COMP90015: Distributed Systems –Assignment 1

Multi-threaded Dictionary Server

NAME: Boyu Li

STUDENT ID: 878890

E-MAIL ADDRESS: boyul@student.unimelb.edu.au

TUTOR: Xunyun Liu

1. Introduction

The Client–server model, referred to as the C/S structure, is a network architecture that separates the client from the server. Each instance of the client software can make a request to a server or application server. In this assignment, it requires a client-server system, which can achieve an application of a dictionary. The communication between clients and server is achieved by TCP socket. Moreover, multiple clients can access the server and perform operations concurrently. The server system applies multi-threaded architecture, and the threads are created per connection. (*COMP90015: Distributed System - Assignment1 Multi-threaded Dictionary Server*, 2018)

In this report, the problems are explained in detail, and the components of the system are briefly described. Then, an overall class design and the interaction diagram will be demonstrated. The critical analysis of the implementation and the conclusion of the assignment are illustrated finally.

2. Components of the system

The project is programmed by JAVA programming language. The operating system is Windows 10 professional 64-bit. Moreover, and the CPU of this system is Intel i7-8750H. The Memory of the running environment is 16GB.

3. Design of the System

Basically, the design of the project can be divided into two parts, including the design of server and client.

3.1. Class design

The project consists of six categories. Four of them belong to server design (See Figure 1) and the other two classes belong to client design (See Figure 2). The description starts from server design.

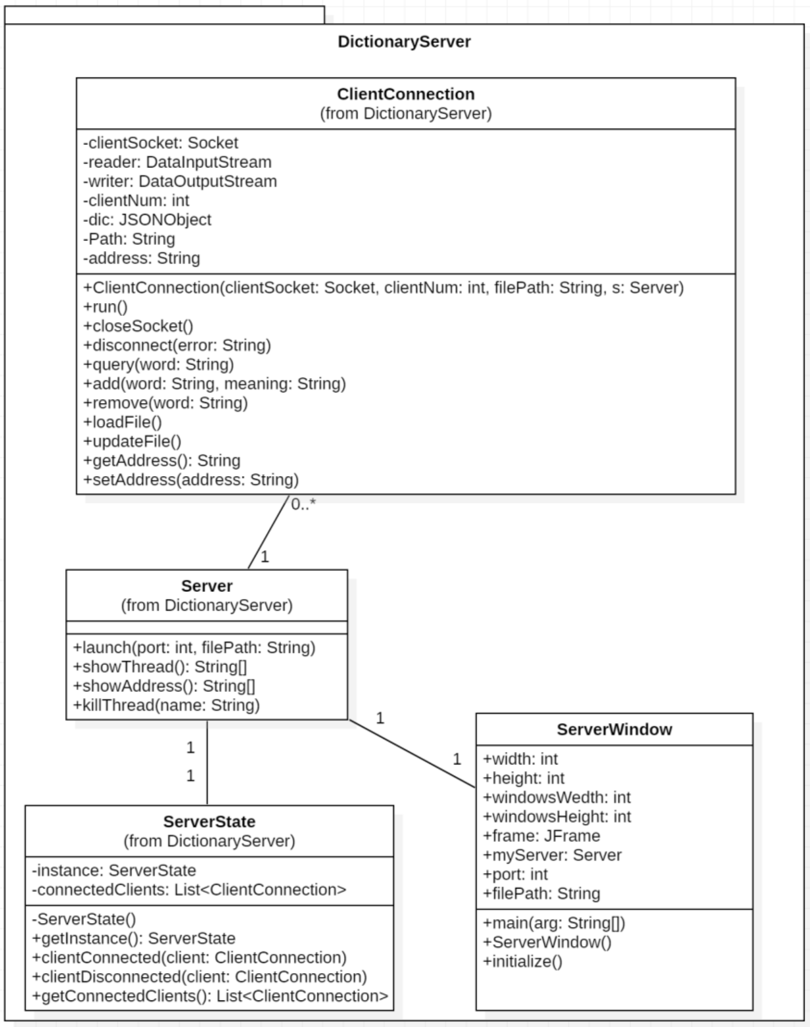


Figure 1: Class Diagram (DictionaryServer)

