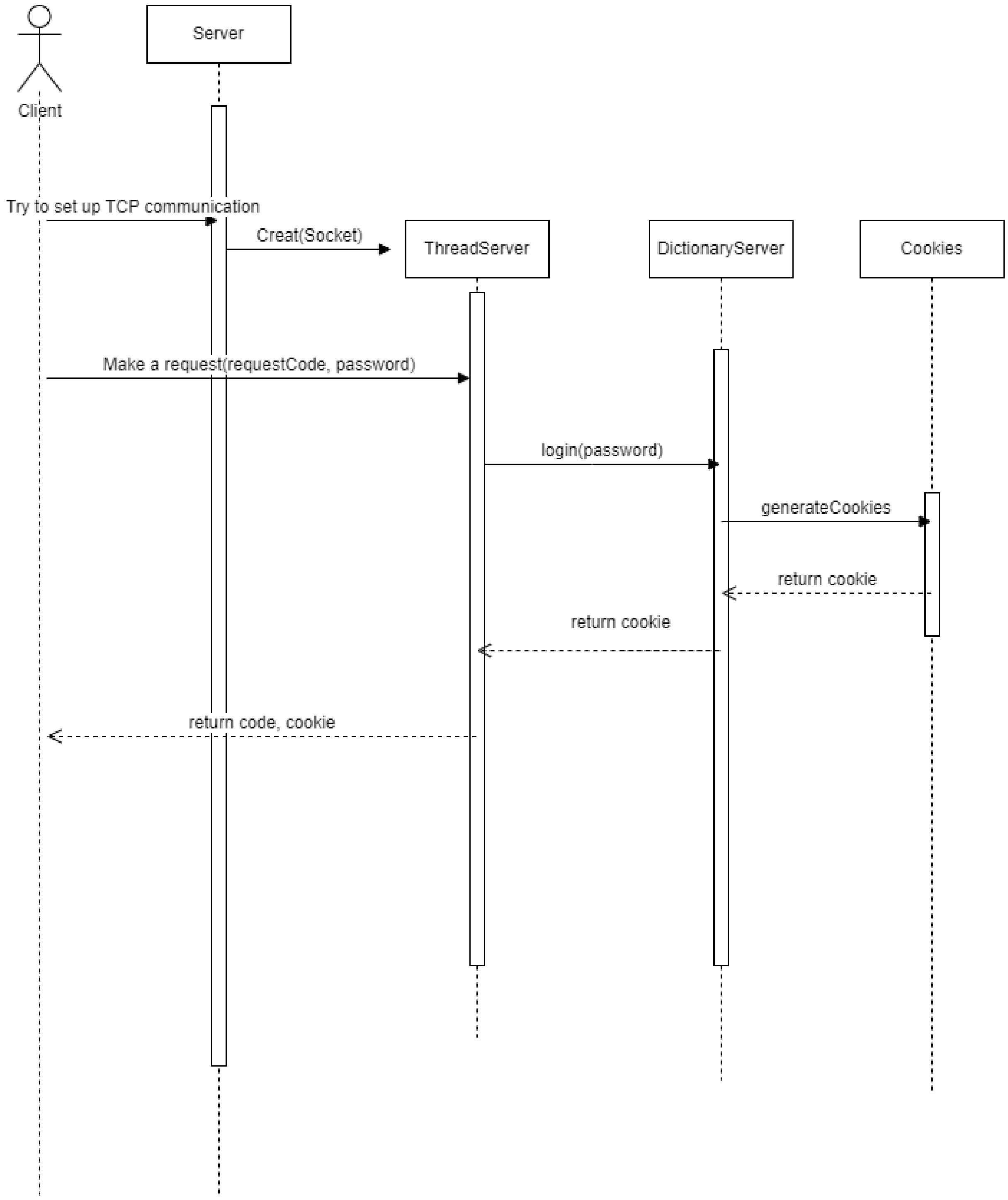

Figure 5. interaction diagram of login on the server side.
It is the success scenario that the server handles the login request from the client. DictionaryServer would check if the password is valid. The returned cookie is the encrypted time that the client logs in. An alternative scenario is that DictionaryServer finds that the password is not valid. Then it would reply to the client that the password is not valid.

