COMP90015: Distributed Systems

Assignment 1 Multi-threaded Dictionary Server

Name: Jinzhe Shan

Student ID: 1069912

Email: jinzhes@student.unimelb.edu.au

Tutor: Haowen Tang

Content

I. Project Introduction

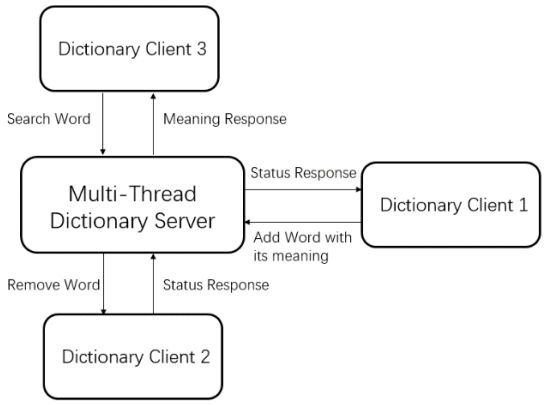
This is a Multi-thread Dictionary Server project related on Assignment 1 of COMP90015 Distributed Systems on the University of Melbourne. In this project, I implemented a multi-thread dictionary server that can response queries (including :search the meanings of word, add a new word with its meaning and remove an existing word) from many clients simultaneously based on the client-server architecture and multi-thread concept learning from the subject.

II. System Overview

In this part, I will introduce the overview of this project, which includes the server/client model and thread model.

A. Server/Client model

There are many types of System Architectures such as Server/Client model, Peer-to-Peer model. I chose Server/Client model to implement this multi-thread dictionary project. It is not only because of the name of this assignment "Multi-thread Dictionary Server" but also because that dictionary server is required with higher performance and availability than clients to make sure meet a number of queries from clients at the same time.



B. Thread model

There are four architectures of multithread servers: Worker pool, Thread-per-request, Thread-per-connection and Thread-per-object. In this project, I compared Thread-per-request with Thread-per-connection and chose Thread-per-request model eventually, since the scalability of Thread-per-request can better than that of thread-per-connection. Imaging the situation that we use Thread-per-connection, some clients who have connected with server but send two continuous queries from a long time, which really wastes the resource of dictionary server. Therefore, I select Thread-per-request model, which creates a thread when client send a query and kills the thread after response to client.

III. Implementation Details

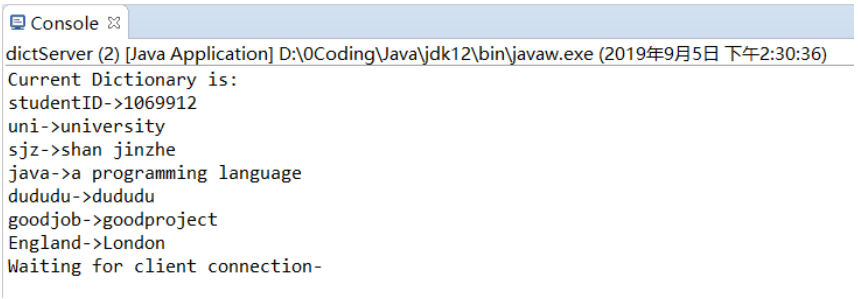
The specific implementation of this project will be introduced as follows. There are three java files: dictServer.java, dictClientFrame and dictServerFrame. TCP connection is used in the project to provide reliability.