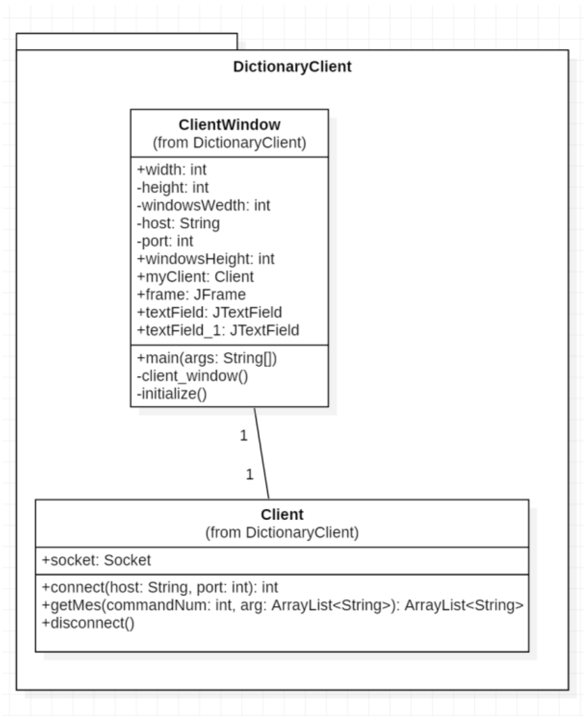


Figure 2: Class Diagram (DictionaryClient)

3.1.1. Server

Server class contains the main thread. As we know, this is a multi-Thread system. The server assigns one main thread to listen the assigned port. Then, it invokes sub-threads to process the operations of its corresponding client.

The error handling only covers the error of received socket.

3.1.2. ServerState

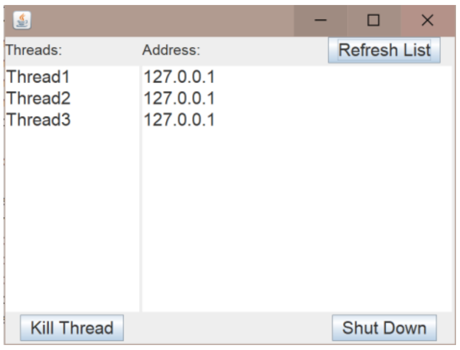
This class achieves some utilities to supervise the clients connected to the server. It maintains an ArrayList that records all the connected clients. Additionally, the operations of adding and removing client are defined in the class as well.

3.1.3. ClientConnection

This class account for more functions in class design. It aims to handle sub-threads to perform the operations of the clients, which include querying, adding, removing the words. The attributes of this class contains the information of each incoming client, the dictionary object and data stream objects. Therefore, the main operation to data are defined in this class. When the threads retrieve the commands and the processing object from stream, it will try to achieve these operations by processing the data in the cache or in the file.

3.1.4. ServerWindow

The class aims to achieve the GUI of server (See Figure 3). The entry of server system is defined in it. After initializing the window of server, the class launch the server automatically. There is a list tool showing the current threads and their IP address. The *Refresh* Button is used to refresh the list anytime. Two buttons at the bottom. One is used to kill the selected socket, and the other one is designed to shut down the server.



Threads:	Address:
Thread1	127.0.0.1
Thread2	127.0.0.1
Thread3	127.0.0.1

Figure 3: The GUI of Server

3.1.5. Client

Comparing to the server, the design of client is much simpler, which only contains two classes. The client class defined its communications with the server. The methods "connect" and "disconnect" are used to establish and close the connections with the server. Importantly, the method "getMsg" enable the clients exchange messages with the server. The messages can be distinguished by using different parameters, and the protocol of the communication is based on Json.

The error handlings cover transmission error, which is raised by sending or getting through data streams. Another error is data parsing error.

3.1.6. ClientWindow

The GUI of the client system is built here (See Figure 4). When the window is initialized, a connection can be established by clicking on a button. The main panel of the client includes a JtextField (to input word), a JtextArea (to show or input the meaning), another JtextField (to display the execution result), three labels (to indicate I/O) and five buttons (to perform basic operations).

