COMP90015: Distributed Systems – Assignment 1 Multi-threaded Dictionary Server

Yingying Guo 1382000

yinguo3@student.unimelb.edu.au

1.Problem context

As in the period of data communication and information sharing, how to request information in an efficient and reliable way is important. The multi-threaded Dictionary Server is a component that helps users quickly query the meaning(s) of a word, and add or delete the word to manage the dictionary. What's more, as the development of the user requirements and balance loads of server, there is a need to make a better solution.

In this project, the aim is to design and implement a multi-threaded Dictionary Server, which can handle multiple requests from clients. This project is a strong and quick response system with ensures the operations on the dictionary but also keeps the integrity of data and reduces the resource abusing.

2.Brief description of system

The system consists of client and server, which is required to run the server to listen to the connection requests before clients run to send the connection requests to establish connection and communication later.

2.1 Configuration

- Download the zip with the name of the jar files, and unzip it, all supporting files to run the system already in the package code.
- Then right-click the package code to build two terminals in this directory.

2.2 Execute the system

- **Execute the Server**

```
java -jar DictionaryServer.jar <server-ipaddress> <server-port>
```

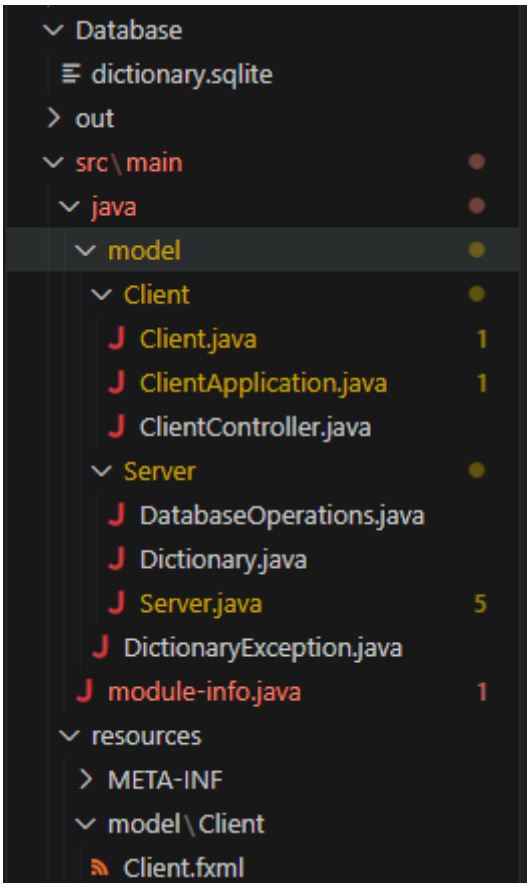
- **Execute the Clients**

```
java --module-path "javafx-sdk-20.0.2/lib" --add-modules
javafx.controls,javafx.fxml -jar DictionaryClient.jar <server-
ipaddress> <server-port>
```

<server-ipaddress>: the IP address for server waited to be connect
<server-port>: the port for connecting to server

3.Class design

TCP is used in this system to ensure reliable communication between clients and server. In addition, all possible errors have been handled in the system, such as Command line parameter errors, network communication errors, I/O and database errors. As for the concurrent requests, are handled by thread-per-connection in the system to ensure scalability. In this system, the classes structure as below:



Figure[1]: File Structure

- Client
  - Client
  - ClientApplication
  - ClientController
- Server
  - DatabaseOperations
  - Dictionary
  - Server
  - ClientHandler
  - Main
- DictionaryException

3.1 Client

Figure[2]: Client UI

If we run the DictionaryClient.jar file, then would first show the UI pane as shown in Figure [2], and capture the inputs of words and meanings from users and corresponding click events to generate request messages and send to server to handle. The result would show diverse information, including whether the operations succeed in the server and notice users give some valid inputs.

The UI loading is implemented in the class ClientApplication by calling the Client.fxml file. class ClientApplication also initiates the socket connection.

Class ClientController just focuses on monitoring and listening to the click event and implementing the corresponding requests and sending to the server.

The last class Client implements the connection between clients and server.

3.2 Server

Class DatabaseOperations encapsulated with direct operation with the database, such as connection to the database, and CRUD operations.

Class Dictionary encapsulated the methods by calling the operations of the database and throwing the errors, which can generate responses to Client requests in the class ClientHandler.

Class Server focuses on connecting with clients in the TCP stream.

Class ClientHandler is a class that extends class Thread, which handles all the requests from clients and sends the responses.