A. Components: Server, Client and Dictionary

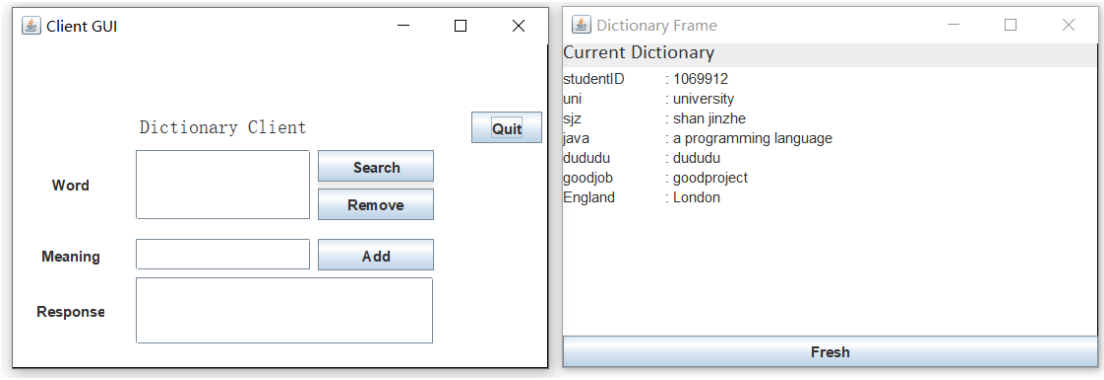
This project has three main components: server, client and dictionary:

1. In the part of server, the main program (dictServer.java) provides multi-thread dictionary services for many clients in the same time.
2. When it comes to the part of client (dicClientFrame.java), dictionary clients can input word they want to search or remove and add new word with its meaning easily by clicking mouse.
3. Considering user experience of this dictionary project, I implement a dictionary module (dictFrame.java), which is a dictionary GUI providing a clear view of current dictionary and users can search, remove.

Firstly, server program will be set up. It shows the current dictionary and waits for connection from client.

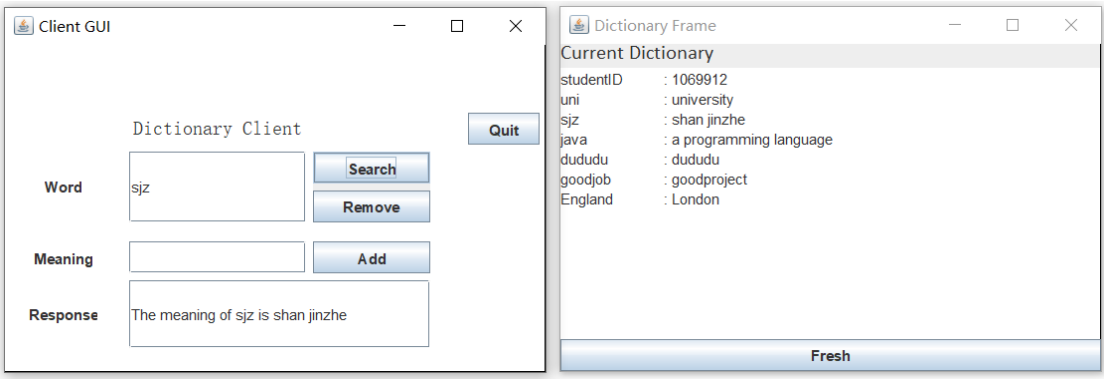


Then, Clients launch the dictClient.java and then both client GUI and dictionary GUI appear in front of clients. Clients can search, remove and add word easily.



