COMP90015: Distributed Systems

Assignment 1 - Report

Shuangkun Fan

Student ID: 1131667

1. Problem context

In this assignment we were required to design and implement a multi-threaded dictionary server using a client-server architecture, where multiple clients could concurrently query the meaning of words, update and delete and add a word, and sockets and threads must be used explicitly. For the implementation of this project I used the Transmission control protocol (TCP) socket, because the connection and data transfer between client and server is more reliable. For the server, the project uses the thread-per-connection mechanism to implement a multi-threaded server. By using thread-per-connection, speared threads for each connected clients are created, the server is able to quickly response to client requests without waiting for other requests to complete, it guarantees user could get fast response when querying a word. In addition, A separate thread for each connection prevents the user side from interfering with other users' requests, ensuring server stability and responsiveness in the event of multiple users sending requests.

1.1 Saving Dictionary

This project stores all the words added by the user in a text file called "dictionary", which is more compact than json when storing simple key-value- paired values. If the program crashes, the user can open the text file and see it, which is more readable than json, and it is simple and widely supported. It is also useful when a user wants to share their own glossary list.

1.2 Error Handling

Both client and server error handling are controlled and descriptive error messages are provided to inform the user of what they have done wrong. For example, user input errors, client and server connections.

2. System Components

There are two main components of this system, the server package and the client package. Server package have three java class which are DictionaryServer, ServerThreadConnection, and ServerUI. Client package have two java class which are DictionaryClient and ClientUI.

3. Class design & Interaction diagram

- 3.1 Use case Diagram

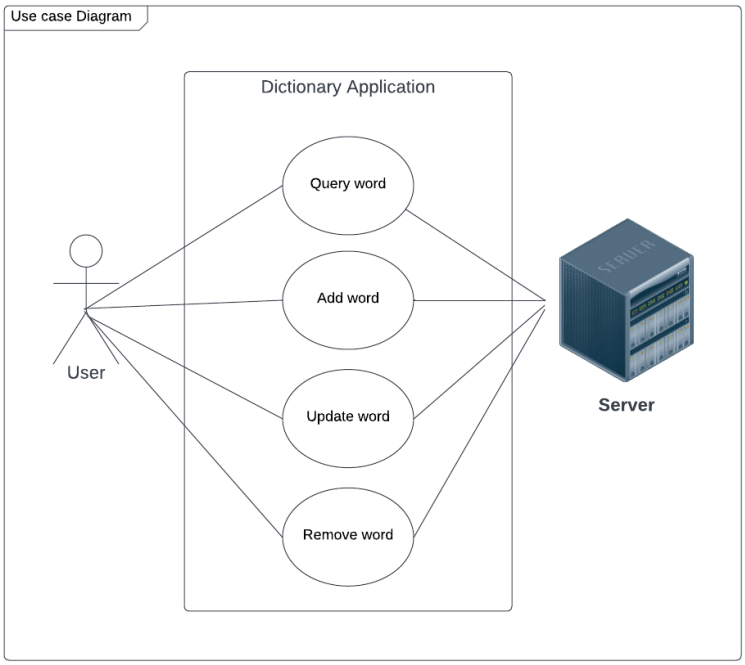


Figure 1: Application Use Case Diagram

The system *Dictionary Application* has the *User* and *Server*. The user has four use cases with the server which are "Query word", "Add word", “Update word” and "Remove word".

