COMP90015 Distributed Systems – Assignment1 Report

Multi-threaded Dictionary Server

Name: Xudong Ma studentID: 822009

Email: xudongm@student.unimelb.edu.au

Tutor: Xunyun Liu

1. Resource

The dictionary using in this project is a json file cited from GitHub [1], but I made some variances on the raw resource so that it can fit the demand here more specifically. Finally, the file contains a JSONArray which stores many JSONObject. Every object is a word recording. Each JSONObject contains two components, a key named “word”, and another named “meanings”. The key “word” corresponds to a String which is the spelling of a word. While the key “meanings” corresponds to a JSONArray which contains a set of JSONObject. Every JSONObject here only has one key named “meaning” and it corresponds to a meaning of the word. Specially, the reason for using JSONArray rather than a String array to store the meanings of words is to increase the openness of the dictionary. For example, if we want to add an attribute of word class for every meaning of the words. JSONArray can easily achieve this by adding a new key “class” for every object inside the JSONArray. While, String array would make tasks like this much harder.

Apart from the dictionary file, this project also uses a json file handing package “json-simple-1.1.1.jar” and the “WindowBuilder” module of java.

2. Overall Design

The multi-threaded server in this project is implemented using thread-per-connection architecture and all the communication takes place via TCP sockets. Therefore, all communication is reliable. All the messages passing from clients to the server are formatted inside a JSONObject which contains two or three components. They are two string attributes named “command” and “word” respectively if the user wants to search or remove words. While, if the user wants to add words, there will be another JSONArray attribute named “meanings” which contains all the meanings of the word. Similarly, all the messages passing from server to the client are also JSONString. Besides, in order to speed up the communication, all the input and output stream is packaged using “BufferedWriter” and “BufferedReader”.

2.1 Server

The Class diagram of the server is shown in Figure 1. There are three classes here. DictionaryServer is the class where the main method inside. All the attributes and methods in this class are static in order to share the dynamic data among all the clients. Most of the attributes are easy to understand according to their name except “searchWords” and “ifChanged”. Firstly, “searchWords” is used to store all the word only with their spelling (without their meanings). The items inside “searchWords” correnspond to the items stored inside the JSONArray “words” one by one. This design is used to simplify the word search operation. It is clear that JSONArray is more difficult to traverse and search based on the spelling of a word. Therefore, if we update “searchWords” and “words” synchronously, we can drop by the item in “words” using

the index of item in “searchWords” and this will save a lot of time. As for “ifChanged”, it will be described later.

DictionaryServer involves a private method “startTheServer” which is used to load the dictionary from the hard disk. Besides it offers six public methods for all the users to use. Clearly, “addWord”, “search” and “removeWord” are offered as the interface for the client to add, search or remove words. Among them, the return value of “removeWord” and “addWord” are boolean because they only need to inform the user if their operations are successful. While the return value of “search” is a JSONObject because it needs to return the meanings of a word if it exists in the dictionary or the message to inform the user the word they want to search is not in the dictionary. More importantly, “addWord” and “removeWord” are synchronized as we do not want to see two users add the same word using different meanings at the same time and this is illegal. The attribute “ifChanged” is used here to mark if the data in the memory has been changed. If someone has changed the word collection, like remove or add a word, the method “updateDatabase” will be run to store the data in memory to the hard disk. For a similar reason with “addWord” and “removeWord”, “updateDatabase” is also defined to be synchronized.

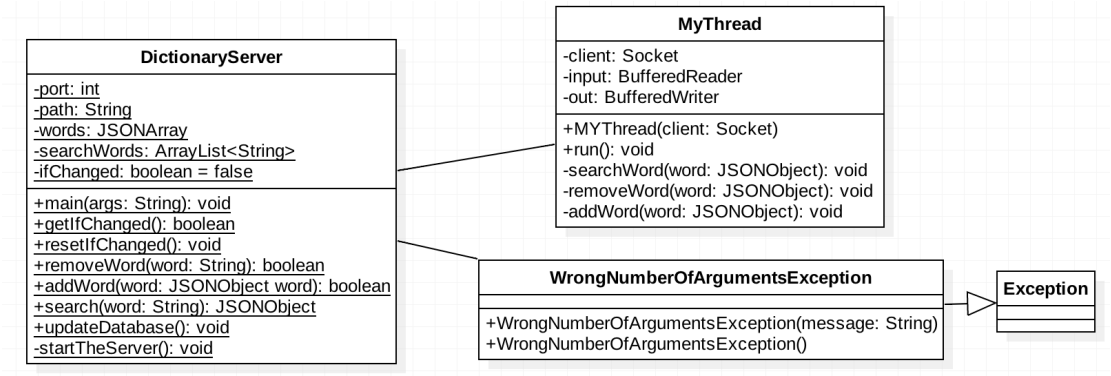

Figure 1. The class diagram of Server

As for the class “MyThread”, when a client is connected to the server successfully, an instance of it will be created to keep the communication between this client and the server. Once the client exits, this thread be stopped.

