COMP90015 Distributed Systems

Assignment 1 Report

Student name: Wendong Chen

Student ID: 931018

Username: wendongc1

Tutor: Lakshmi Jagathamma Mohan

1. Project overview

This assignment requires students to write an online dictionary Java program using client-server architecture. In addition to applying Java socket and multithreading techniques, the design of program is open. We can implement a various of functions according to our own ideas, including GUI design, error handling, HCI, etc.

The structure of my program is a separately running client and server model. Once running the server model with the input port number and dictionary file path, the server will listen to this port. Every time receiving a connection request from a client by means of socket binding, the server will create a thread for this connection, i.e., thread per connection. Interaction between the server and a client is processing within the thread which the server created for this client.

2. System components

2.1 Server

The design and implementation details of server are as follows:

There are two java class files in "DictionaryServer.jar", which are "DictionaryServer.class" and "Job.class".

In "DictionaryServer.class" has the program entry ("main" method), it can be started by typing "java -jar DictionaryServer.jar <port> <dictionary-file>" in cmd (or Terminal). The server will bind an assigned port number through the first parameter <port> passed in and listen to connection requests. The second parameter passed in is the dictionary file path. During the interaction with clients, the server will do IO operations to the dictionary file on this path. Whenever receives a connection request, i.e., the socket is successfully bound, the server will create a thread for this connection and pass job (the second class "Job.class") to it. In addition, I designed a GUI for server-side. Whenever clients connect, send requests(query, addition, deletion), the messages will be showed on the GUI.

A Job object is claimed with two parameters (the bound socket and dictionary path) in "DictionaryServer.class". Job inherits "Runnable" interface and it's passed into the thread to handle client's requests. The code fragment is as follows:

```
Job job = new Job(socket, path);

Thread t = new Thread(job);
```

This implements a thread-per-connection architecture., i.e., one thread serves a client. There are three kinds of client requests which are query, addition and deletion and these are defined in Job, respectively.

- Query request

At first the server reads the dictionary file and translates its content into an ArrayList. Then, check whether the query word exists in the ArrayList. If it does not exist, returns a message "Word does not exist!" to client. Otherwise, the server will return meaning(s)

of the word.

- Addition request

At first the server reads the dictionary file and translates its content into an ArrayList. Then, check whether the query word exists in the ArrayList. If it does, the server will return “Word already exists!” Otherwise, sends a “ok” message to the client. If the client receives a “Word already exists” message, the adding operation will be prohibited. But if receives a “ok” message, the server can continue to edit one or more meanings of the new word. After edition completes, the server will receive the meaning(s), write it into the dictionary file and send a “Addition success!” message to the client.

- Deletion request

At first the server reads the dictionary file and translates its content into an ArrayList. Then, check whether the query word exists in the ArrayList. If it does not exist, returns a message “Word does not exist!” to client. Otherwise, the server will delete the word and its meaning(s), update the dictionary file, and send a “Deletion success!” message to the client.

2.2. Client

The design and implementation details of client are as follows:

There are five java class files in “DictionaryClient.jar”, which are “DictionaryClient.class”, “ClientGui.class”, “Query.class”, “Addition.class”, and “Deletion.class”.

The entry for client program is in “DictionaryClient.class” (“main” method), and it can be started by typing “java -jar DictionaryClient.jar <server-address> <server-port>”. The two parameters are the server IP address and the port number listening by the server. After inputs the correct parameters, the connection is established successfully. Then, a ClientGui object will be claimed and the bound socket will be passed to it.

By calling the go() method in “ClientGui.class”, a GUI interface will be generated with three functional buttons which are “Query”, “Addition” and “Deletion”, respectively. Clicking the “Query” button will generate an object of “Query.class” and pop up a new window for query words. Similarly, clicking the other two buttons will generate objects corresponding to them and popping up new windows with different functions.