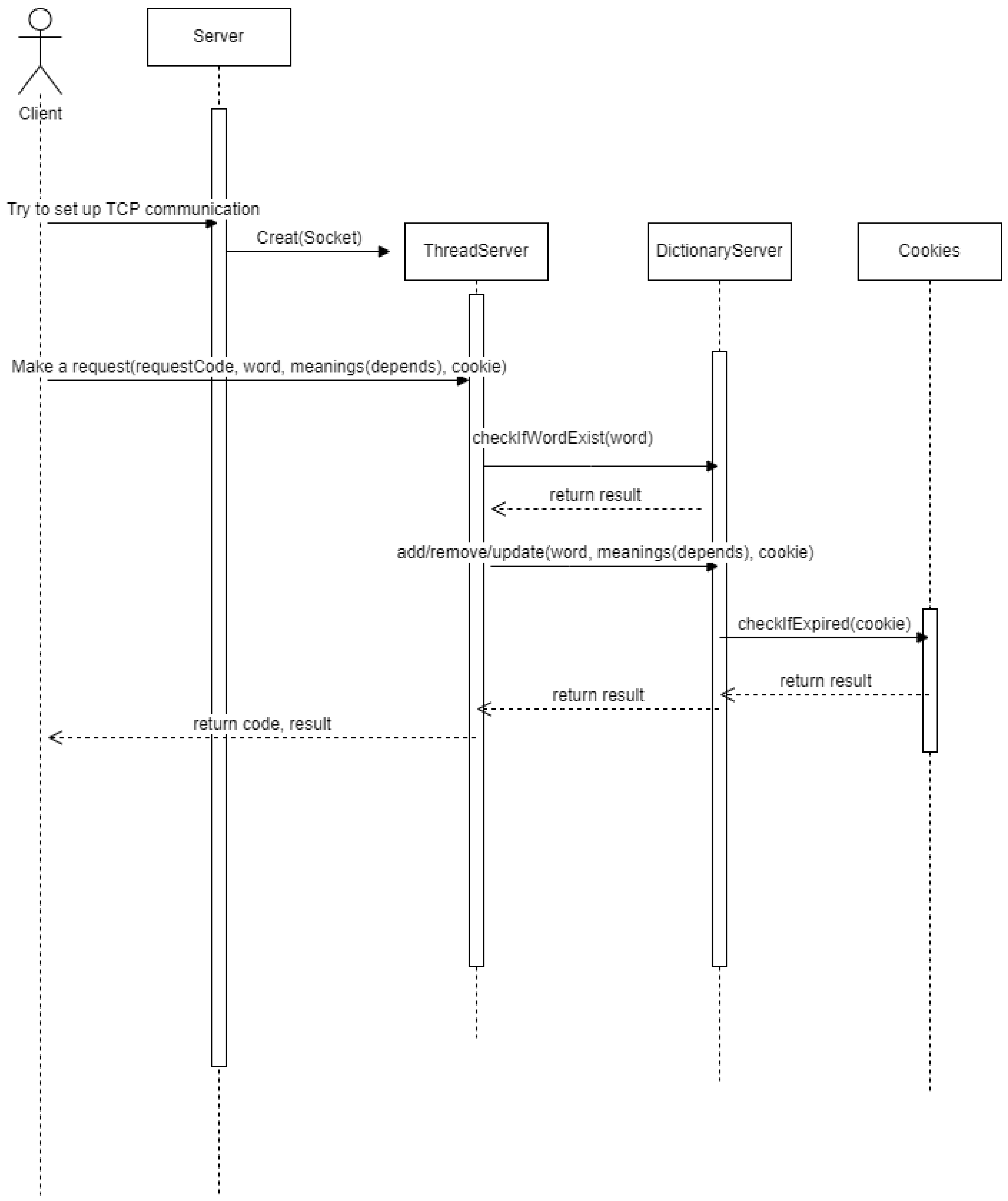

Figure 6. interaction diagram of add/update/remove on the server.
It is the success scenario that the server handles the add/remove/update request from the client. ThreadServer would check if there is the word (for add) / no word (for update or remove). DictionaryServer would check if the cookie is still valid.

