COMP90015 Distributed Systems
Assignment 1 Report

Liguo Chen
Student ID: 851090

April 17, 2021

Problem

In this assignment, the problem to be solved is to build a server, which provides word meaning add/lookup/update/remove services. Together with the server, a client is also built, with which a user can interact to communicate with the server to accomplish a lookup task.

Some technical requirements for the server includes:

- the only way of communicating with the client is through socket
- communication between the server and the client has to be reliable
- needs to be able to serve multiple clients at the same time

Some technical requirements for the client includes:

- a graphical user interface(GUI) is needed for user to perform any operations

System Components

The system contains two components: one for server and the other for client.
For the server component, there are three modules in this components:

- Server module: this module is the main module of the server component, containing the logic of the server(e.g. establishing connection, distributing tasks, etc)
- GUI module: this module contains the logic for the server graphical user interface(GUI)
- Task module: this module contains the detailed logic about the incoming request task to be performed by the server

For the client component, there are two modules in this components:

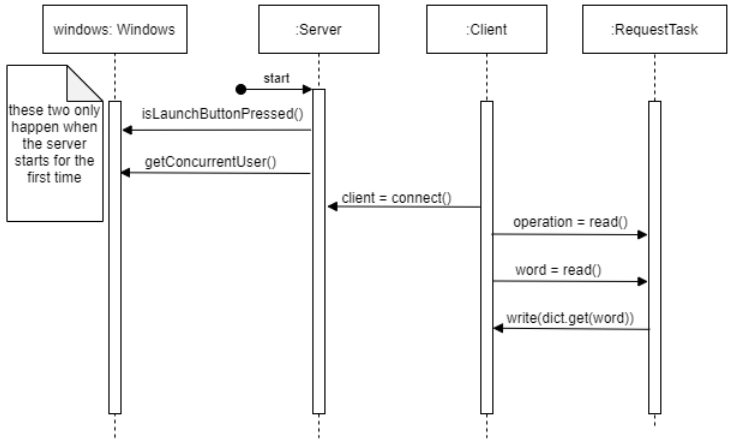
- Client module: this is the main module of the client component, containing the logic of the client(e.g. connecting to the server, sending request, etc)
- GUI module: this module contains the logic for the client graphical user interface(GUI)

Design

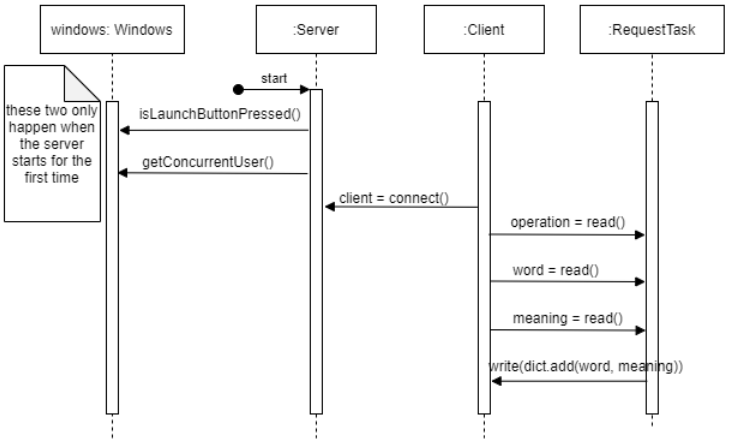
The server component contains the following classes:

- ***Server***: this is the main class of the server
- ***Windows***: this is the class responsible for the GUI of the server
- ***RequestTask***: this class represents a request task sent from the client

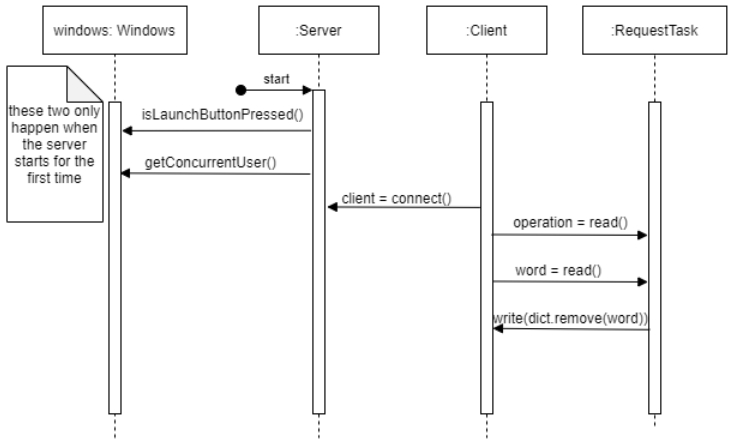
Sequence diagram of the server responding to a **SEARCH** request



