THE UNIVERSITY OF MELBOURNE

School of Computing and Information Systems

COMP90015 Project Report

In Partial Fulfillment
of the Requirements for the Degree
Master of Engineering

Student Username: lingxuank
Student Name : Lingxuan Kong
Student ID : 957828

Contents

1 Introduction

The multi-thread systems play an important role in people daily life, it can provide enough security when concurrent clients query, have ability to deal with independent failure. In this project, we mainly focused on designing a multi-threaded server which allowed concurrent clients to do some basic operations, like adding, searching and removing word in the existed dictionary file. There are two main technologies that we have already covered in the lectures before, which will be used as lowest level of abstraction for network communication and concurrency in this project:

- **Socket**
- **Thread**

This report will mainly talk about the design procedures of this multi-threaded dictionary server, including architecture, class components, interaction with its diagram, and error handle model. In the end, this report will also provide significant critical analysis of the whole project.

2 Design Procedure

2.1 Architecture

The all communications happened via sockets in this project. In order to keep reliability of communication, I decided to use TCP communication Socket, which can be considered as a connection-oriented protocol. The TCP provides reliable, ordered, and error-checked delivery of a stream of Soctets (bytes) between applications running on hosts communicating via an IP network [1].
The connection model of program is **thread-per-connection:** whenever a new connection is created, a thread is created. For the server, one thread will be handling connections between clients and one thread will be handling operations such as searching words in the dictionary, etc.

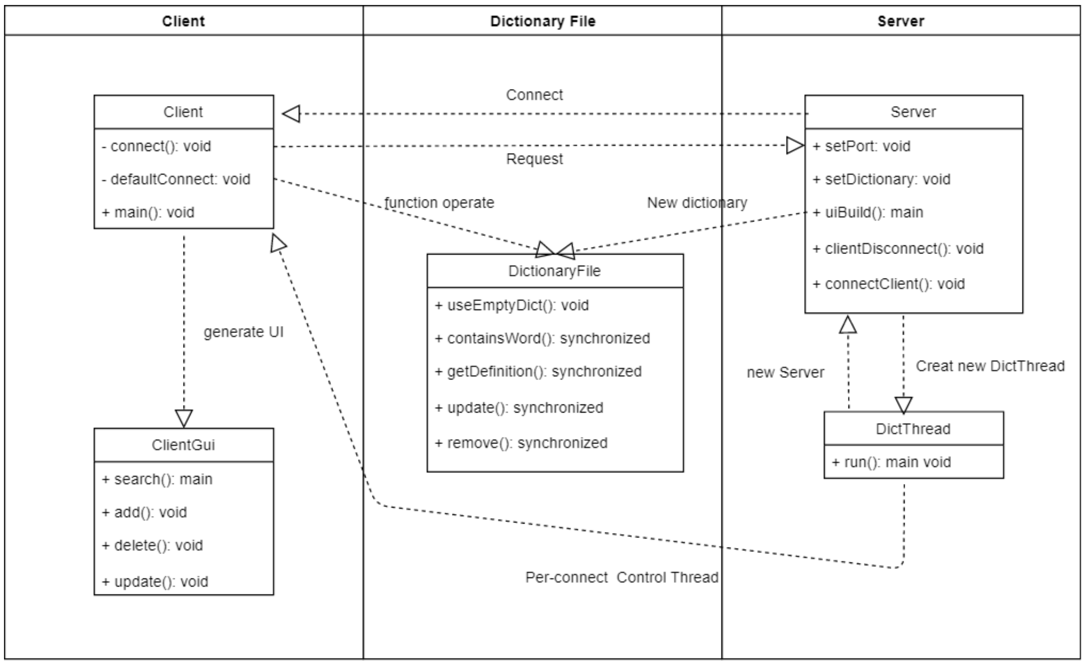


Figure 1: Class Components

Based on figure 1, we can observe that there are three different parts of this program:

- **Server:** Sever with its UI, and thread control. The server can be considered as the most important class in this program, control the client connection and disconnection, management of port and IP Address, handle the operations of clients to dictionary. The main function of **DictThread** is controlling thread, thread-per-connection.
- **Dictionary:** First detect the validity of dictionary of user input. If input and dictionary is valid, then this program will use this dictionary file. otherwise, it will create and use this one empty dictionary file.
- **Client:** There are 4 main functions of client, query, add, delete, and update. When these two class executed, a UI of client will be generated, user can enter the word and click the buttons on the UI to execute some functions.

