COMP90015 Distributed System Assignment1

Name: Renfei Yu (Klaus)

SID: 1394392

Login Username: reyu

Introduction

1. Problem Context

This assignment requires the creation of a dictionary server that can be used by multiple users at the same time using a client-server separation model. In this assignment, we have been explicitly asked to implement the project using threads and sockets so that multiple users can access the server and perform add, delete, modify and query functions on the dictionary in the server at the same time, and our project will be able to clearly handle all kinds of errors and unexpected operations. Also, our project will provide Graphic User Interface (GUI) for user friendly operation.

2. Socket and Multi-threaded techniques

I used the Thread per connection model in this project because I felt it was easier to implement and better suited to the requirements of the project, using this model would create fewer server requests and use fewer resources than the Thread per request model.

The project requires each client to be able to connect to the server and interact with the dictionary, so we need stable links to ensure the user experience. So I used the TCP protocol, which ensures a stable and reliable connection between the user and the server, and TCP is also better implemented and less error prone. Most importantly, it ensures the order of arrival of user actions, which we need such as adding a word by user 1 and then user 2 can change it, so guarantee order arrival is very important. Although TCP is slower than UDP because TCP needs to wait for an acknowledgment from the server, but for the reliability concern, we have to use TCP as well.

System Components

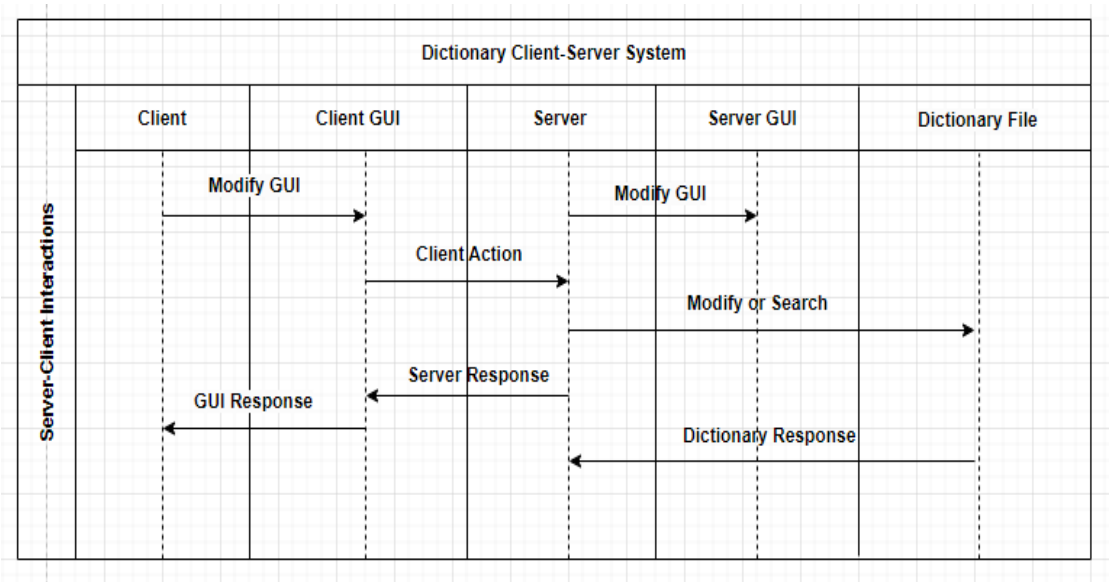
The multi-threaded dictionary is implemented in five parts: Client, Client GUI, Server (containing the main Server and the client handler to deal with concurrency), Server GUI and the dictionary file. These four parts are implemented with three files: Client, Server and clientHandler.

1. The server has the role of dealing with client actions and do modification or query to the dictionary.
2. The client Handler will help server to deal with the multi-threaded implementation that male sure multiple users can access the server simultaneously.
3. The server GUI will display the information about dictionary, use connection

and user actions history.

4. The client is responsible for processing the server's responses and return information then displaying them on the Client's GUI.
5. The Client GUI displays an interface and gives the user intuitive feedback.
6. Dictionary is a file created by server and will store the dictionary for future purpose

In the following flowchart we can see the interaction between the client, the server and the dictionary file.