In “Query.class” defines the details of the query window. User types the word to query in the textField. After clicking the “Query” botton, the server will return the meaning(s) of the word or a “Word does not exist!” message in the textArea.

In “Addition.class” defines the details of the adding window. User types the word for insertion in the textField. After clicking the “Enter” button, the server will return whether the word exists in the dictionary. If it already exists, operation stop. But if it does not exist, a new windows will pop up and you can continue to add meaning(s) of the word.

In “Deletion.class” difines the details of the deletion window. User types the word to delete in the textField. After clicking the “Deletion” button, the server will return a “Deletion success!” or “Word does not exist!” message.

3. Error handling

3.1 Server

Server-side exception handling details are as follows:

- Input from the console for what concerns the parameters passed as command line.

a) Incorrect parameter counts (0 or 1 parameter) or in wrong sequence (not first <port> and then <dictionary-file>).

b) The port number (<port>) is not an integer, such as “123k”.

c) The port number exceeds the maximum value (higher than 65535).

If errors above happen, a window will pop up with a “Please enter correct port number and dictionary path ^_^” message.

- The dictionary file is in “.txt” format, but the path (<dictionary-file>) is not end with “XXX.txt”.

A new window will pop up with a “The dictionary file should be a “.txt” file” message.

- Incorrect port number

The port is already in use. A window will pop up with a “Address already in use. Please change another one.” message.

- Cannot find the dictionary file

If the dictionary file does not exist. A new window will pop up with a “Cannot find the dictionary file!” warning and the server program will continue to run. As for the client program, it can run normally. The dictionary file will be generated automatically until a client perform an “Addition” operation (add a word). After that, the server will do IO operation to this dictionary file without warning.

- Client disconnection

The server-side GUI will show that “XXX disconnected.”

- Concurrency problem

Read (write) operations on dictionary file can only run independently, i.e., multiple read (write) operations cannot be performed simultaneously. The solution is to add a “synchronized” modifier to the read (write) method. In this way, when multiple users access the dictionary file simultaneously, requests need to be queued and problem

solved.

3.2 Client

Client-side exception handling details are as follows:

- Input from the console for what concerns the parameters passed as command line

a) The port number is not an integer, such as “123k”.

b) The IP address is not in the correct format, such as “999.999.999.999” or “fgg^”.

If errors above happen, a window will pop up with a “Please enter correct server address and port number ^_^” message.

- Incorrect IP address or port number

A new window will pop up with a “Connection failed. Please enter the correct server address and port number.” message.

- Closed server

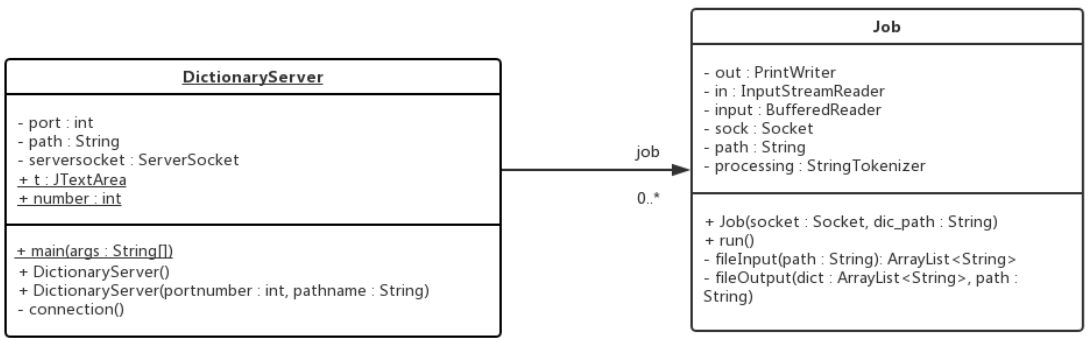
If the server is closed, the GUI of client will not be affected until sending a request (“Query”, “Addition”, “Deletion”) to the closed server. A window will pop up with a “Oops, your connection was interrupted!” message.

