COMP90015: Distributed Systems – Assignment 1

Name: YICHENG JIN

User Name: Yicheng Jin

Student ID: 962546

1. Context

According to a client-server architecture, designing a multi-threaded server as a dictionary that allows concurrent users to query, add, remove words. The server implements a worker pool architecture which allows up to 100 clients to access at the same time. For interaction, the communication is based on TCP.

2. The components of the system

The system is divided into two parts: client and server, and both of them have their own GUIs.

For the server, firstly it creates a socket which bounds to a port and waits a connection from a client. Before a connection is established, it will block on I/O and do nothing. When a connection is accepted, the server will create a new socket to wait for a connection from other clients. Therefore, it needs a worker pool to implement multi-threaded. In this assignment, it is designed to run at most 100 threads concurrently, and if the number exceeds, the new threads will be added to the cache queue. In a particular thread, the server will receive commands and deal with them. According to the specific requirements, the server send message back to the client, and modify the document of dictionary. It is worth mentioning that the dictionary is saved as a JSON file.

For a client, it will request connection from the server at the beginning. Then, the client can send requirements which including querying the meaning, adding new words, removing existing words, and killing the thread to the server. When the server returns message, it will be displayed on the GUI of client.

3. Class design and interaction diagram

3.1 . Class design

The project has five classes. Two of them belong to the client (Figure 1), the rest belong to the server (Figure 2).

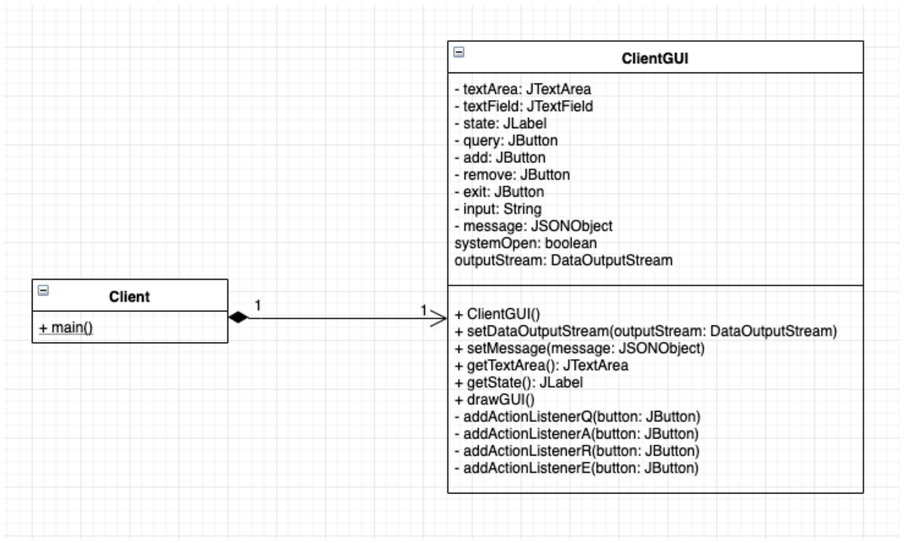

ClientGUI
- textArea: JTextArea
- textField: JTextField
- state: JLabel
- query: JButton
- add: JButton
- remove: JButton
- exit: JButton
- input: String
- message: JSONObject
systemOpen: boolean
outputStream: DataOutputStream
+ ClientGUI()
+ setDataOutputStream(outputStream: DataOutputStream)
+ setMessage(message: JSONObject)
+ getTextArea(): JTextArea
+ getState(): JLabel
+ drawGUI()
- addActionListenerQ(button: JButton)
- addActionListenerA(button: JButton)
- addActionListenerR(button: JButton)
- addActionListenerE(button: JButton)
Client
+ main()
1
1