Class Main accepts the IP address and port to initiate the Server object to wait for connection.

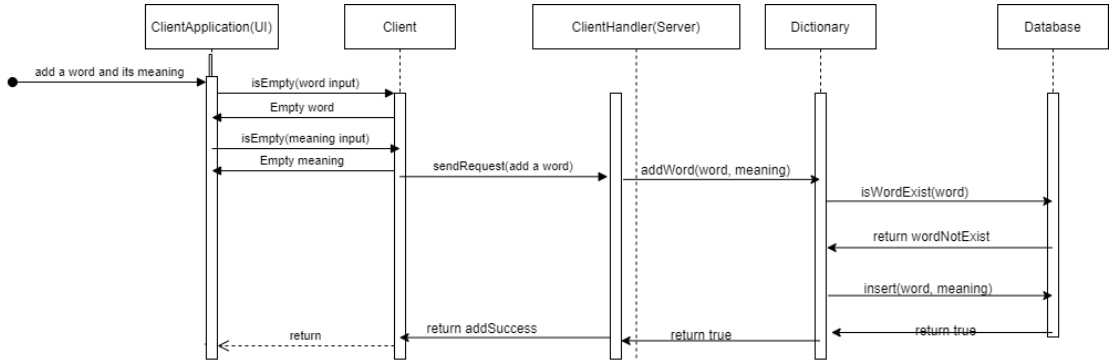
3.3 Exceptions(including DictionaryException)

- IOException, SocketException: Throwing the problems from communication between client(s) and server
- SQLException: Throwing the problems from database connection and CRUD operations
- DictionaryException: Custom Exceptions; Throwing the problems from non-syntax errors, like the word cannot be empty.

Error	**Respond**
Null input in word	Please input a word!
Null input in meanings	Please input meaning(s)!
The word needs to be queried or deleted is not existed in dictionary	wordNotExist
The word needs to be added is already existed in dictionary	wordExist
Multi thread read and write same data at the same time	SQLException database is blocked
Database connect fail	Database connect fail
Delete at database fail	deleteFail
Add at database fail	addFail
Query at database fail	QueryFail
Socket establish fail	Connection fail!
Connection between Client and Server close fail	socket close fail
Infeasible port transfer to run the Client or Server	Invalid input
Infeasible ip address transfer to run the Client or Server	Invalid input
UI loading fail	IOException
Client sends request fail	sendMessage failed!

Table[1] Error and its responding notices

4.Interaction diagram



Figure[3]: Interaction diagram of adding a new word

Message communication format:

```
[action]@[word]@[meanigns]
```

[action]: Operations for “add”, “delete” and “query”.

[word]: Word that need to be handle.

[meanings]: Only when action is add would be used, the meanings of word clients want to add.

5.Critical analysis

5.1 Protocols

Considering the importance of data integrity and reliable communication, TCP is absolutely a better way to implement a system compared to UDP, which ensures both server and clients accept accurate and complete requests and responses.

5.2 The analysis of the work done

5.2.1 Need to improvements

1. Server interface

A server interface has not been developed to help monitor and manage the server operations.

2. Thread Pool Management

In the thread-per-connection model, a new thread is created for each incoming client connection. Each thread is responsible for handling communication with a particular client without interfering with other clients, which has the advantage of simplicity. However, when dealing with a large number of clients, each thread created for each connection consumes memory and other system resources, then it becomes resource-intensive and less scalable. On the other sides, the thread pool model can reduce resource abuse by maintaining a pool of pre-allocated threads rather than creating new threads for each connection. When clients connect to the server, they are assigned to the available threads in the pool. Once the client's request is processed, the threads are returned to the pool for reuse.

5.2.2 Excellence and Creativity

1.Connect to database

All dictionary data is stored in the database, which means that there is no need to read the whole database every time when accepting requests from clients. Then we can save space and storage resources to improve efficiency.

2.Making a limitation on each request that every time a request is processed by the current thread to just one, preventing simultaneous reading of data, which can lead to read/write problems.

3.Reasonable system design, through the class encapsulation method to ensure that the code neatness and low-coupling.

4. The design of the thread improves scalability.

6. Conclusion

In a nutshell, implementing TCP ensures the connection of communication and thread-per-connection ensures the concurrency running of the system. In addition, scalable, error handling, and data consistency are excellent points in this project. However, there are also some improvements that can be made in the thread resources management and concurrency access. In conclusion, this is a good chance to practice the theory of thread and socket, enhancing the understanding of these theories.