COMP90015 Project 1: Multi-threaded Dictionary Server
Christina Xu (#945756)

Introduction

Multi-threaded dictionary server allows concurrent clients to connect and make requests. There are three types of requests supported by the server: query the definition, add a new word, and remove a word. Sockets and threads are used as fundamental technologies in this project.

In this project, the multi-threaded server is implemented by a thread-per-connection architecture. Socket choice is TCP. A JSON self-made dictionary with less than twenty words is used.

The System

- **Architecture**

Thread-per-connection is chosen as the architecture. The server is able to support multiple client connections. One thread is created for each client if the client is connected to the server successfully. The thread can take as many requests as client needed.

- **Service**

The dictionary server supports three kinds of operations: query, add, and remove.

- **Query:** Query the meaning of the word.

  If the word is in the dictionary, it returns the definition of the word.

  If the word is not in the dictionary, it returns error messages.

- **Add:** add word and definition to the dictionary.

  If the word is in the dictionary, it returns error messages.

  If the word is not in the dictionary and the definition input is not null, the word and definition will be added to the dictionary.

  If the word is not in the dictionary but the definition input is invalid, it returns error messages.

- **Remove:** remove a given word from the dictionary.

  If the word is in the dictionary, the word will be removed from the dictionary.

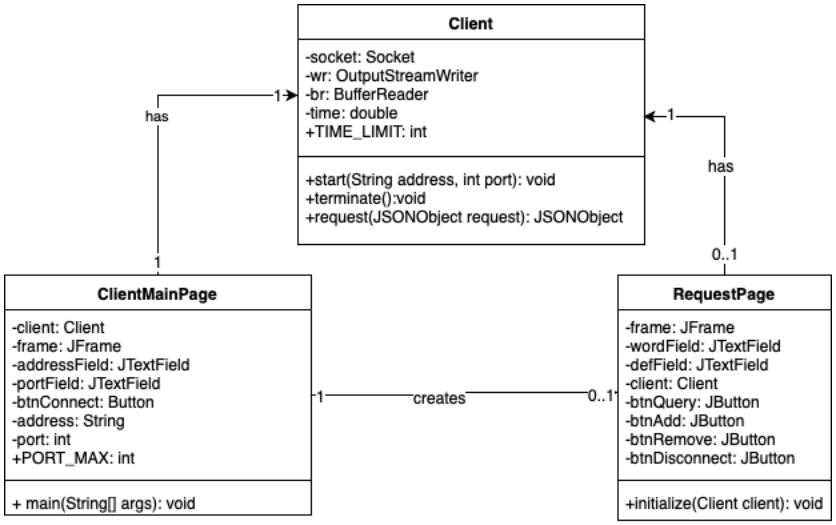
  If the word is not in the dictionary, it returns error messages.

Class Design

(This section shows UML for the client and server, and the sequence diagram.)

