MULTI-THREADED DICTIONARY SERVER

Assignment 1

Josh Feng

1266669

APRIL 2, 2024

UNIVERSITY OF MELBOURNE

COMP90015 - Distributed System

1. Problem Context

In this project, a multi-threaded server system including server-side and client-side was designed and implemented. The system runs multiple users connecting to the one server and the users can send requests to the server and get replies accordingly.

The purpose of this system is to create a dictionary server where:

a. User can query the word to server and get the meanings of the word.
b. User can add a word and its meanings to the server.
c. User can update the meaning of the existing word to the server.
d. User can delete the existing word and its meaning to the server.

The system establishes an information exchange protocol between the server side and the user side through the TCP protocol, and the server can handle concurrent requests and handle them in appropriate manner. The server side and the user side also handle appropriate errors. The server also handles illegal request from the user.

2. System Components

2.1 Server

The server uses a thread-per-request architecture, where the server opens a thread for each request sent by the user and sends the result to the user once the request has been processed, and finally the thread that contact with user will be closed. The server will read a JSON file as dictionary data, and server runs multiple users sending requests at the same time, since all the correct requests are to the only one dictionary to call or modify the data accordingly, so the dictionary data is critical region, the server will handle the synchronisation issue appropriately to avoid ambiguity in the data that may prevent the user from getting better feedback. The server will respond according to the corresponding request has responded to the possible errors in the request, after each modification of the dictionary data, the server will be saved to the original file address. In addition, the server provides a GUI for record server information, operation dialog and a visualisation of the dictionary data content.

2.2 Client

The user will use the GUI throughout the entire process to complete all operations, and all operational feedback and potential errors such as failing to establish a connection with the server will be reflected in the GUI. For the user, they can request Query, Add, Delete and Update operations, each of which establishes a TCP connection to the server and closes the connection after receiving feedback from the server. The client doesn’t have to keep connecting to the server all the time, this makes the server consume resource allocations. The GUI is based on SWING and provides users with a simple, quick to get started and lightweight interface, and provides feedback to the user for each operation.

2.3 Environments

SDK version must be (at least): 21.0.2.

Dependency:

groupId: com.fasterxml.jackson.core

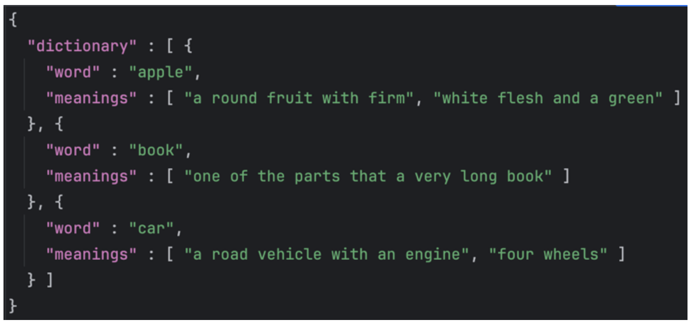
artifactId: Jackson-databind

version: 2.16.1

3. Design Details

3.1 Dictionary Data

The data of dictionary stored as a file in JSON (Figure 1). The outside key is “dictionary”, it is use for identifying this is a dictionary data file, it contains a group, for each group member it must have keys of “word” and “meanings”, they cannot be empty. The “word” is used to identify what the word is, and the “meanings” is a group to store what it means.



```
{
  "dictionary" : [ {
    "word" : "apple",
    "meanings" : [ "a round fruit with firm", "white flesh and a green" ]
  }, {
    "word" : "book",
    "meanings" : [ "one of the parts that a very long book" ]
  }, {
    "word" : "car",
    "meanings" : [ "a road vehicle with an engine", "four wheels" ]
  } ]
}
```

Figure 1: dictionary data structure example

Data is parsed by using Jackson in the Java library, Jackson provides a rich API for reading and writing data and provides getter and setter methods to access object properties. WordFormat.class is created to represent the words in the dictionary and its meanings, and DictionaryFormat.class is created to represent the entire dictionary. The server will interact with the dictionary data using the methods in Dictionary.class, which relies on DictionaryFormat.class to manage the dictionary data, and by accessing WordFormat.class to get or add new words and meanings (Figure 2).

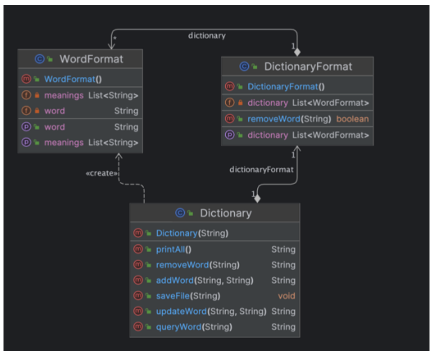


Figure 2: Dictionary Management Class Diagram

3.2 Client

Figure 3 is the diagram of classes in the Client side.