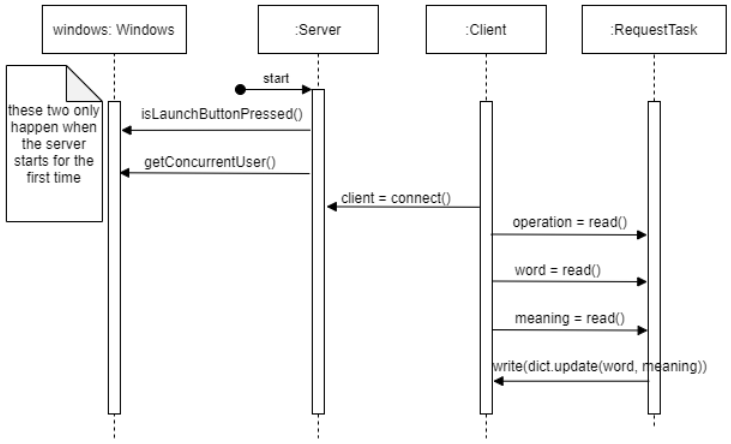
Sequence diagram of the server responding to an **ADD** request



Sequence diagram of the server responding to a **REMOVE** request



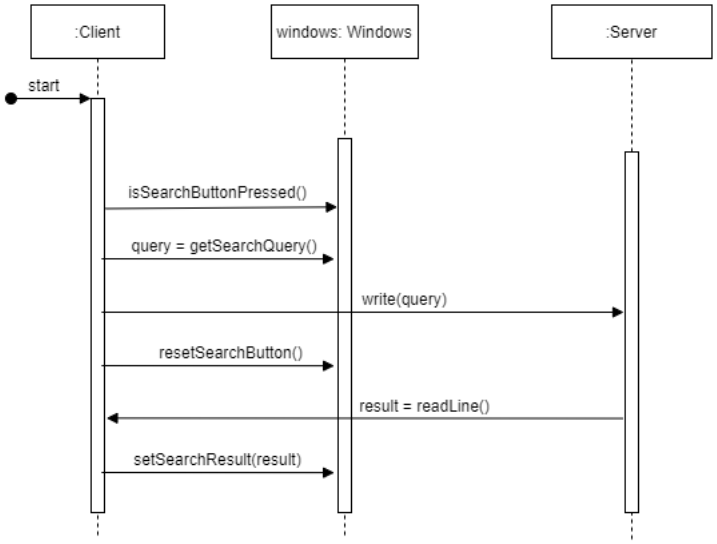
Sequence diagram of the server responding to an **UPDATE** request



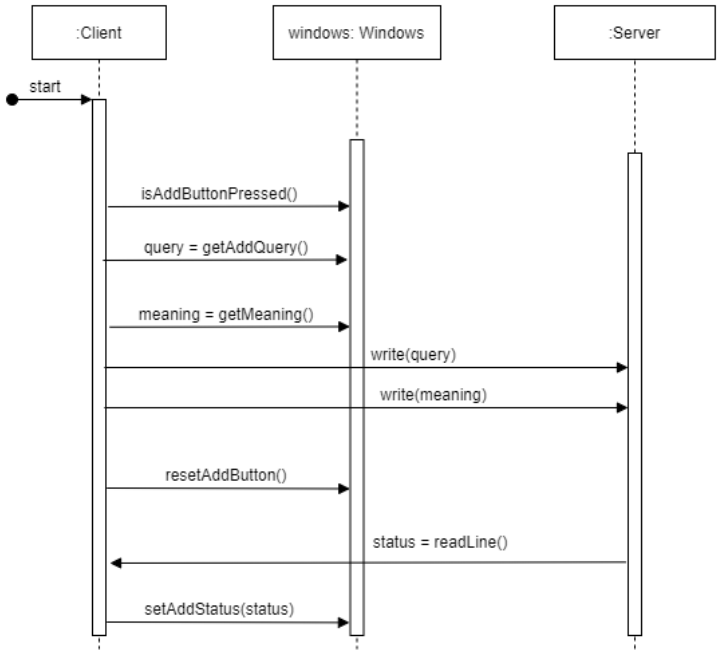
The client component contains the following classes:

- ***Client***: this is the main class of the client
- ***Windows***: this class is responsible for the GUI of the client