3.2 UML diagram

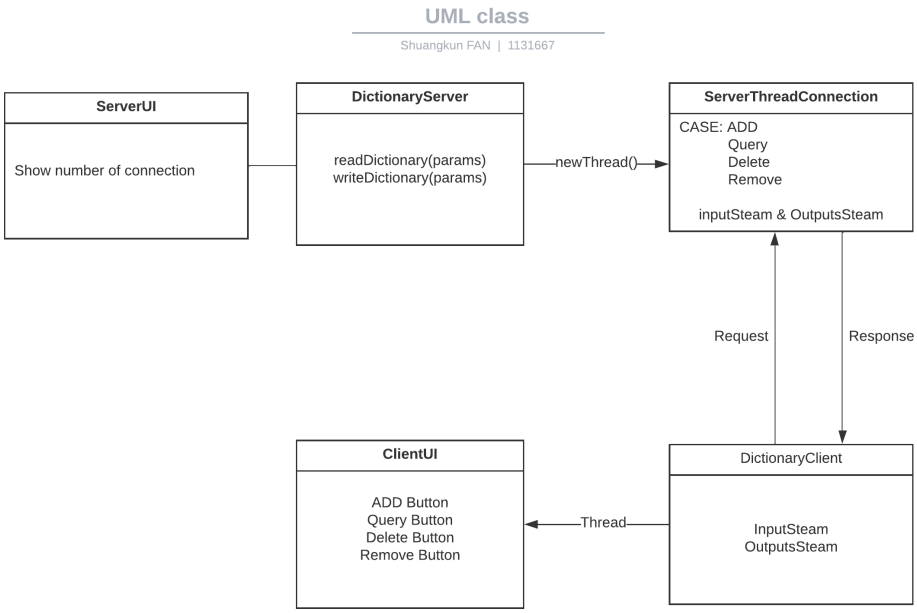


Figure 2: UML class diagram

As mentioned above, this project has a total of five java class files, they are clientUI, DictionaryClient, DictionaryServer, ServerThreadConnection, and ServerUI. The role of the DictionaryServer is to open the server socket to establish a server given a port number by user, and handle requests from the client, and read and write the dictonay.txt file into ConcurrentHashMap which allows a sequence of operation to be executed.

Once the server is up and running, serverThreadConnection creats a thread connection for every client, and creates I/O steams to communication with client, like handling different requests, such as querying or updating a word.

DictionaryClient and ClientUI are, as their names imply, used to creates socket object to build the connection to the IP address and port number given by DictionaryServer and to build a thread to create ClientUI. When client is connected to server, there is also a ServerUI to display how many connections (Threads) there are from the client.

4. Analysis

In summary, all the functionality required for the project is implemented, all errors are handled appropriately, and the requirements of the project specification are met. However, I think that there are some areas that could be optimised. Firstly, the project is implemented using a thread-per- connection, which is still connected to the server when the client and server are not interacting and takes up the server's resources, and the thread only ends when it exits. This may cause a load on the server, and I think it might be possible to try sleep the current thread when the user is not interacting with the server, thus reducing the server's load. In addition, the user interface of the server only shows the number of connections and does not have many functions. I think it would be worth considering the possibility of adding a button to shut down the server or to disconnect a specific connection.

-
-

5. Excellence

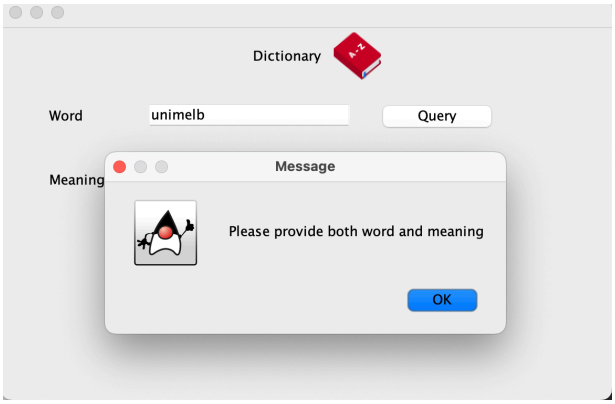
- Error message will show if a port number given by user is already in use to provide message tells user to change anther port.

```
} catch (BindException e) {
    System.out.println("Port number address already in use, Pick another one");
```

- Server side's error are handled on server side. eg. If port number is wrong or server is not ready yet then error message dialog will show informative error message,

```
} catch (ConnectException ce) {
    JOptionPane.showMessageDialog( parentComponent: null,  message: "Wrong port number or server connection error");
} catch (UnknownHostException e){
    JOptionPane.showMessageDialog( parentComponent: null,  message: "UnknownHostException Error");
} catch (IOException e) {
    System.out.println("IOException error");
}
```

- Client side's error like user input error is all handled on client side which reduce the workload of server. Eg. a user trying update a word must enter both 'word' and 'meaning', if one of them are missing, a window will be displayed to show error message.



- When server is accidently terminated or closed, a pop-up window will let notify user that server is closed, it gives user sense of in control, user will not keep using a "useless client".

```
//what about client if server closed?
JOptionPane.showMessageDialog( parentComponent: null,  message: "lost connection from the server");
if(client != null){
    client.close();
}
clientUI.interrupt();
return;
```

6. Creativity

a) Create a Server-Side UI showing the number of connections.
b) Server-side showing the IP address