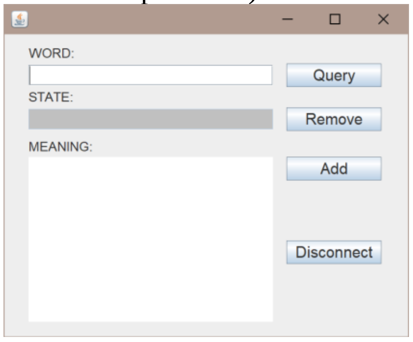


Figure 4: The GUI of Client

3.2. Interaction diagram

3.2.1. Sequence Diagram

The sequence diagram of the project shows below (see Figure 5). There are 5 actions shown in the diagram. Firstly, server managers launch the server by interface, and the connections with clients can be established by TCP socket. Then, connected clients can exchange messages with the server to proceed the operation such as querying words, add words and remove words. After this, clients are able to request disconnection with the server. Finally, the server can be shut down by the server managers.

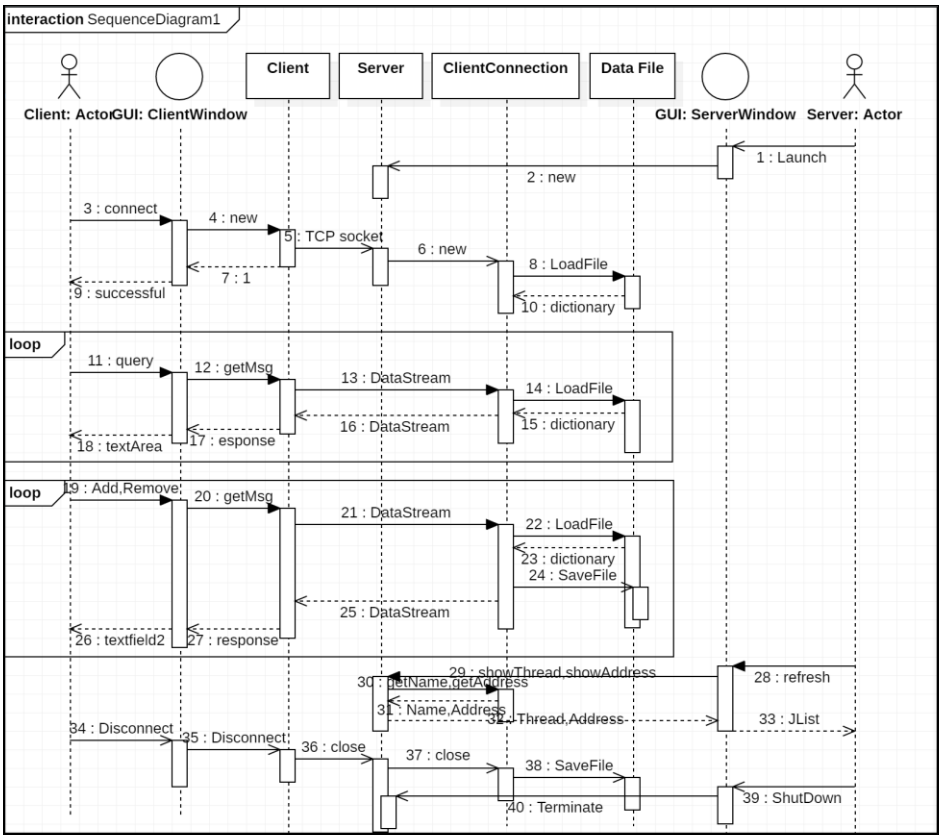


Figure 5: Sequence Diagram

3.2.2. Communication Diagram

(a) Server Operation

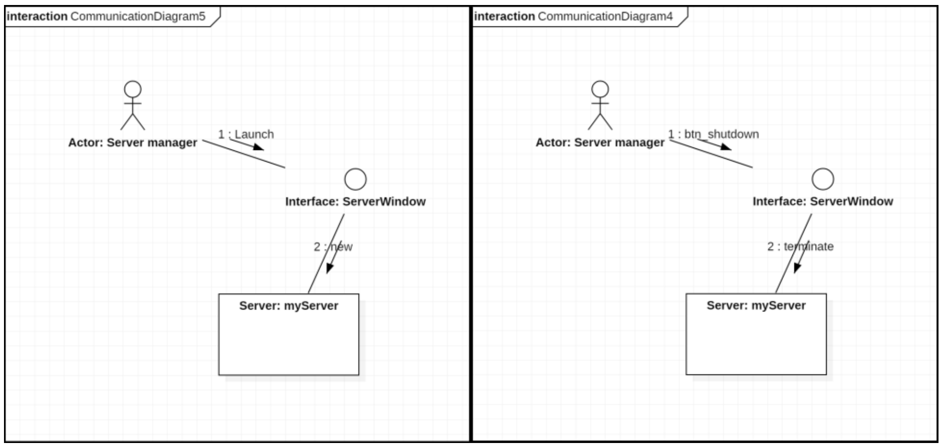


Figure 6: Server Launch and Shutdown

(b) Connection

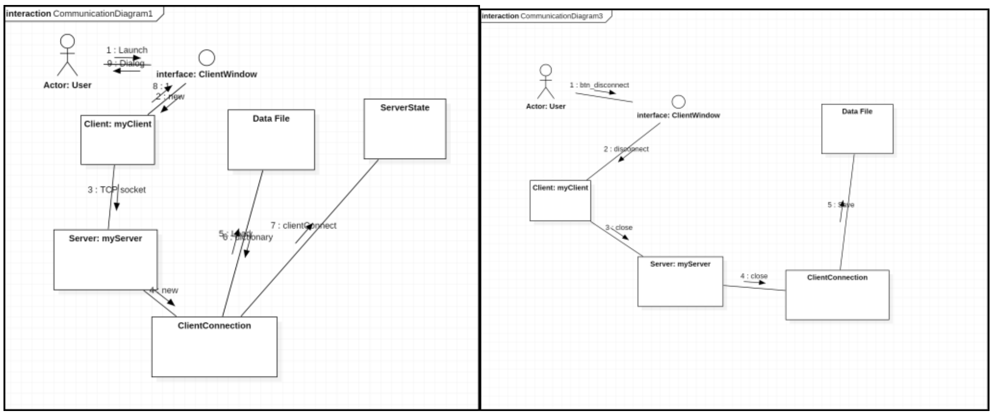


Figure 7: Connect and disconnect

(c) Basic operation (query, add and remove)
After establishing a reliable connection between clients and the server, they can exchange message through data streams. The sub-threads can process these basic operations directly.

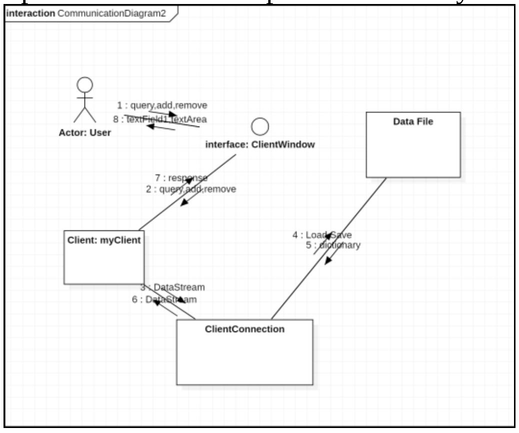


Figure 8: Basic operation (query, add and remove)

4. Critical analysis

Throughout the whole process of achieving the implementation of this C/S architectural system, thread-per-connection design is used to process multiple threads. The

4.1. Data format

After applying for various data structures, Json is considered as the most efficient one. As the protocol requires, the transmission should be formatted. Comparing to XML, Json has faster processing speed, and it is suitable to process small data. In the requirement, the data size is limited, so the Json format is more effective.