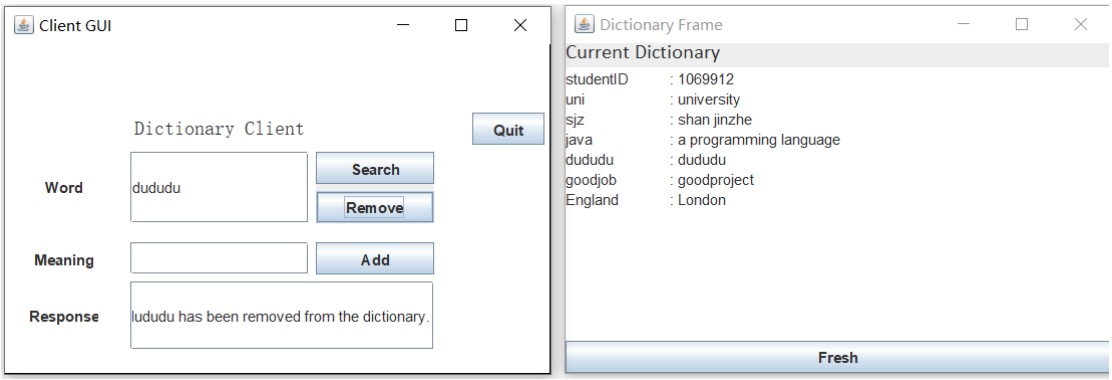
B. Functional implementation

1. Search word

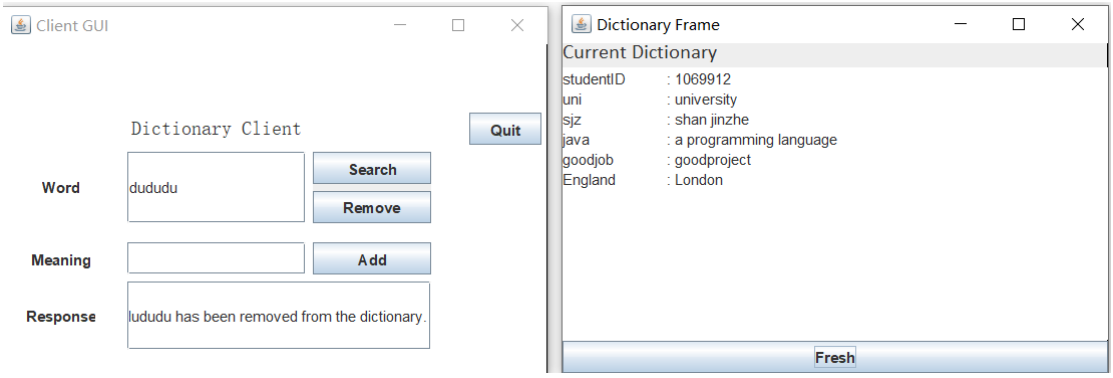


2. Remove word

for example, a client wants to remove the word of “dududu”. This client type the word of “dududu” In the text field of word and then click the button of “Remove”

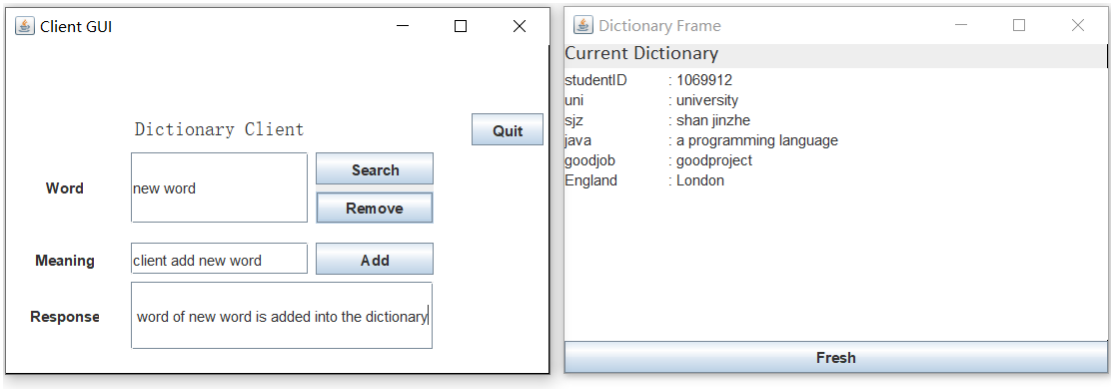


After clicking, the client will receive the response from dictionary server “The word of dududu has been removed from the dictionary.” Then the client need to click the button of “Fresh” in the right Dictionary GUI and current dictionary has been changed.