“WrongNumberOfArgumentsException” is used to handle the situation if you input wrong numbers of arguments from the command line when starting the server.

2.2 Client

The structure of the client side is easy to understand. As shown in Figure 2, there are three classes here. The main method inside the class “DictionaryClient” gets a connection with the server and then using the client instance as the argument to create an instance of “ClientGUI”. Then the GUI will finish all the communication between the current client and the server. GUI will be discussed in a particular part below in this report. Similarly, “WrongNumberOfArgumentsException” is used to handle wrong numbers of arguments problems in the client side.

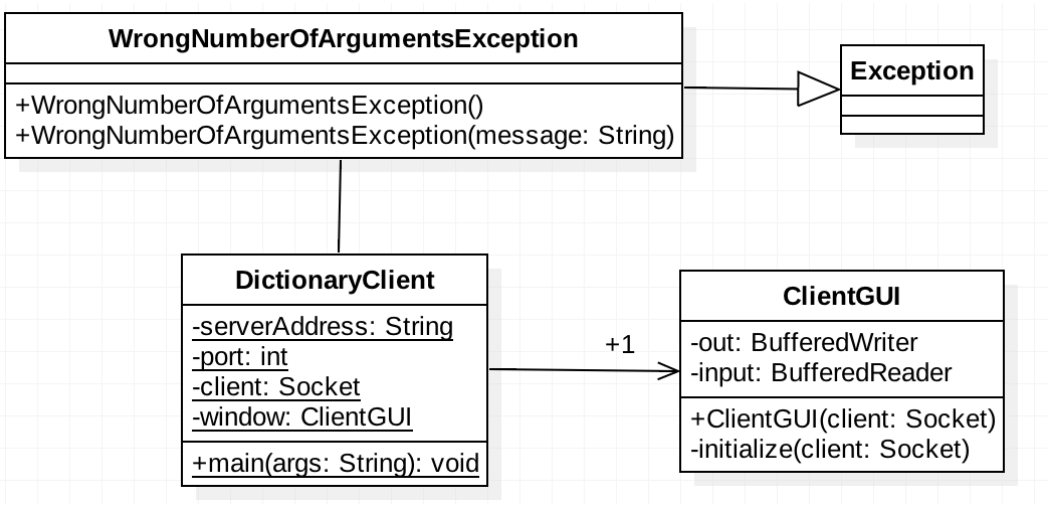

Figure 2: The class diagram of Client

3. GUI

The GUI for client is mainly implemented using “WindowBuilder” of java. This design offeres three buttons “Search”, “Add Word” and “Remove Word”. Additionally, a single line text field is used for the user to input word they want to search, remove or add. Then a text area with scrollbar is used to show warnings or the results of the operations. If a word has too many or too long meanings, the user can use scrollbar to watch all of them. As shown in Figure 3 below:

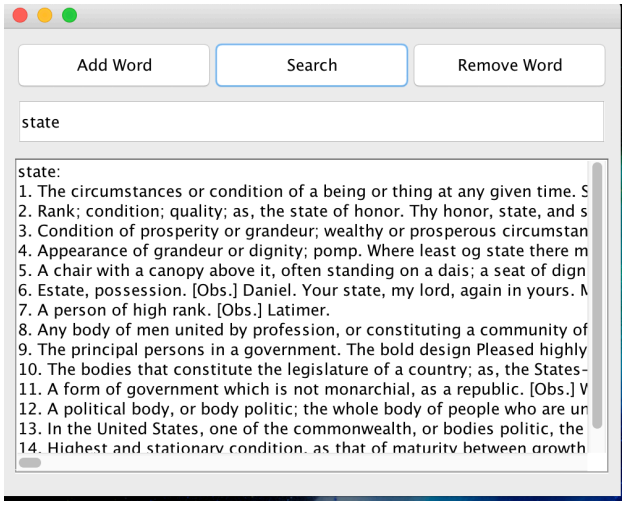

Figure 3: GUI demonstration

Furthermore, some errors in the server or in the client side will result in a window being popped up to show the error message. For example, if the server is shut down during the communication, while the user still wants to use the service, a window would be popped up to inform the user “The server is offline” and then exiting the client process. As shown in Figure 4.