4.2. Error handling

The errors that may occur in the project are various. It is important to handle the possible errors and knowledge user the type of each error. The basic error type is data errors. Once a nonexistent word is intended to query or remove, or an existed word is intended to add, the errors occur on the server. Therefore, the server knowledge the clients which error type appearing. Input error appearing in the GUI of clients. Once the user query and remove without inputting the word, or add an item without inputting meaning, the request cannot be achieved by the server. So, this kind of errors is handled on clients' systems. Besides, the other error types include transmission errors, I/O errors and connection errors are handled in their corresponding classes.

4.3. Data consistency

Due to the design of multi-threads, data consistency is important to make the system reliable. In my design, the sub-thread needs to load the data file when a connection is assigned to it. However, during the first operation and possible one more operation, the data file may change due to operations from other concurrent sub-thread. To solve it, loading file is needed before each operation of each sub-thread.

5. Conclusion

In conclusion, the main design of this project includes TCP socket, multi-thread, Json data exchange format as the transport protocol and data storage structure, and GUI. After achieving this project, I get clearer concepts of the thread and socket and experience of how to apply them. Moreover, this assignment gives me a lot of inspiration on how to apply the GUI on java project.

Reference

The University of Melbourne. (2018). *COMP90015: Distributed System - Assignment1 Multi-threaded Dictionary Server* [PDF]. Melbourne.