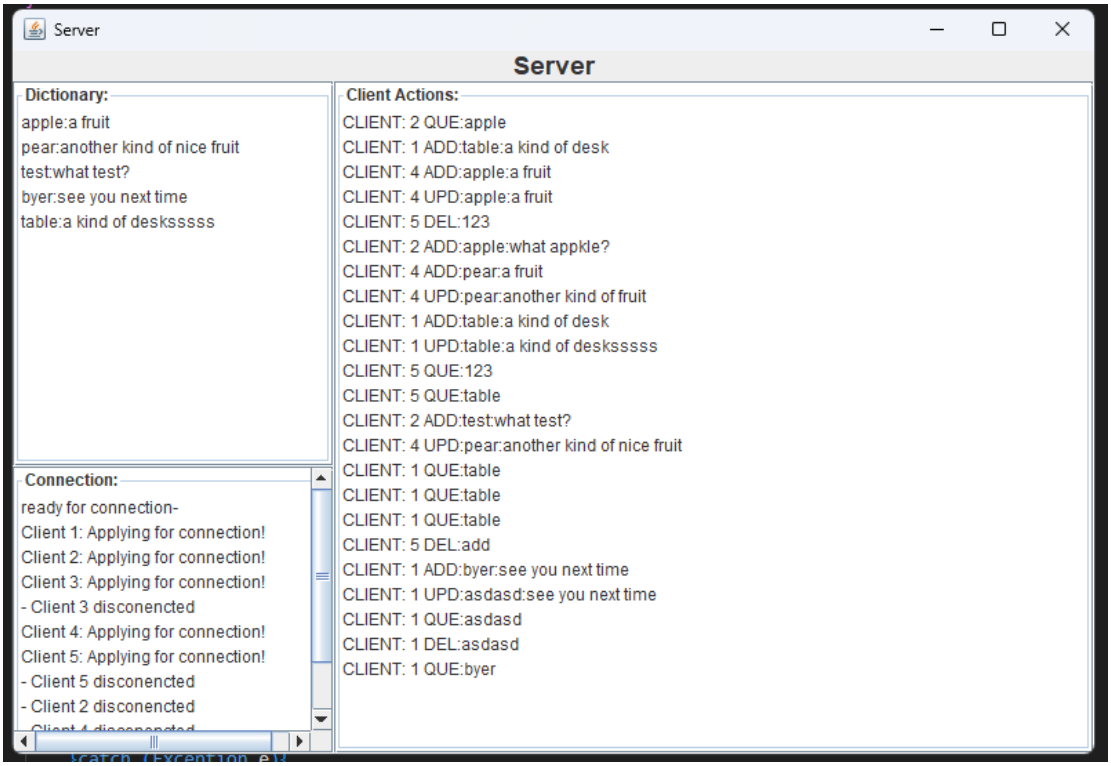
flowchart of the system process

Class Design

1. Server

Dictionary server as the centre of processing user requests, will set the server port and dictionary file through input argument, if the user input dictionary file is not found, then the dictionary file will be automatically created by the server, in each server startup, the program will automatically store all the words in the dictionary in the local ConcurrentHashMap for later use. After each change to the ConcurrentHashMap, the program will automatically make the same changes to the dictionary file to ensure that the program's ConcurrentHashMap and the dictionary file are always synchronised. The server accepts user requests and establishes a connection, if the connection is successful, it allocates a new thread and hands it over to the clientHandler to handle the user requests. The server has its own GUI, all the user's connections or disconnections, the user's history of requests and the contents of the current dictionary will be displayed on the server's GUI.