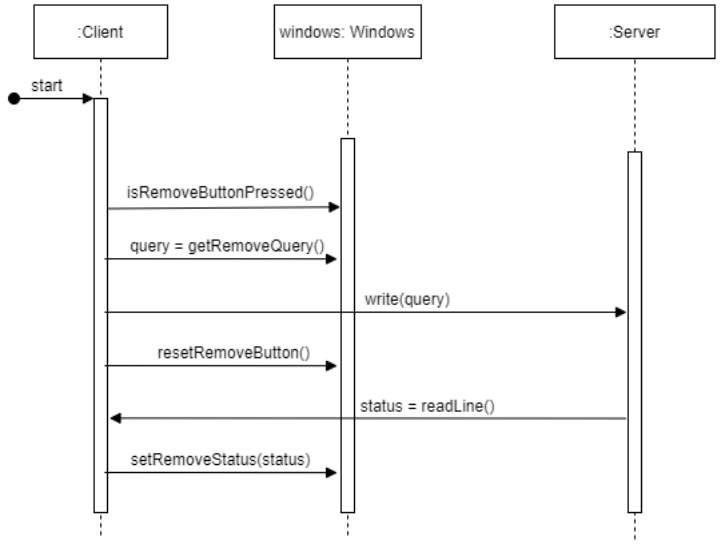
Sequence diagram of the client making a **SEARCH** request



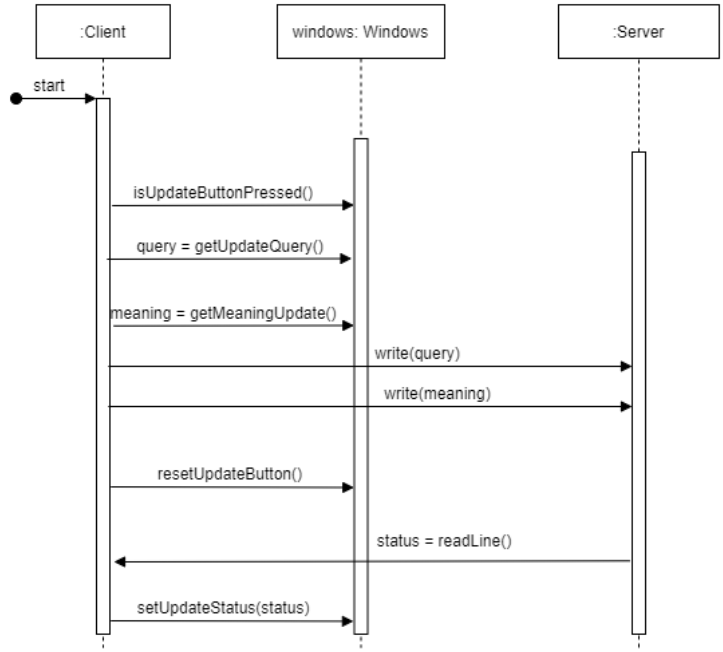
Sequence diagram of the client making an **ADD** request



Sequence diagram of the client making a **REMOVE** request



Sequence diagram of the client making an **UPDATE** request



Analysis

The system

- TCP connection is used to establish the communication between the server and the client. Therefore, each request is delivered to the server safely and the meaning returned from server is accurate(i.e. data is not corrupted on the way).
- the concurrency ability of the server is achieved by using a thread pool. As a result, the server has better performance and avoids having latency in execution due to frequent creation and

destruction of threads for short-lived tasks.

- Different from normal web servers, serving web pages for client, the threads in the pool in my system is distributed based on client connections. Each client will be given a thread for executing requests. The thread is released into the pool when the client disconnects. The consideration is that, unlike web server, where it is rare that the same client requests multiple web pages within a short period of time(it takes time to read a web page) and it's wasteful to have a dedicated thread for each client, my dictionary server may have clients wanting to search the meanings of many words(meaning of a word takes shorter time to read). Thus, allocating a dedicated thread for each client is a sensible choice in this scenario.
- the meanings of words is stored in a hashmap in the server main class. This resource is not locked when a thread is using it as the probability of conflict is quite small and not restricting the access results in better performance of process client requests(no need to wait for the lock to be released)

Conclusion

The server implemented meets the minimum functional requirements and works as a MVP(minimum viable product). For future improvements, the work can focus on GUI prettification, additional types of requests and pop-up showing error messages.

Excellence

- program related errors are displayed on the terminal
- for the client, server related errors are displayed on the GUI, e.g. when the server is lost
- advantages:
  - use of thread pool to achieve the concurrency while avoiding the latency of thread creation and destruction (as discussed above)
  - enables the user of the server to choose the size of the thread pool to better suit the hardware setup
- disadvantages:
  - the thread-per-client design is not suitable when there is a need for large number of users to connect to the server at the same time

Creativity

- implementation of GUI for the server, which enables the user of the server to choose a size for the thread pool and shut down the server
- use json, which is widely-used file format, to store the words and meanings as the initial data. It's easier to integrate with other systems later. Also, there are many efficient json parsers that can be used for json file containing millions of words.
- request results returned from server are assigned a color based on their status: green for success(in case of search, there is a match), red for failure(in case of search, there is no match). So, it's easier for the client user to get the feedback