COMP90015: Distributed Systems

Assignment 1: Multi-threaded Dictionary Server

Author: RAO YUNHUI

Student number: 1316834

1 Problem Context

In this assignment, a multi-threaded server allowing concurrent clients to search the meaning(s) of a word, add meaning for a new word, remove an existing word and update the meaning of an existing word that use a client-server architecture is required.

In the designed server, a thread-per-request architecture is implemented to avoid resource waste for idle connection, and TCP protocol is utilized to ensure reliable connection between the server and clients. Synchronized function is utilized to achieve concurrent requests from clients. JSON is used to exchanged messages between server and client.

For dealing with program failure, errors are handled in a way that the client will receive brief notification in graphical user interface (GUI) and detailed error message in the stack of the program will be print at the command line of server. Also, when the client itself encounters an error, the detailed error message will be printed on command line while the brief description will be notified on GUI for clearness. The errors include: parameters error, network connection error such as unknown host error, IO error, process interruption error and other errors may be encountered. Besides, the server will deal with illegal requests such as querying, removing or updating a non-exist word, and adding an existing word.

2 System Components

2.1 Server

The server implements a thread-per-request architecture in order to prevent resources waste of idle connection. To achieve this, the server has a server socket that is always up for listening to requests of clients. Once the server socket heard a request, the server will create a thread to handle the request, in which legal requests will be passed to dictionary and illegal requests will be refused and an error message will be sent back to the client. The JSON-format response will be packed using the class Response so that proper format of response is ensured. In the dictionary part of server, synchronized functions are utilized to handle concurrent requests.

2.2 Client

The client implements a graphical user interface (GUI) to make user experience better. The client is always up and create a thread to send request for each action the user makes. The JSON-format message will be packed using the class Message so that proper format of request is ensured. The client is quite light and do not check the requests, leaving the server-end to check

as the server has more information.