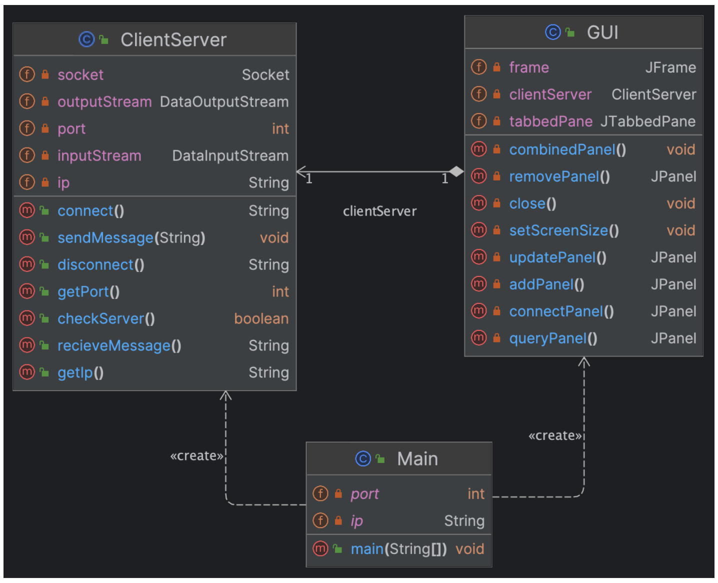


Figure 3: Client Class Diagram

This specification must be used to run the client in terminal:

➢ java -jar DictionaryClient.jar <server-address> <server-port>

Once client start running, first check that the parameter format has been provided and is correctly formatted. The client does not connect to the server automatically after running, it only tries to connect to the server when the user takes Query, Add, Delete or Update operations. Once the user performs the operation, the client will first check if the input information is complete, then it will establish a TCP connection with the server, if it is unable to connect it will pop up a message window to inform the user that it is unable to connect, the reason for not being able to connect may be that the server is not started, or the server-address or server-port is incorrect. Once connected to the server, the client will send three or two separate messages:

1. Command: tell the server what the request is

2. Word: A word provided by user
3. Meanings: meanings of word provided by user (only for Add and Update request)

The client GUI provides four panels, which corresponds to Query, Add, Delete and Update, users can click on the tab to switch (Figure 4).

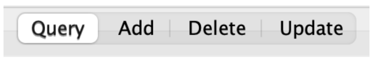
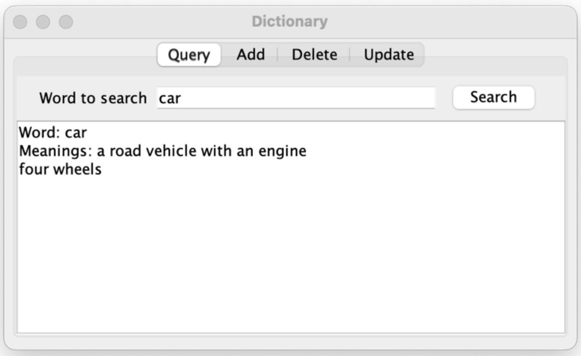


Figure 4: Client's User tab

Figure 5 is a Query panel, user can type in the word they want to search for and then click on the button to search for it, with the feedback will be display in the text area below.

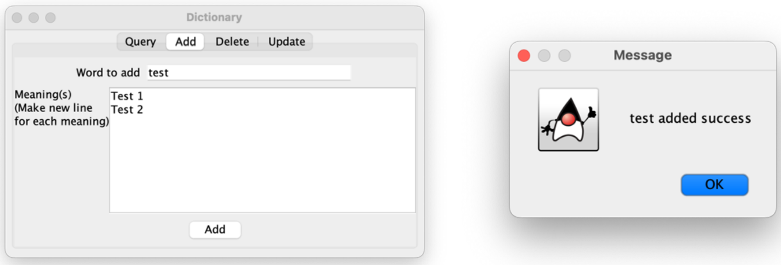


car

```
Word: car
Meanings: a road vehicle with an engine
four wheels
```

Figure 5: Client's Query Panel

Figure 6 is Add Panel, user need to type in the word and its meanings (each line for each meaning). The feedback will pop up a message window.



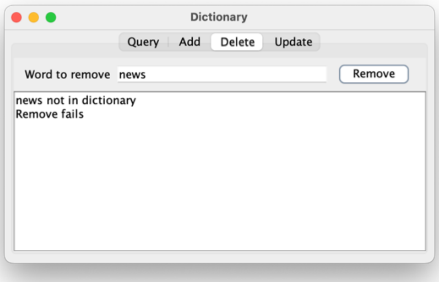
test

Test 1
Test 2

test added success

Figure 6: Client's Add Panel

Figure 7 is Delete Panel, user need to type in the word they want to delete. The feedback from server will be showing in the text area below.

