Distributed System Assignment 1 Report

Name: Pengbo Xing

ID: 1287557

Date: 2022/04/19

1. Problem context

In this project, it realizes the basic communication between client and server. In a word, multiple clients send requests to server and server reply to it by using TCP java socket. And the server using multi-threaded to deal with the requests from server at the same time, I use thread-per-connection architecture which will create a thread for each client connection. There are four basic requests that client send to server, query, add, delete, and update. When the client sends query operation to server, the server will search the word in the dictionary and replay the meaning to the client, when the client send add operation to server, the server will add the word and meaning that client gives to the dictionary. When the client send delete operation, the server will remove the word and its meaning when the client sends update operation, the server will update the meaning of word to a new meaning that client give. (I will talk about the creativities I made in following parts)

2. Components of the system

There are three components in the system, they are Client, Server, and UI. More specifically, there are two user interfaces, one for Client and one for Server. In Client, it is used for handling the user activities from client UI, like mouse click, then send it as request to Server, and handling the Server replay. The Server is used for handling the request from Client and show it in Server UI and replying to the Client.

3. Class design and Interaction diagram

a) Class design

There are five classes, Client, Server, ThreadConnection, ClientUI, SerevrUI. In Client, it just has a main method, during the main method, it just creates a socket object, and then create I/O streams for communicating (sending and receiving) with the server. And its communication really happening in ClientUI.

In ClientUI, there are run method for running the whole system when Client calls it. And initialize method is the main operation we did in UI, There are four blanks for receiving input and one blank for output,

- Word blank is for receiving words that user give, for querying, adding, updating, deleting.
- Meaning blank is for receiving meanings that user give for adding, updating.
- Import File blank is for receiving the filename that user give for importing that file to our dictionary.
- Export File blank is for receiving the filename that user give for exporting out dictionary to that file.
- Last blank is for showing any reply from server, like any mistakes the user made when using the ClientUI, tips of adding a word successfully…
- There are six buttons for sending request to Server.
- Query the Word button, when clicking it, the Client will send the word to Server using the self-design message exchange protocol: “Query-SPL-Word-SPL-” also

it will wait for response (successful or not) from server.

- Add the Word button, when clicking it, the Client will send the word and meaning to Server using the self-design message exchange protocol (will not show that specific in following method): "Add-SPL-Word-SPL-Meaning" also it will wait for response (successful or not) from server.
- Delete the Word button, when clicking it, the Client will send the word to Server also it will wait for response (successful or not) from server.
- Update the Word button, when clicking it, the Client will send the word and new meaning to Server, also it will wait for response (successful or not) from server.
- Import File button, when clicking it, the Client will send the filename to Server, also it will wait for response (successful or not) from server.
- Export File button, when clicking it, the Client will send the filename to Server, also it will wait for response (successful or not) from server.

Back to Server, it has three methods, readDict, writeDict, and main. readDict read the dictionary file to our HashMap, writeDict write our HashMap to the dictionary file. During the main method, it opens a server socket, and wait for client request, when one connection is made, it opens a thread for that connection, and wait for the next connection. If another one comes, it makes a new thread for the second connection and wait for the next.