3 Class Design and Interaction Diagram

3.1 Server Package

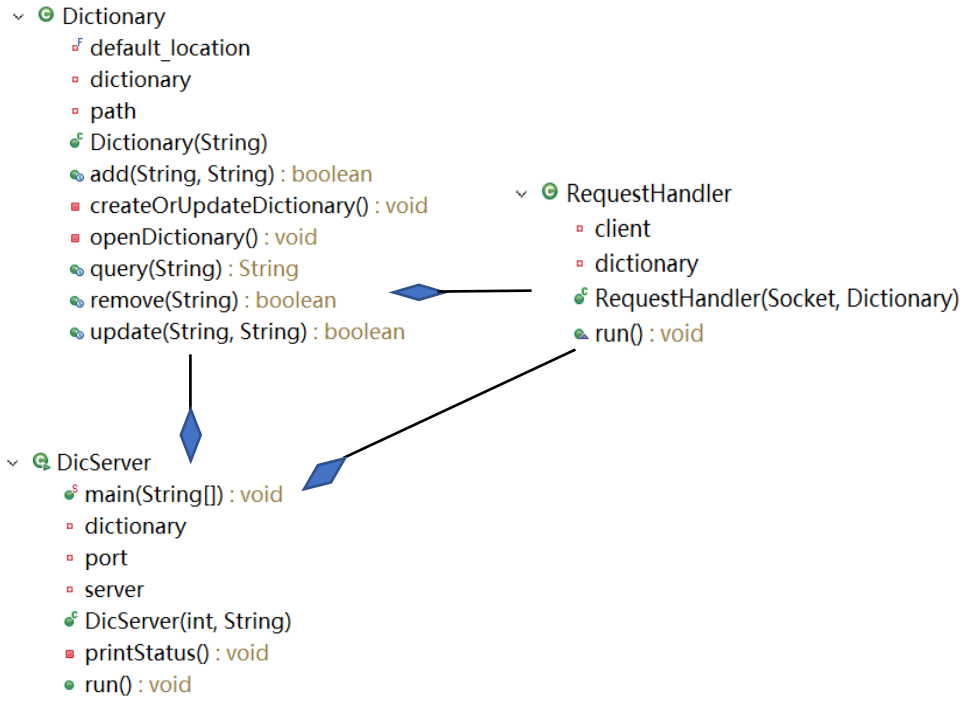

Figure 1: Server Package UML

When the server is called, it first checks the parameters given and will run if they are valid. The parameters should be <port> and <dictionary file>. The server will bind with a port and create a Dictionary object for further processing. Then a server socket is created for listening for clients' requests. Once a request is heard, the DicServer will immediately create a RequestHandler object, which is a thread, to handle the request. The Dictionary object is passed to the RequestHandler for processing legal request. After the RequestHandler processes the request, is will send a JSON back to the client, whose format will be introduced in Response class in common package.

3.2 Client Package

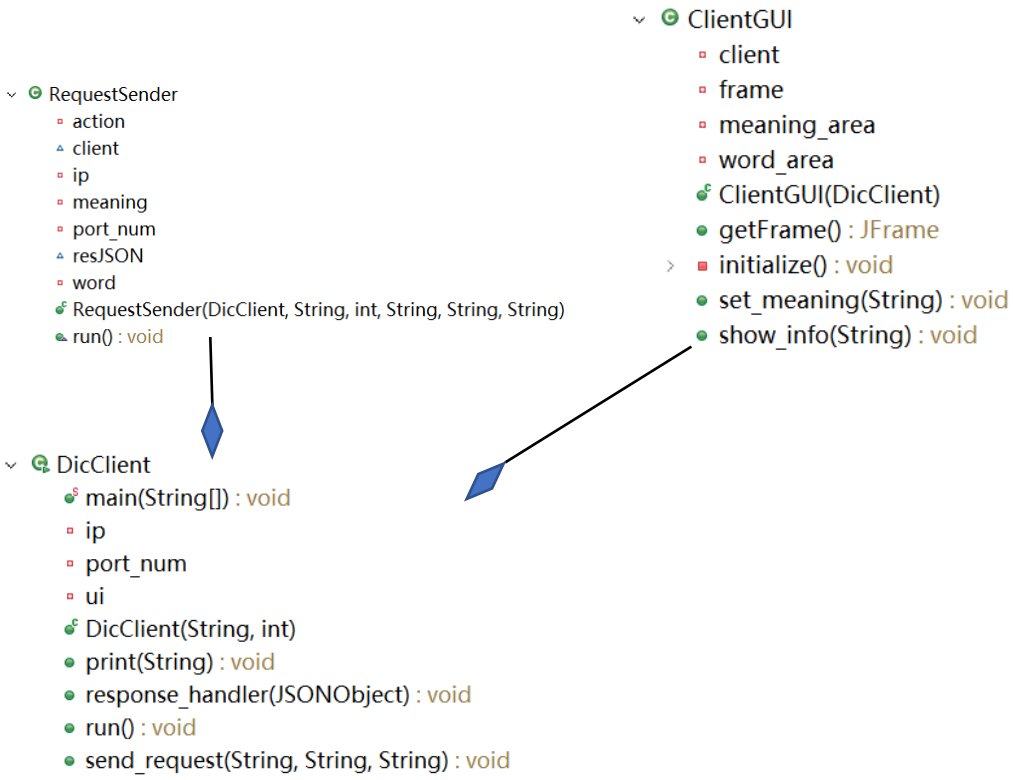

Figure 2: Client Package UML

When the client is called, it first checks the parameters given and will run if they are valid. The parameters should be <server-address> and <server-port>. The client creates a DicClient Object and record the server address and port for further binding. Then the client will create a ClientGUI and show a graphical user interface (Figure 3). The ClientGUI and send the requests users make on GUI to the DicClient and the DicClient immediately creates a RequestSender object to handler the request. The response sent back will be received by RequestSender and passed to DicClient and processed and shown on GUI.