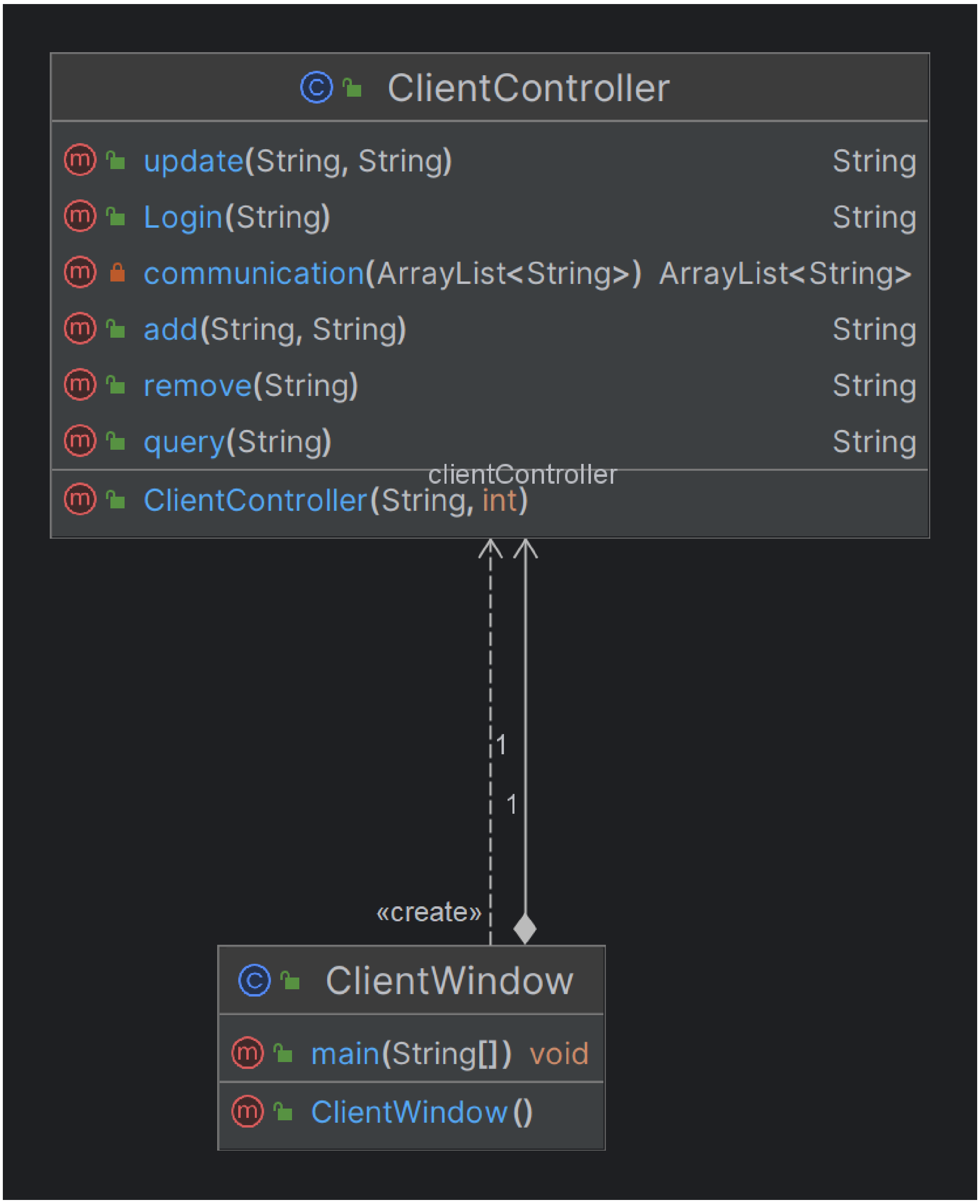

Figure 7. Class design of the client.