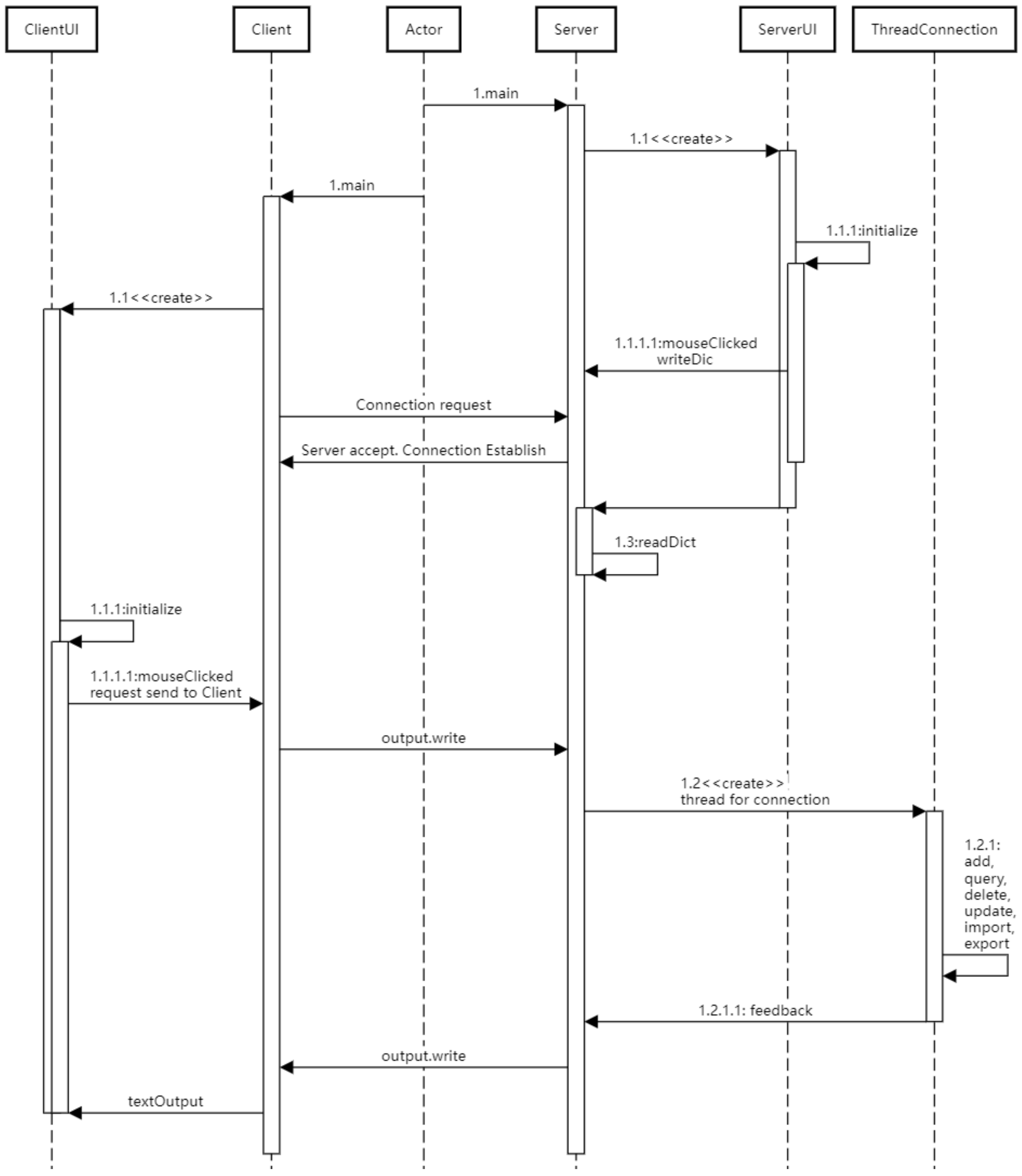
In ThreadConnection, that's the real place for operating the request from Client. There are 9 methods: ThreadConnection constructor, query method, add method, delete method, update method, export method, exportFile method, importFile method, and run method.

- ThreadConnection constructor, Server using it for make a connection.
- Run method, Server call this to run multithread server. It deals with the request from client and split it into different parts using the self-design split note "-SPL-", it split it into method, word/filename, meaning. Then check the method for further operation.
- Query method, check the word and reply it meaning to Client.
- Add method, add the word to dictionary (HashMap)
- Delete method, remove the word and its meaning from dictionary.
- Update method, delete the word and its meaning first then add the add and its new meaning in.
- Import file method, import the data in filename to out HashMap, my self-design format is "Word-Meaning"
- ExportFile method is helper function for export. Export method just export the HashMap data to a file.

In ServerUI, it's run method for running the UI when server call it. It has a table which using for store Client information and operation a client did. It only records the successful operation. And there is a button called "Clear Dictionary" when click it, the HashMap will be cleared all data will be removed.

b) Interaction diagram

Server&Client Interaction diagram



4. Critical analysis and conclusions

When I design the Server UI, I am thinking that I could first control all client connection, like disconnect one of them if I want, second see their operation, but fail to do the first one. I also decide to design a login/register for Client, also for each account, I want to build a favorite that they could add their searched word in their favorites. But that’s too complicated for me and I don’t have that much time. I will try to achieve them in the future.

I am using synchronized on query, add, delete, update, importFile, and export method, in this case, the dictionary is locked when one user does something on it. They could not operate it at the same time.

Some difficult that I meet: understanding the client and server are not difficult, because we already see a lot both in class and tutorial. The first difficult for me is the architectures, it’s easy to understand, but hard to implement (fully understand) when first do them. Finally, I fully understand them and choose connection per thread, Because I think that is a balanced architecture based on time cost and hardware cost, so it’s very suitable for this project. After user establishes a connection, we establish a thread for this connection and run it until that client close the connection. It saves the cost of establishing and destroying thread. We don’t need to create lots of threads like thread-pre-request. And it’s more flexible compared to the worker pool architecture, we don’t have to limit the number of threads that server using.

Second part, not difficult but waste me lots of time. I forget to put “\n” when sending requests, in this case, the other side could not receive my information. I was very difficult to find what’s wrong because I check my code lots of times but could not find error. Then I compare my simple practice code with tutor’s demo. Finally find the error.