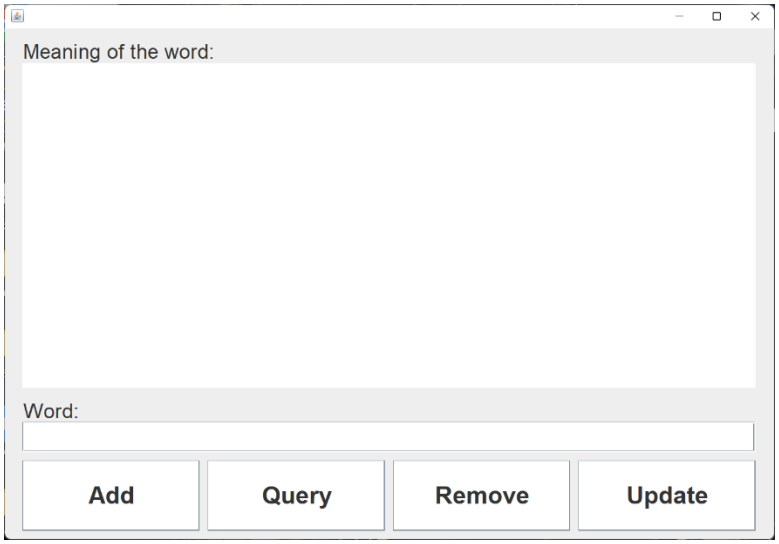

Figure 3: GUI of the client

3.3 Common Package

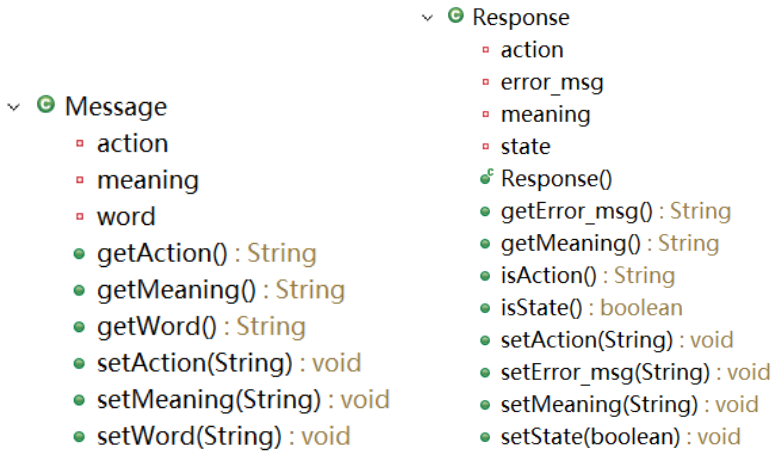

Figure 4: Common Package

The Message and Response class in common package is to help create JSON objects so that JSON objects created are ensured to be in proper format.

The fields of JSON created using Message is shown Table 1 and the fields of JSON created using Response is shown Table 2:

action	One of “QUERY”, “ADD”, “REMOVE” and “UPDATE”, shows what the action the user wants to take
meaning	Need to append this in “ADD” and “UPDATE” action so that the meaning of the word is added or updated in dictionary.
word	The word the user wants to deal with

Table 1: JSON fields of Message-create JSON

state	Whether action was success
action	Action made
meaning	Will be the meaning if the action is “QUERY” and empty if else
error_msg	Will be the error message is any error encountered and empty if no errors happen

Table 2: JSON fields of Response-create JSON

3.4 Interaction Diagram

The interaction diagram between a single client and the server is shown in Figure 5.