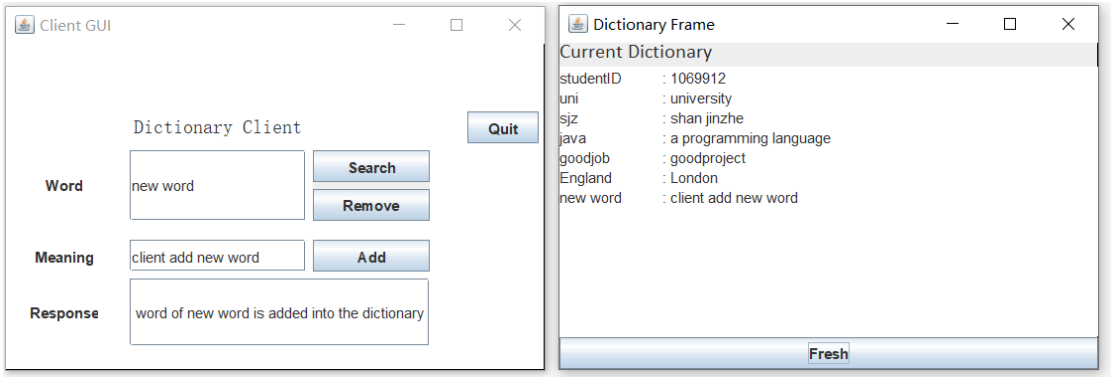


3. Add word with its meaning

If client B want to add a new word “new word”. B need to type the word in the word text filed and the meaning of new word in the “meaning” text filed and then click the button of “Add”. After clicking, he will receive the response from dictionary server “The word of new word is added into the dictionary”.



When B click the button of “Fresh”, B will see the new word in current dictionary.



C. Thread Security

When it comes to Parallel, the security of shared resources is the biggest challenge. In this project, dictionary is shared resources for multiple clients. In order to guarantee the security of thread, I used “synchronized” key word to implement the method of “addorremoveDict()” in “dictServer.java”, which means only one client thread can add or remove word from dictionary simultaneously. The method of searchDict() is used to search the meaning of word. Many clients are allowed to search the meaning of word at the same time because it does not change the dictionary.

```
private static synchronized void addorRemoveDict(String msg, Socket client) {
    BufferedWriter out = null;
    try {
        out = new BufferedWriter(new OutputStreamWriter(client.getOutputStream(), "UTF-8"));
        String[] msgList = msg.split("-");
        String type = msgList[0];
        String word = msgList[1];
        if(type.equals("add")) {
            if (msgList.length !=3) {
                System.out.println("Error: please input the word you want to add with its meaning");
                out.write("Error: please input the word you want to add with its meaning");
                out.newLine();
                out.flush();
            }
            else if (dictionary.get(word) == null) {
```

IV.Excellence and Creativity

A. Excellence

My implementation of this project is far more than the minimum requirements from assignment handbook. I have done some excellent work in this project as follows.

1. Exception Handling

   There are many exceptions, especially socket exceptions when I implement this project. I did exception handling in many aspects.

   Because socket communication is based on IP address and port number, when the port number is occupied the socket connection will throw a bind exception.

```
} catch (BindException e) {
    System.out.println("The port is using now,please exit it and try again."+"\n");
    return;
```

   In addition, when clients input nothing but click the button of “search” or “remove” or add a word but forget input its meaning, it may cause NullPointerException, so I add a analyzing condition about this situation.

```
private static void searchDict(String word,Socket client) {
    try {
        BufferedWriter out = null;
        out = new BufferedWriter(new OutputStreamWriter(client.getOutputStream(), "UTF-8"));
        String meaning = dictionary.get(word);
        if(word!=null){
        if(meaning!=null) {
            System.out.println("the word is in the dectionary.");
            out.write("The meaning of " + word + " is "+ meaning);
        }
        else {
            System.out.println("the word is not in the dectionary.");
            out.write("The word of " + word + " is not in dictionary");
        }
        out.newLine();
        out.flush();
    }
    } catch (NullPointerException e) {
        System.out.println("Please input the word you want to search");
    }
```