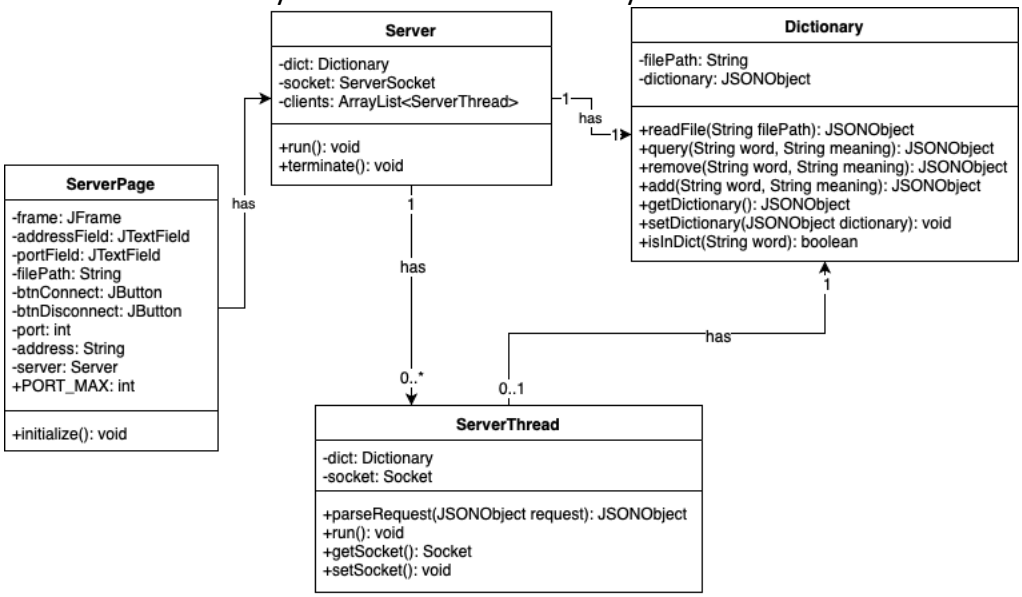
- **The Client**

The client contains three classes: Client, ClientMainPage, and RequestPage. The main function is inside the ClientMainPage class. The ClientMainPage lets the client makes connection to the server with correct address and port inputs. The RequestPage lets the client makes query/add/remove requests to the server.



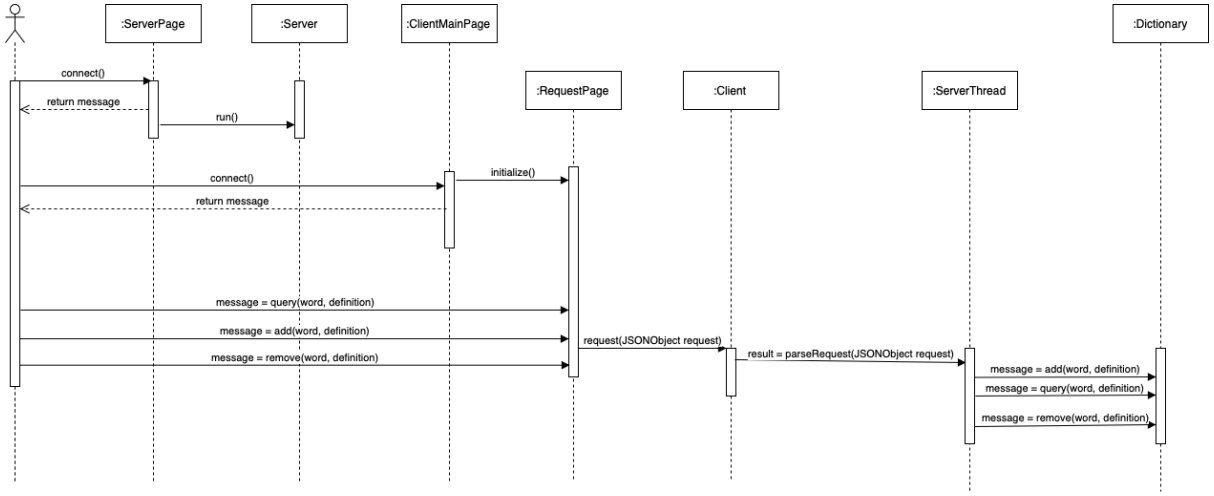
- **The Server**

The server contains four classes: Server, ServerPage, Dictionary, ServerThread. The main function is inside the ServerPage class. The ServerPage lets the server connect given an address and port number. The Server class can hold a list of clients(threads). The ServerThread class handles requests and parse requests to response. The Dictionary class reads JSON dictionary file and handles dictionary services.



- **Sequence Diagram**

The sequence diagram shows the interaction between server and client. The server needs to be connected first and the client can be connected to the server afterwards. The model supports three types of dictionary operations.

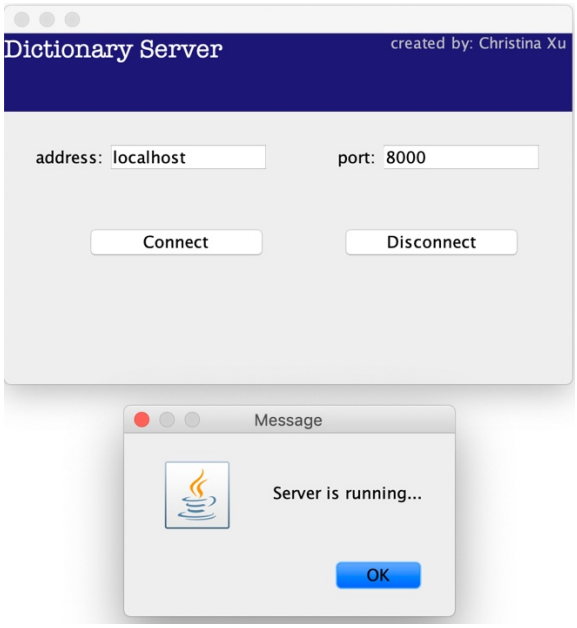


User Interface

The GUI implementation used for this project is Java Swing. Though it is not required, but the system contains GUI for both server and client.