The following screenshot shows the interface of Server GUI



Screenshot for Server GUI

2. clientHandler

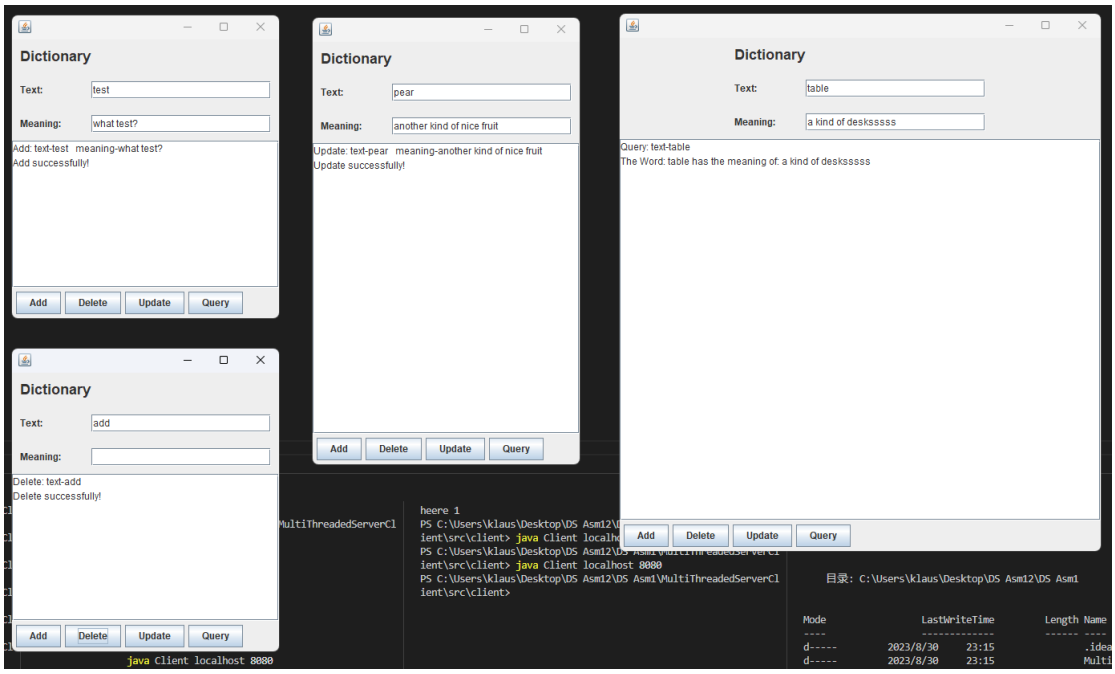
ClientHandler will extends Thread and handle the concurrency implementation for the project. Each clientHandler will be used to handle one client for the server since this is a Thread-per-connection model. When user sends request, the clientHandler will analyze the request as “Actions (add, delete, update and query)”, “Text (the word)” and “Meaning (the meaning of the word”.

3. Client

The client class is used to set a new client and connect to the server, also each client is related to its own GUI and the actions on GUI will

be sent to the server through client. There is some simple detection of user's input on the client side, for example add and update requires both text and meaning field to be filled, delete and query requires text field to be filled, otherwise there will be an error shown on the GUI.

The following screenshot is what the client GUI looks like



Screenshot of client GUI

Excellence

In the project, all kind of errors are been handled properly in all the classes, incorrect input or the server-client connection problem are handled in both client side and server side.

1. Server

1.1 When running the server, the user needs to input the server port and dictionary file, if one of them is missing, an error will be reported, prompting the user "You should input the <port number> followed by the <dictionary file name>"

1.2 The server will create a server based on the user's port input, and if the creation is unsuccessful, it will output "This is an IO exception -server" to prompt the user.

1.3 When the user wants to link to the server, if the connection fails the server will report an error, reminding the user of "IO exception at this point - server".

1.4 If a problem occurs while the server is manipulating the dictionary, an error will be reported depending on whether it is a read or a write.

2. ClientHandler

2.1 If the request received from the user is query or delete, but the word the user wants to query does not exist, an error will be reported on the user GUI "Error: This word does not exists".

2.2 If the request received from the user is add or update, but the word the user wants to query already exists, an error will be reported on the user GUI "Error: Already exists this word".

2.3 If the user closes the client GUI or exits in the terminal, the server GUI will display "- Client "+client number+" disconnected").