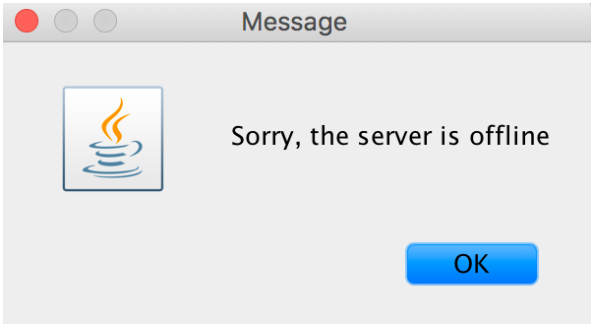

Figure 4: Error message demonstration

4. Failure Model

4.1 For both server and client

If wrong numbers of command line arguments are inputted when starting the system. A "WrongNumberOfArgumentsException" will be thrown and caught so that a window with the informed message is popped up and then exiting the system. For the server, the message will be "Sorry, please input two arguments: port filePath". While, for the client, the message will show "Sorry, please input two arguments: serverIP port".

4.2 For server

If the port number inputted is illegal, such as if a string is inputted, a "NumberFormatException" will be thrown and caught. Then a window will be popped up to inform you "Sorry, please input an int number as the port number" and then exiting the system.

If the server cannot find the dictionary following the path you provided, a "FileNotFoundException" will be thrown and caught then popping up a window showing the warning message "Sorry, cannot find the file following the path you provided". After that, exiting the system.

If the dictionary file provided not fit the format we supporting, a "ParseException" will be thrown and caught so that a warning window is popped up and informs you that "Sorry, the file you provided doesn't fit the format needed" and then exiting the system.

If the port you provided is unavailable, an "IOException" will be thrown and caught then a window will be popped up and show that "Sorry, the port is unavailable" and then exiting the system.

If a client exits abnormally, a "NullPointerException" will be thrown and caught, then the server will print a message "Sorry, the client is offline." While, if the client exits normally, the server will print a message "Client offline".

All other operations from the client which result in "IOExceptions" of the thread will make the server print a message "Sorry, something wrong happened inside the io stream!!!".

4.3 For Client

If the server's address you provided is not a legal address or it is unmeaningful, an "UnknownHostException" will be thrown and caught. Then a window will be popped up and show that "Sorry, can't find the host" and then exiting the client system.

If the server is not available, like it is offline or the network is shut down, a window will be popped up and show that "Sorry, fail to access the server, please check the network connection or if the server is online." Then exiting the client system.

If the user wants to add word without any meaning, the user will receive a warning message "Sorry, you can't add word without any meaning". If the user inputs the word they want to add in a wrong format, he will receive a warning message "Sorry, please separate the word and different meanings using "" ".

If the system is developed and the client can have more kinds of requests. It will be possible that the client's request is unmeaningful so that the JSONParser in the server side cannot parse its request, then the client will get a warning message, "Sorry your request is unmeaningful". (for openness)

If the server is shut down accidentally during communication, a "SocketException" will be thrown and caught, then the client side will pop up a window showing that "Sorry, the server is offline". After that, the client program will be exited.

For the clients who want to search the meanings of a word, if the server provided something unmeaningful, the client will print a message "Sorry, the server offered an unmeaningful message." in the text area.

5. How the system works normally

5.1 Query the meanings of a given word

The user needs to input a word they want to search inside the text field, then click the button "Search". If it is inside the dictionary, the client will receive a JSONString, then the client parses the JSONString to show the meanings of the word in the text area. While, if the word does not exist in the dictionary, the client will also receive a JSONString. After parsing it the client could get the message "The word is not in the dictionary" inside that object and show it inside the text area.

5.2 Add a new word

The user should input the word and its meanings they want to add in the text field following the format of "word" "meaning1" "meaning2" ... If the word is not in the dictionary, the user will receive a message "Sorry, the word is not in the dictionary". If the word is added successfully, the user will receive a message "Word has been added to the dictionary". All the word addition operations are permanent. Once happening, the result will be stored both in the memory and in the hard disk of the server. In order to give the user a more fluent experience. The disk writing operations are hidden, which means that once the addition operation succeeds in the server's memory, the user will

receive a result message, then the server will do the file writing operation on the background. The word addition operations are synchronized so that different users cannot add words at the same time, because they may add a same word but giving different meanings at the same time.

5.3 Remove an existing word

If the user wants to remove a word which does not exist in the dictionary, he will receive a warning message “Sorry, the word does not in the dictionary. While, if he wants to remove an existing word, the operation will be successful and then the user would receive a JSONString message which contains the message “The word has been deleted”. Similar to word addition, the operation of successful removing will also happen both in the memory and in the hard disk. While the user would receive result rapidly in about 3 milliseconds even though the disk writing operation may cost about 2 seconds.

6. Conclusion

In conclusion, the system designed here can finish the functional requirements of searching, removing and adding words concurrently by multiple users. Additionally, this design also provides a proper failure handing model and some optimization operations to give the user a faster and better experience. What’s more, it is open and easy to develop in the future.

Reference:

[1] https://github.com/matthewreagan/WebstersEnglishDictionary