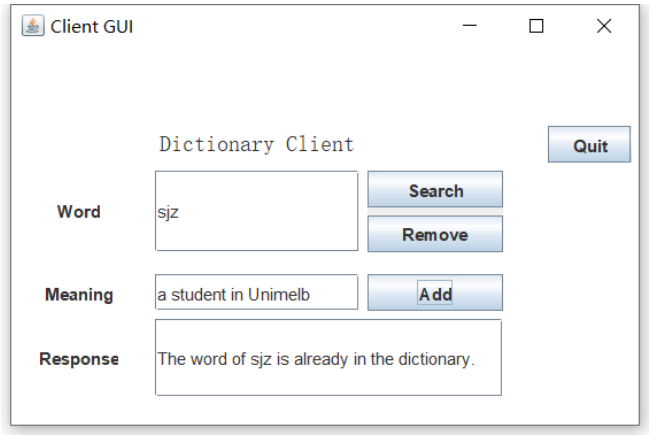
2. Input fault-tolerant

   As the reason of multifarious input from clients, I designed many input fault-tolerant to address almost any cases.

   For instance, when client want to add a new word but forget input its meaning, the response from server will remind the client to add new word with its meaning.

When client search or remove a word which is not in dictionary, the response from server will remind the client to try another word.

In addition, when clients add a new word but it has existed in the dictionary, the response from server will remind the client.



3. Optimization GUI exit

The default method when GUI exit in swing is just close the GUI window, it cause a connection exception because the socket is already disconnected but the input is may still transport data. Therefore, I override the method of GUI exit to make sure that all sockets and input/output streams are closed and send message of client exit to server.

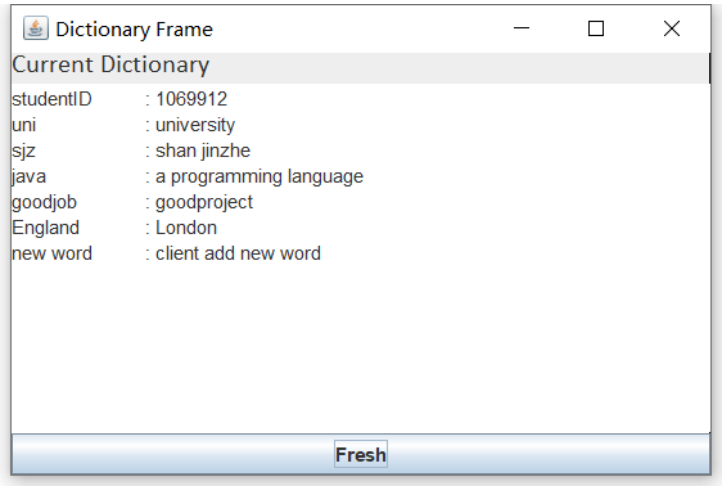
```
// close the window after close connection
class MyWindowListener extends WindowAdapter{
    @Override
    public void windowClosing(WindowEvent e) {
        super.windowClosing(e);
        exitClient();
    }

private void exitClient() {
    try {
        socket = new Socket(ip, port);
        output = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream(), "UTF-8"));
        output.write("Client is exiting...");
        output.newLine();
        output.flush();

        output.close();
        socket.close();
        System.exit(0);
    } catch (IOException e1) {
        e1.printStackTrace();
    }
}
```

B. Creativity

As the consideration of user-friendly software, I do implement a dictionary GUI providing a clear view of current dictionary when users search or remove words. This also can fresh when dictionary changes.



What is more, the operation of search, remove and add is implemented by HashMap, which is not only realize response quickly with the growth of dictionary volume but also avoid repeated words.

V. Conclusion

Ingenious functional implementation is the advantage of this project. In addition, there are some points can be improved in the future works, such as more beautiful graphical interfaces and scalability of huge dictionary volume or a number of clients' access.