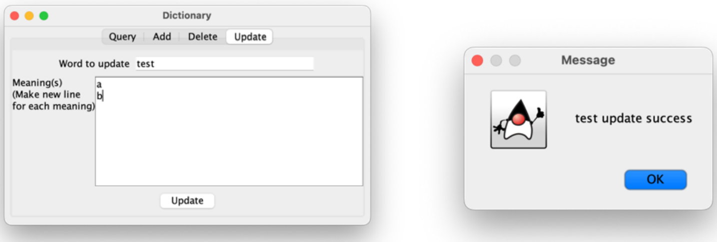


news

```
news not in dictionary
Remove fails
```

Figure 7: Client's Delete Panel

Figure 8 is Update Panel, user can update the meanings of word by type in the word and meanings, the exist meanings won't be added. The feedback will pop up a message window.



test

```
a
b
```

test update success

Figure 8: Client's Update Panel

3.3 Server

Figure 8 is the diagram of classes in the Server side.

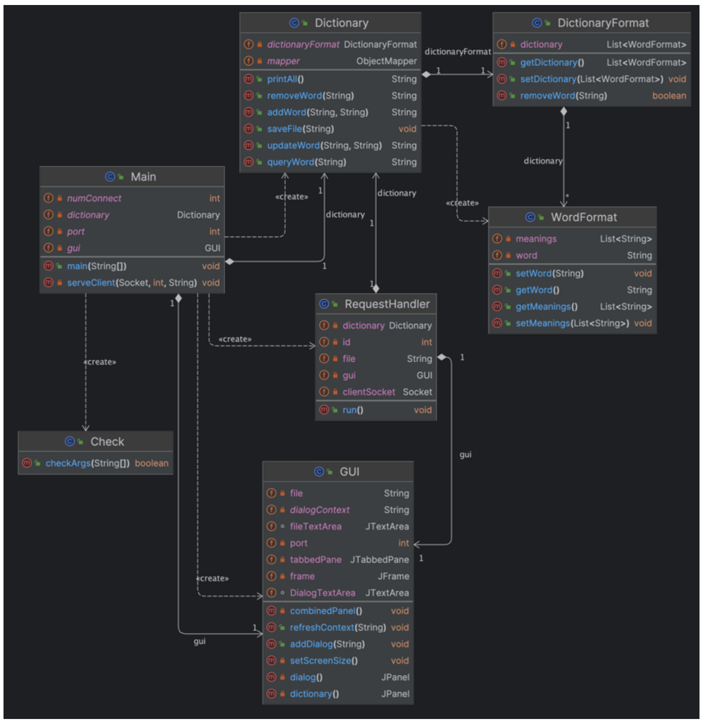


Figure 8: Server Class Diagram

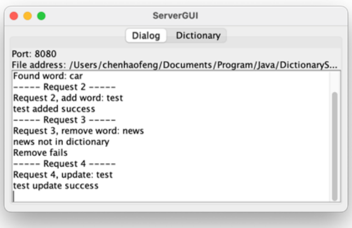
This specification must be used to run the client in terminal:

- ➢ java -jar DictionaryServer.jar <port> <dictionary-file>

Server will first check whether the parameter format is provided or not, if the provided port number is bound it will prompt an error that the server will not start, and if the provided file path cannot be found it will not start the server either.

Once the server receives a connection request from a client, the server creates a thread that is dedicated to this one client's request, the main thread of the server continues to listen for any other user-side connection requests.

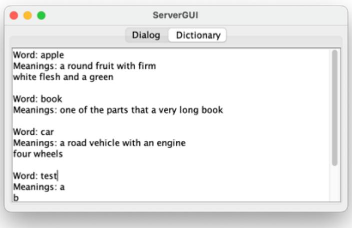
The server also provides a GUI and it have two tabs. Figure 8 shows the dialog panel of the server, which is used to display the current basic information of this server, the port number and the file address, and the text area below is used to print out all the operation requests.



```
Port: 8080
File address: /Users/chenhaofeng/Documents/Program/Java/DictionaryS...
Found word: car
----- Request 2 -----
Request 2, add word: test
test added success
----- Request 3 -----
Request 3, remove word: news
news not in dictionary
Remove fails
----- Request 4 -----
Request 4, update: test
test update success
```

Figure 8: Server's Dialog Panel

Figure 9 shows the dictionary panel, which is used to display all the data for the dictionary, it will be refreshing after any user makes changes.



```
Word: apple
Meanings: a round fruit with firm
white flesh and a green

Word: book
Meanings: one of the parts that a very long book

Word: car
Meanings: a road vehicle with an engine
four wheels

Word: test
Meanings: a
b
```

Figure 9: Server's Dictionary Panel

3.4 Interaction diagram

Figure 10 shows an interaction diagram, demonstrates a user-server interaction.