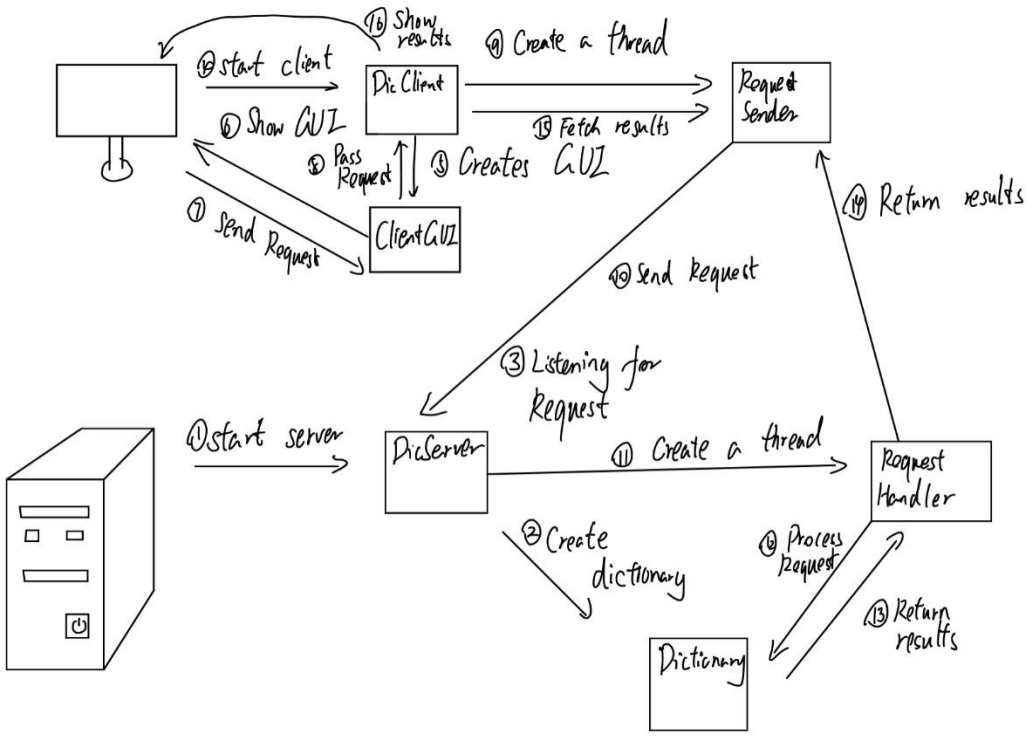

Figure 5: Interaction Diagram

4 Critical Analysis

4.1 Architecture Selection

The options given for implementing server architecture includes thread-per-connection, thread-per-request and worker pool architecture. The thread-per-request architecture is selected considering following reasons: for a dictionary application, the user will be idle for most of the time as the user will only query words from time to time, so the thread-per-connection architecture is not appropriate as it will waste resources. The worker pool architecture also seems good but the dictionary does not have many users so a limitation on number of threads is not a must.

The thread-per-request architecture is scalable as most users won't query words at the same time so the server has the ability of handling many clients.

4.2 Error Handling

For both server and client, the error of parameters and other startup error such as connection error will be handled locally at the command line. The advice of fixing the errors will be shown on the command line and the operator can follow it and restart.

After starting, the errors in user's instructions such as empty meaning when querying will be handled by the server and send an error message to the client and show it on GUI so the client can fix the error. For IO errors that may happen at the server end, the server will show the

error at the server end and notify the client and the client knows that there's something wrong with the server and stop wasting time trying. The server also has the error message so the techniques can quickly fix the error.

Considering all above, the error happens in using process can be handled properly and will not cause a problem in user experience.

4.3 Portability

The program has portability as it is written in JAVA and the JVM machine is supported on all platforms so the codes do not have to be written twice to run on another platform.

4.4 Creativity

4.4.1 Humanization Design

If one of add, remove and update actions is performed by the user, a confirmation window will be shown as the user may accidentally click on the button while no confirmation window will be shown if the user takes the query action as the query does not modify the dictionary and will not cause harm, so it is better to quickly show the result even if the user accidentally clicks on the query button.

4.4.2 Timeout Handling

Although the reliability is ensured by the TCP protocol, the timeliness is not ensured. If the internet is in congestion and the response arrives too slow (1 second time limit is set), the client end will notify the user that it takes too much time for waiting for response. Then the user can make a note on the word and read through the sentence, waiting for the internet to be clear and look up the words later to save waiting time.