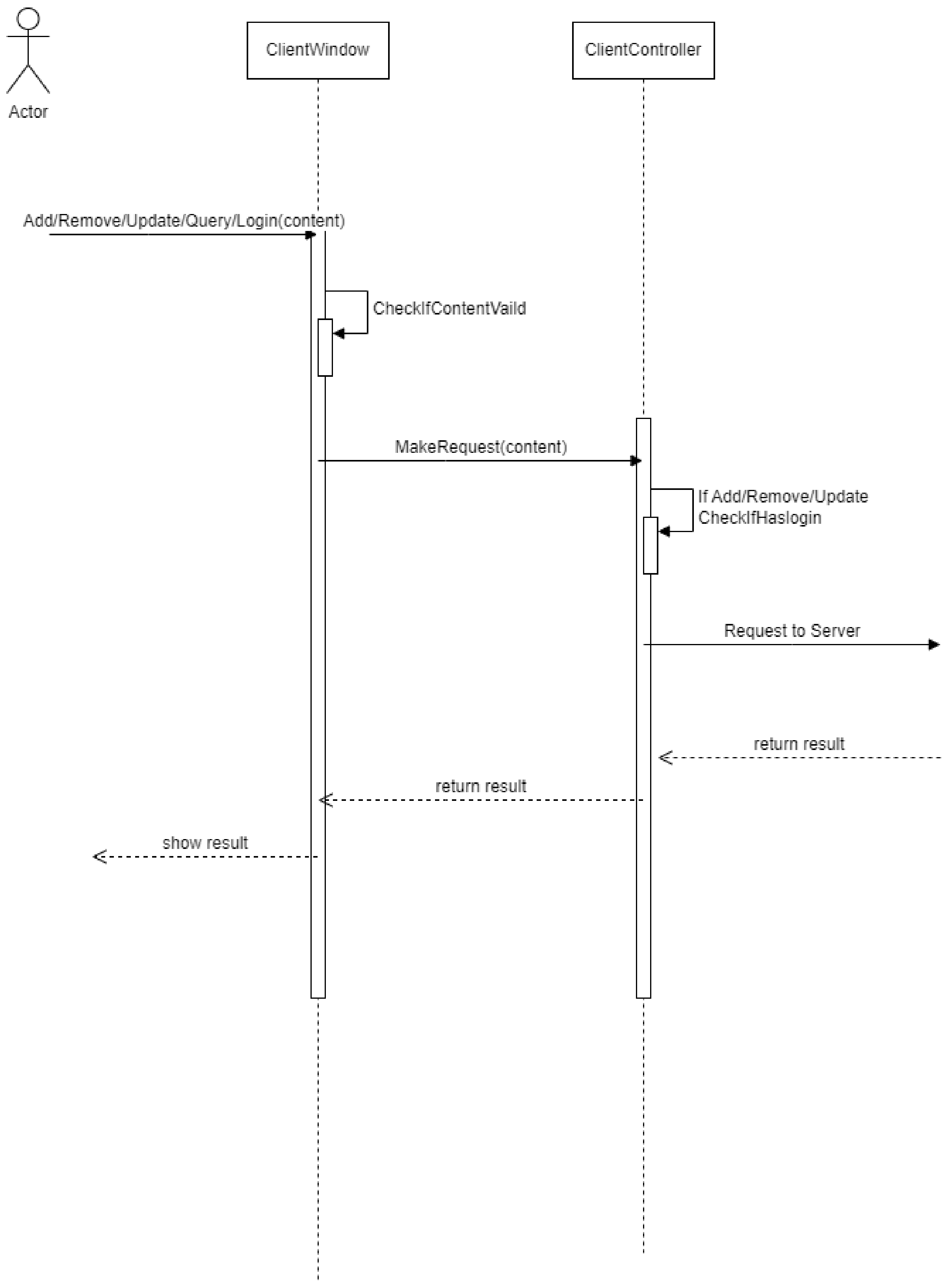

Figure 8. interaction diagram of client.
It is the success scenario that the client handles the request from the user.

A critical analysis of the work done followed by the conclusions.

TCP

The communication between client and server is over TCP. As the communication is required to be reliable and the protocol could provide sufficient reliability.

Thread per request

The server is adapting thread per request. The benefit is that it could reduce the complexity of the system. And the client could send multi requests and reduce the wait time. The disadvantage of such design is that it may reduce the max number of clients that the server could handle simultaneously as there is an upper limit of connections that the server could set.

Java Object Serialization

In this project, Java Object Serialization is used to store the dictionary. The benefit is that JAVA natively supports it. The disadvantages are the content of the file is not human readable and the saved file cannot be used after changing the template.

Synchronisation protection

The add/remove/update functionality of DictionaryServer is protected by synchronisation .

Excellence elements:

Content check:

The overall structure of the application could be modelled as front-end, back-end and remote server. In each part, there would be a checker that checks if there are mistakes made by the user and sends an error message to remind the user. On each layer, the check would check as many as possible depending on the information it could have. The front-end would check if all required text fields are filled with text. The back-end would check if the user has login to use features that require login (add/remove/update). The server would check if the login has expired or the request is allowed. The benefit of such a design is that it could reduce unnecessary calls and communication. For example, the server could check if the required text field is empty. But it would require extra communication between the client and the server. And it would reduce the complexity of the server. The disadvantage of such design is that as the server doesn't have the comprehensive check, users could send junk to the server if they develop their own client. Or even worse, the server may be destroyed by the malicious code sent by the user.

Error showing:

When using GUI, if there is an error, such as unable to set up a connection. There would be an error message shown on the GUI and a reminder message that suggests the user to contact the support team. The idea is close to the blue screen of death of windows.

Creativity elements

Login and Cookies:

All features that would change the dictionary require login. To further improve the security, cookie is introduced. The cookie used in this system, as mentioned, is a piece of encrypted string that indicates the time of login. It would be used to check if the user made the request too late after the login. If so, the user would be required to login again. The cookie is stored on the client side, so the server is not required to track all users. The string is encrypted, so some users with bad intentions are not able to extend the login by changing the cookie.
Note: there would be one password called “password” provided by the auto-generated dictionary file. The manage system of passwords is not provided as the system is similar to a dictionary server.