- **The Server**

The server only has one page. When inputs for address and port are valid, a return message ("Server is running…") will be shown after the connect button is pressed. When inputs for the address and port are not valid, an error message will be shown after the connect button is pressed. If disconnect button is pressed, the server will shut down.

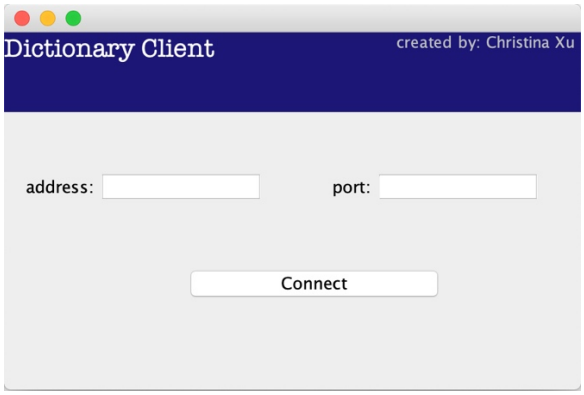


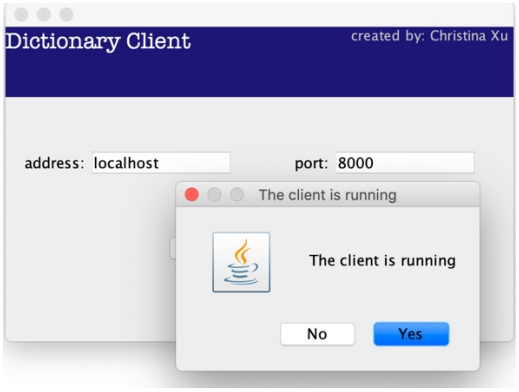
- **The Client**

The client has two pages.

1. login page (main page):

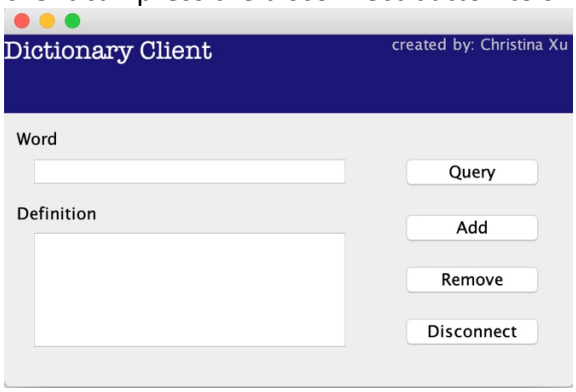
If the connect button is pressed, and address as well as port inputs are valid, the system will show a return message ("The client is running").





2. request page:

On this page, the client can choose to query/add/remove word. After the client finishes, the client can press the disconnect button to shut down.



Critical Analysis

- **Concurrency**

The shared resource is the JSONObject dictionary. Query/Add/Remove operations are synchronized. Since the methods are synchronized, multiple users can access the same dictionary entry. For example, client A and client B can both query for the word “DS”, and the result of “Distributed System” will be returned.

Because the sharing dictionary object is updated, clients can access the updated dictionary all the time. For example, if client A adds a word that is not in the dictionary successfully with the input “cola” as a word and “a magic drink that brings happiness” as the definition, then client B can query the word “cola” and receives the definition of it.

- **Thread-per-connection**

In this architecture, each client has a connection to the server. Because the thread is built per connection, each thread process is responsible to handle as many requests as the client needed. It is noticeable that to keep efficiency, in this design, if no requests have been made by the client within a time limit, the program will automatically kill the process.

Conclusion

The dictionary server, implemented a thread-per-connection, is designed to serve multiple clients concurrently. Socket communication is reliable using TCP. The self-made dictionary file is in JSON. The client and the server have their own user interface, implemented by Java Swing.