3. Client

3.1 When running the client, the user needs to input the address and port, if one of them is missing, an error will be reported, prompting the user "You should input the <server-address> followed by the server <port number>".

3.2 If the target server port that the client wants to connect to is not up and running when the client is running, the message "The server is not ready" will be displayed.

3.3 If there is no input in the Text area when the user clicks the delete or query button, an error will be reported "Error: Text area should not be empty".

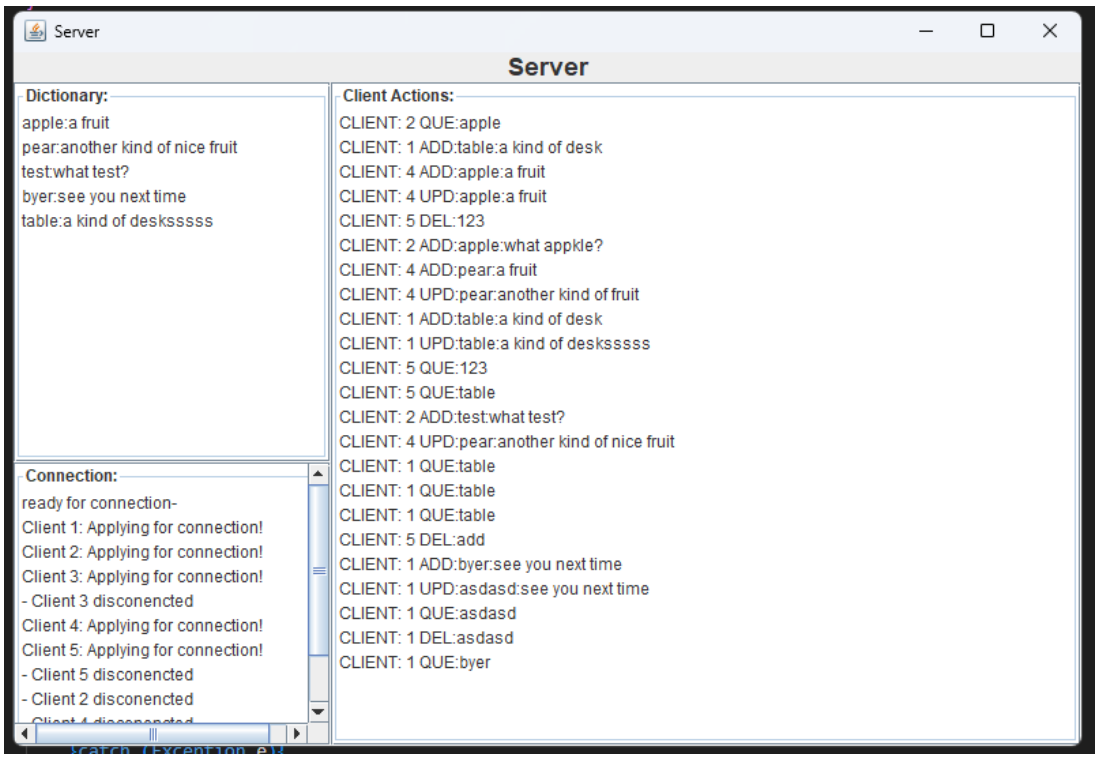
3.4 If the user clicks the add or update button but there is no input in either or both of the Text or Meaning fields, the error "Error: Text and meaning area should not be empty" will be reported.

Also, in this project I use TCP for reliable communication between clients and server. Also, the thread-per-connection are implemented properly to save resource and be reliable.

Creativity

In this assignment, I implemented the server GUI, all the user's connections or disconnections information, the user's history of requests and the contents of the current dictionary will be displayed on the server's GUI.

The following screenshot shows the interface of Server GUI



Screenshot for Server GUI

Conclusion

The dictionary system uses TCP and Thread-per-connection model to provide a reliable connection between server and clients, also implemented GUI for both client and server to make it more user

friendly. The system also implemented good error handling to cover almost every possible cases. But there still opportunities for improvement such as make the concurrency optimizer by implement customer thread pool.