2.2 Interaction

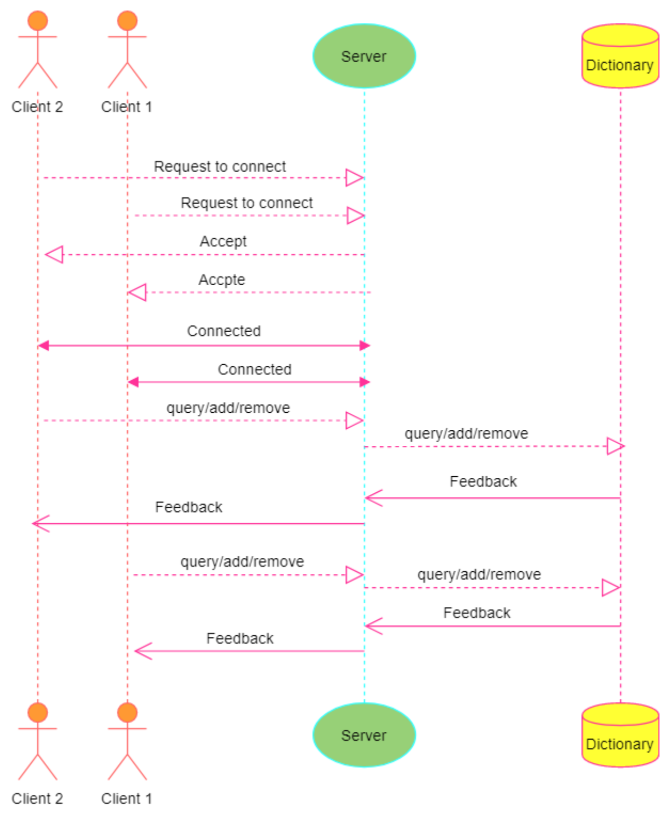


Figure 2: Interaction Diagram

After observation of figure 2, we can draw some significant conclusions of class interaction:

- Client send request to connect and waiting for accepting. After server accept connection request, there will be a new thread for client and server. Based on figure 2 we can find, there is a connection established between client and server.
- Client(Thread) do some basic operations: add, query, and delete. Those commands will send to server, and server will handle those operations to dictionary file.
- Dictionary file will send feedback(response) to server, then server send response to client(Thread). After that, user can see the results on the UI of client.

The data format of the dictionary file is .dat, which can be read and write by computer. Message exchange protocol that I used in this project is **HashMap** in java, on of significant role in Java Collection Framework [2]:

```java
private Map<String, String[]> dictionary;
dictionary = new HashMap<String, String[]>();
```

Based on the previous code, we can easily generate a new HashMap, which should be represented dictionary file.

2.3 Failure Model

TCP can guarantee connection reliability between server and clients, so there is no need for programmer to design a new failure model over TCP. However, some errors may still happen during the execution of this project. For example, input command is not invalid, network communication error, and file I/O exception. Thus, failure model design can be considered as an important part in this project. This report will provide significant solutions to some errors may happen in the program:

- **User input invalid:** Some new exception class will be written in the program. Then the program have ability to handle with these invalid inputs, and give some correct input tips to users.

```java
    /**Exception class to handle with no input*/
class NoInputWordException extends Exception
{
    public NoInputWordException()
    {
        super("Please␣enter␣the␣word!");
    }
}
```

- **Network communication error:** The program ask user to input valid **port** and **IP address.** However, if input port number is not in $(0, 65535)$, then this port number should be invalid. And the program will automatically use its default port number. For those invalid IP address, this program will also use local host automatically.
- **File I/O:** All file IO should be executed in try, catch block with **IOException.**
- **No dictionary file:** If there is no local dictionary file, dictionary path input is invalid, or file nor found in that path when program begin to run, then this program will create a new empty file named: ”dictionary.dat”.

2.4 Program Function

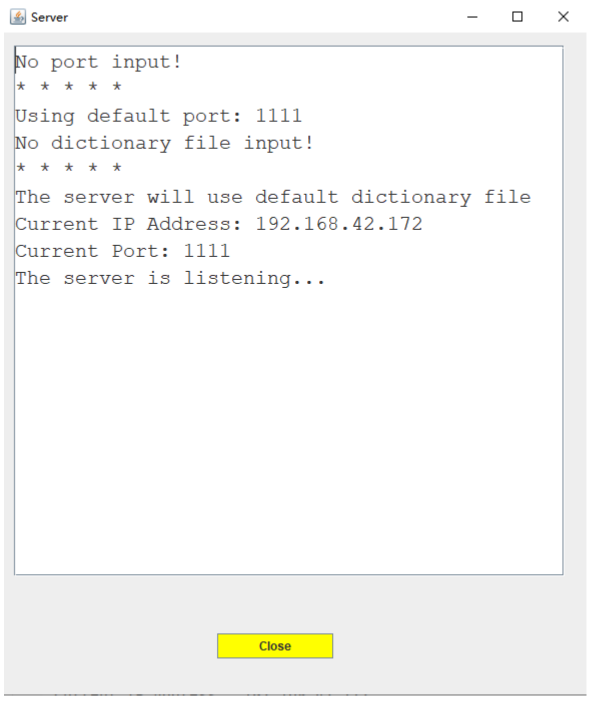
This program should allow clients do some basic operations to dictionary: Query the meaning of word, add new word with its meaning, and remove one word with its meaning.