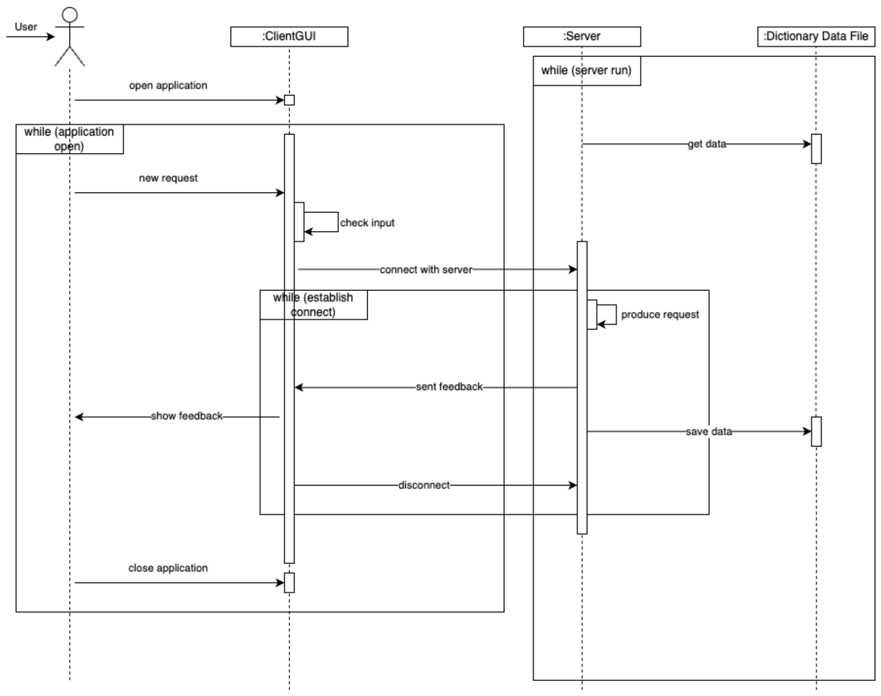


Figure 10: Interaction Diagram

4. Critical Analysis

4.1 Advantage and Disadvantage

The server is based on thread-per-request. This allows each request to operate in a separate thread, so that a request that fails in one thread does not affect other requests. Compared to thread-per-connection, thread-per-connection makes the user to establish connection with the server all the time, and the use may make a request only once in a long time, which leads to waste of server's resources. In addition, the server side and the client side are completely independent (Figure 3 and Figure 8), only need to deal with the communication logic between the two ends, independent of the two ends can improve the scalability of the two ends, in the expansion of the two ends will not affect each other's functions.

There is also problem with using thread-per-request, if there are large number of users requesting to the server during the same period, this can lead the server creating large number of threads which will make resource consumption, and when there are large number of threads, it can be a performance degradation due to the overhead of context switching. In the future server extensions, thread pooling can be introduced to address this potential problem.

5. Excellence

1. This project considers all potential errors and how they are handled, including user actions and connections.

Description	Solution
No corresponding parameter is set (both)	Shows message and shows example, stop running
Port number already bind (server)	Shows message and ask for change, stop running
Try to post request to server but server not started (client)	Pop up a window message, tell user connection fail
Invalid (empty) input (client)	GUI blocks connect, tell user invalid input
Request to query a non-exist word (server)	Send failed result to client
Request to add a exist word (server)	Send failed result to client
Request to delete a non-exist word (server)	Send failed result to client
Request to update a non-exist word (server)	Send failed result to client

Table 1: Error Handling

2. The project designs a whole interactive system on the user side, user can get real-time visual and textual feedback, through the feedback mechanism can improve the user's operating experience. And design language is simple, intuitive, and easy to use (Described in Section 3.2 Client).

3. The report contains class diagrams and interaction diagrams that describe the components of the system and the interactions (Figure 3, Figure 8, and Figure 10). The advantages and disadvantages of this design were also mentioned (Section 4.1).

6. Creativity

1. Created Server GUI for administrators to monitor server basic information including real-time dialog and real-time dictionary data content (Figure 8 and Figure 9).

2. Standardises the dictionary data standard, i.e., the format of the JSON file and using Jackson parser (described in section 3.1). It makes data consistency, which simplifies the server's data handling and operational logic processes. Also brings scalability and facilitates the addition of new attributes to the dictionary data for the future.