Figure 1: client class diagram

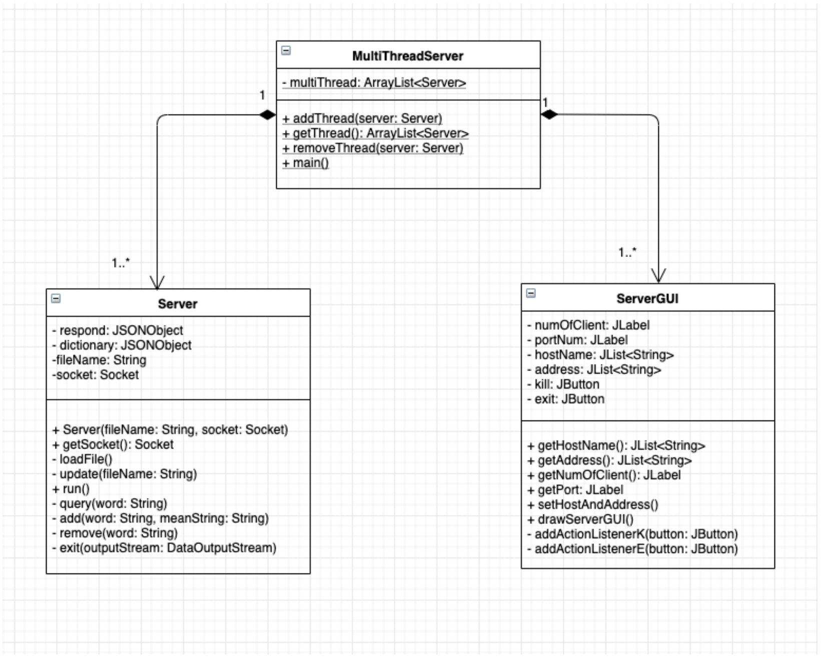

MultiThreadServer
- multiThread: ArrayList<Server>
+ addThread(server: Server)
+ getThread(): ArrayList<Server>
+ removeThread(server: Server)
+ main()
1
1
1..*
1..*
Server
- respond: JSONObject
- dictionary: JSONObject
-fileName: String
-socket: Socket
+ Server(fileName: String, socket: Socket)
+ getSocket(): Socket
- loadFile()
- update(fileName: String)
+ run()
- query(word: String)
- add(word: String, meanString: String)
- remove(word: String)
- exit(outputStream: DataOutputStream)
ServerGUI
- numOfClient: JLabel
- portNum: JLabel
- hostName: JList<String>
- address: JList<String>
- kill: JButton
- exit: JButton
+ getHostName(): JList<String>
+ getAddress(): JList<String>
+ getNumOfClient(): JLabel
+ getPort: JLabel
+ setHostAndAddress()
+ drawServerGUI()
- addActionListenerK(button: JButton)
- addActionListenerE(button: JButton)

Figure 2: server class diagram

3.1.1 Client

The class Client only has a main method, which creates an object of Socket to connect with the client. It also implements the code of receiving message and displaying on the GUI of client.

3.1.2 ClientGUI

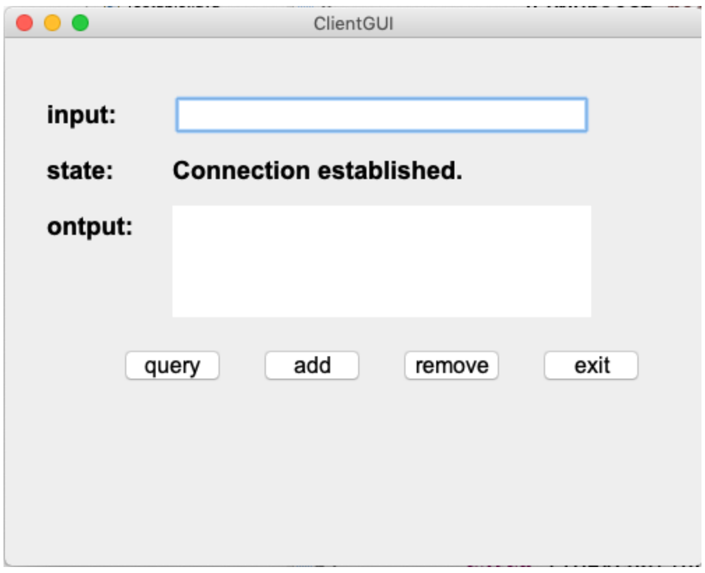


Figure 3: GUI of client

The class of ClientGUI displays a user interface (Figure 3). When the GUI is opened, the connection has already established. Users can input message by an object of JTextField, and get the state(JLabel) and output(JTextArea) from the server. When a problem occurs, users can also get the error information from the output.