- **Query:** Client can query the definition of the word, then the server will give feedback, including the meaning of this word. Or, if the input word is not in the dictionary, the client will receive ”The word is not in the dictionary!”. Additional, if user did not input any word, and click query button, then this program will give ”Please input word” on the user-interface of client.
- **Add:** User can add new word and its meaning to the dictionary. Before this new word added into dictionary, the program will check if it is already in the dictionary, and give feedback to user: ”The word already in dictionary” or ”You have added word to dictionary”. However, the input of word meaning should no longer than 10 rows. If user input has more than 10 rows, this program will show an exception ”InputTooLong”, and ask user to input again.
- **Update:** User can update the meaning of an existed word. User should first input the word that he want to update, and enter the new meaning of this word. After that, just click the update button on the UI. Then the program will check the word validity and of it is in the dictionary.
- **Delete:** This program allows user to delete word and its meaning in the dictionary. Before execute this operation, the program will check that if the word that user want to delete in the dictionary.

In addition, this program named Multi-threaded Dictionary Server, which allows multi-threaded. In another word, the program allows concurrent with multi-client.

2.5 User-Interface

This program has two different user-interfaces, used **Java Swing** to build:



```
No port input!
* * * * *
Using default port: 1111
No dictionary file input!
* * * * *
The server will use default dictionary file
Current IP Address: 192.168.42.172
Current Port: 1111
The server is listening...
```

Figure 3: User-Interface of Server

Figure 1 shows the UI of server. Based on this photo, we can obtain some significant information and parameters: **Port** and **IP Address**. If user do not input, or input port is invalid, this program will use default port and local IP Address.

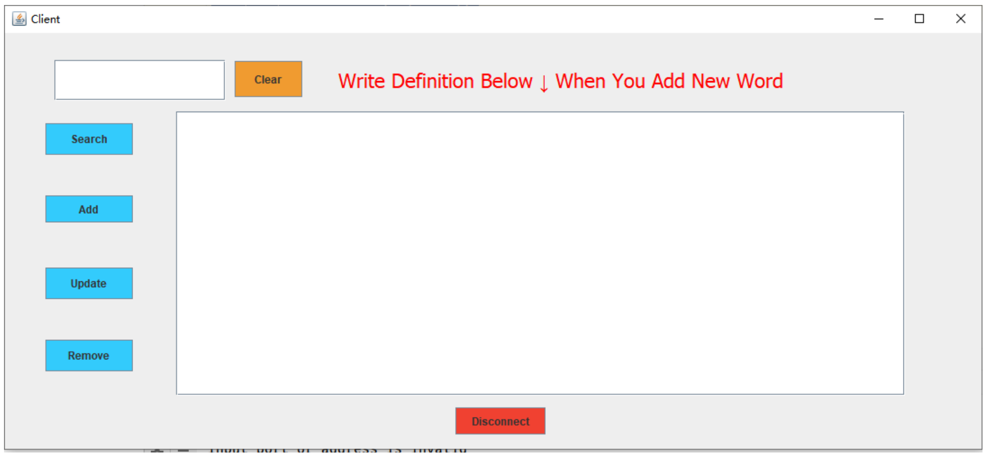


Figure 4: User-Interface of Client

The UI of client shown in figure 2. According to this figure, we can find that there are 4 different buttons, search, add, update, and delete, which represents different functions in this program. The small text field allows user input the word, and the big text field will provide feedback, including meaning of word, and some error outputs.

3 Conclusion and Critical Analysis

To conclude, this program mainly focus on design a Multi-threaded Dictionary Server, which allows user to do some basic operations, search, add, update and delete word to a dictionary. The program use **thread-per-connection** and **TCP/IP** connection to keep communication reliability. This program also has error handle mechanism to deal with some errors or exception. The user-interface designed by Java Swing, user can just click the buttons on UI to execute operations.
However, there are still some limitations on this project, which can be improved in project 2 :

- **The protocol of message exchange can be improved:** When user input the meaning of word, the input should less than 10 rows, or the program will give an error. Because if there are too many rows in the dictionary file, the query will become very slow. However, if we design another high-efficient file transform protocol, the input will not be limited.
- **The scalibility of thread-per-connection:** The scalibility of thread-per-request is better than thread-per-connection. Thus, I can try to design thread-per-request in other multi-thread model.
- **User-Interface can be improved:** The color scheme of the interface is too monotonous. And in the client UI, there is no enough input tips to users. The user may do not know the way of where to input the searching word, and how to input the meaning of the word.

References

[1] Wikipedia. (2019, aug) Transmission control protocol. [Online]. Available: https://en.wikipedia.org/wiki/Transmission_Control_Protocol

[2] oracle. (2019, mar) Hashmap. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html