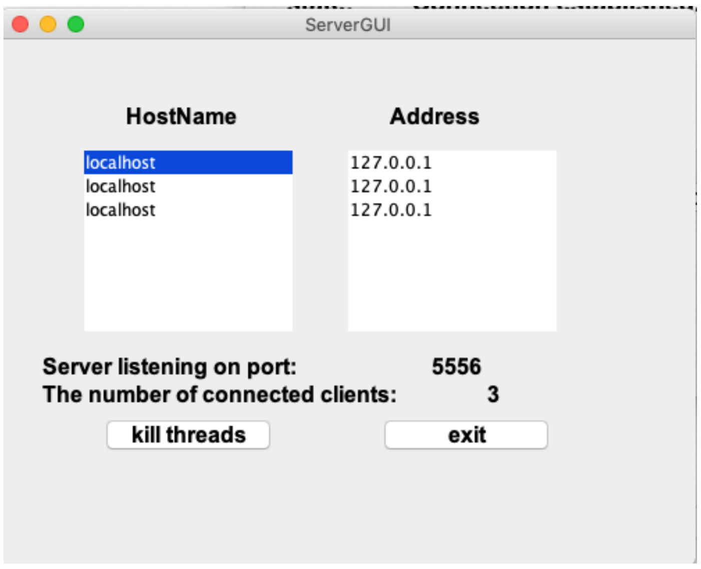
3.1.3 MultiThreadServer

This class is similar to the class of Client, which also display a method of main. It creates an object of ServerSocket. When a client request connection, it will create a new thread from the worker pool to build connection. In addition, it also create a static list to store the message of each thread and relative static methods of the list.

3.1.4 Server

Server class inherits from Thread class, it has a method of run which is called by MultiThreadServer class. In the run(), it processes input information and executes corresponding operations, such as returning word's meaning, adding new words, removing existing words from the dictionary.

3.1.5 ServerGUI



HostName	Address
localhost	127.0.0.1
localhost	127.0.0.1
localhost	127.0.0.1

Server listening on port: 5556
The number of connected clients: 3

Figure 4: GUI of server

The GUI of server can be seen in Figure 4. It displays host name and address of the clients, the listening port number and the number of connections. Moreover, it also allows users to kill specific thread by clicking its host name and close the system.

3.2 . Interaction diagram

The collaboration diagram can be seen in Figure 5, and the sequence diagrams of client and server can be seen in Figure 6, Figure 7.

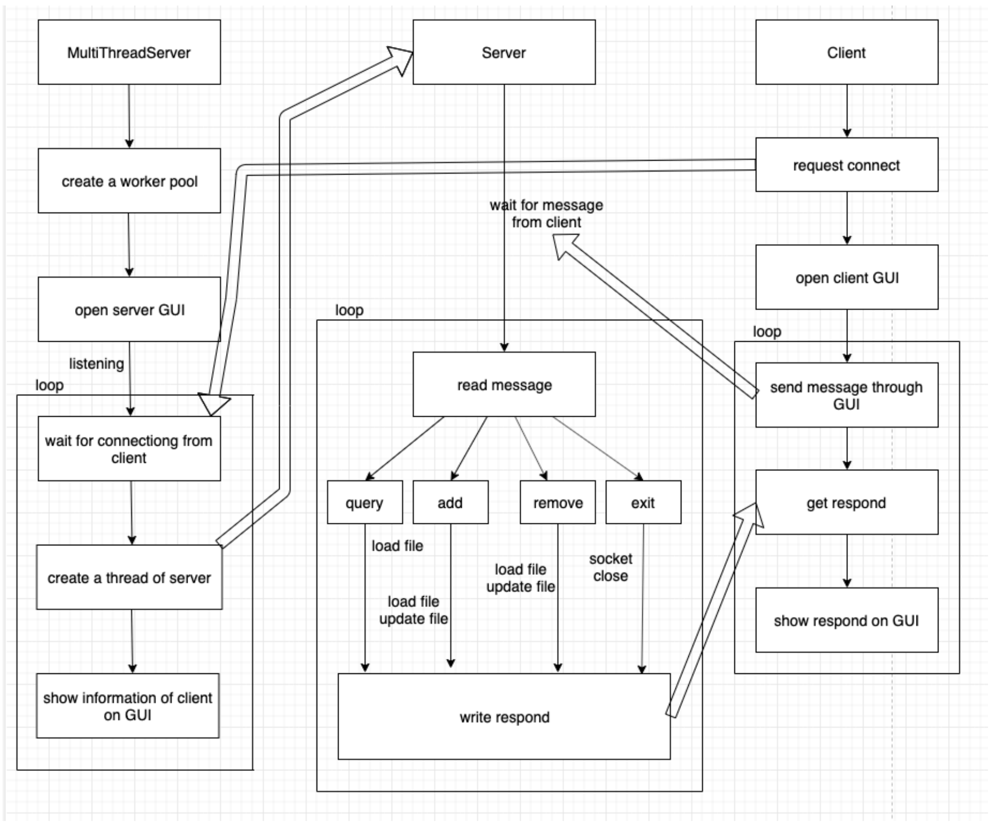
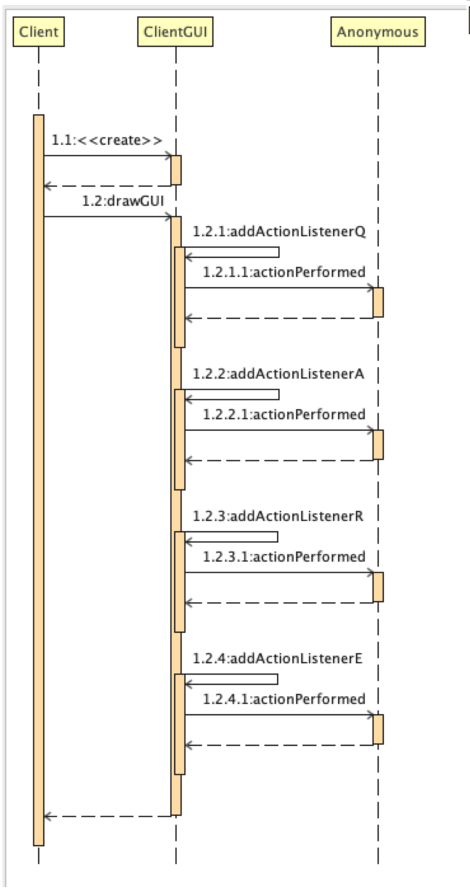


Figure 5: interaction diagram



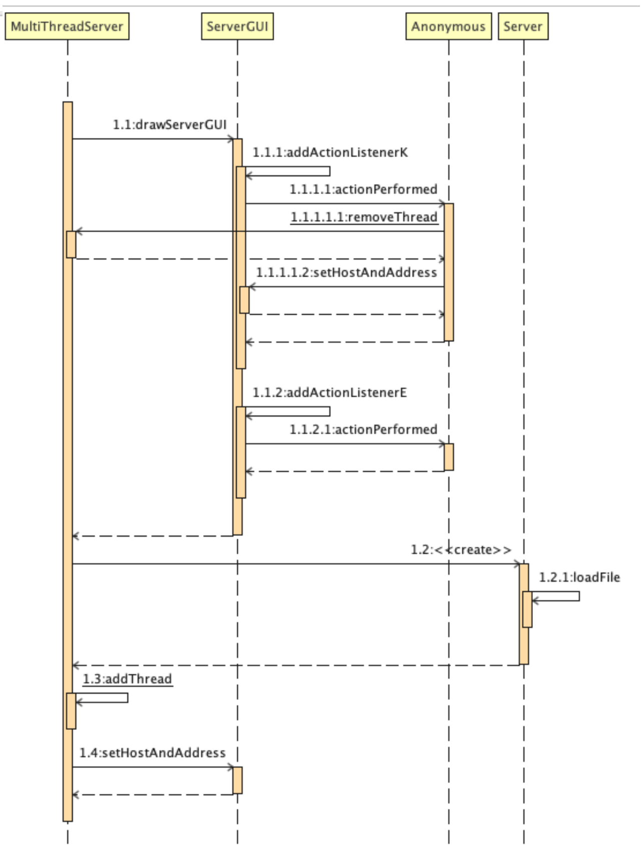


Figure 6, Figure 7: sequence diagram of client and server

4 Critical analysis

Advantages:

i. Implementation a GUI for managing the server, which is convenient for human-computer interaction. through the GUI of server, the manager can get host name and address of each connected client and kill any thread.

ii. Error handling. For most possible problems, the architecture is designed to throw them with proper exceptions and display error information on the GUI, through which the problems can be quickly located and solved.

iii. Data consistent. The methods such as loading file, removing words are designed with using “synchronized”, which can prevent multiple clients access a method at the same time.

iv. Data storage. The dictionary is saved as a JSON file. JSON has two property: key and value which is suitable for storing dictionary. The key for words and value for meanings. Therefore, the data access is convenient and effective.

Disadvantages:

i. Scalability. Up to 100 threads can work simultaneously.

ii. Users cannot be prevented from maliciously adding or deleting words.