- Empty request

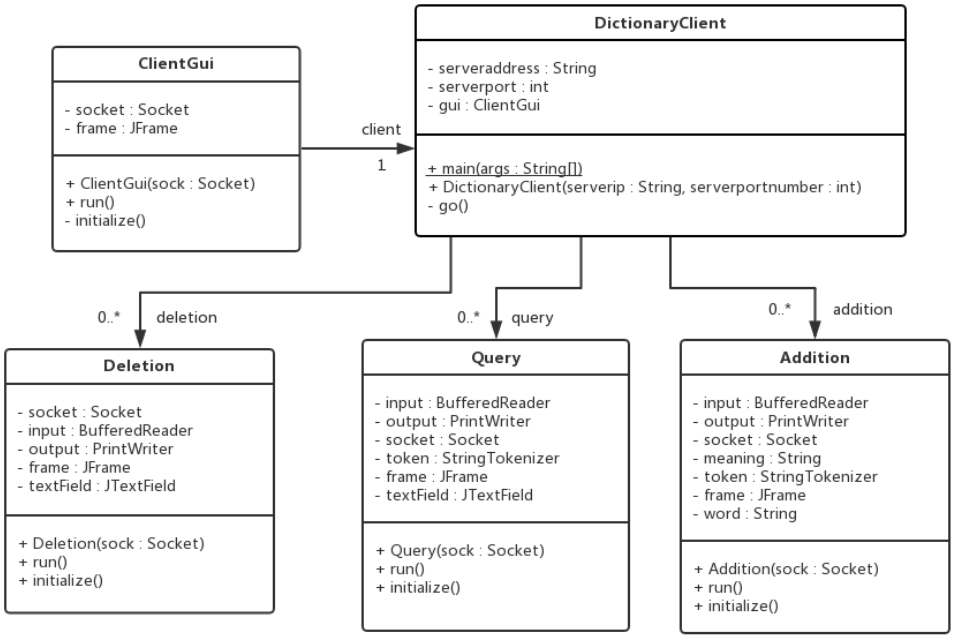
If client wants to query, add, or delete an “empty word”, i.e., “”. A new window will pop up with a “Insert a word.” message. If user enters an empty meaning, a new window will pop up with a “Meaning is empty!” message. If the user does not add any meanings for a new word, a new window will pop up with a “Addition failed! Add meaning, please.” message and this addition operation fails.

4. UML

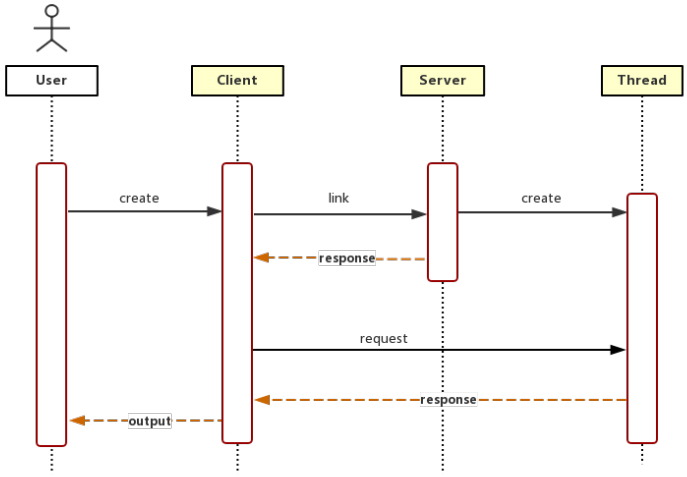
4.1 UML of server



4.2 UML of client



4.3 Sequence diagram



5. Conclusion

In this project designed a multi-thread server that allows multiple users search, insert, or delete words simultaneously based on Java socket and multi-threading techniques. There are a lot of errors and exceptions need to be considered for implementing this client-server model. In the process of solving these problems deepened the understanding of Java and client-server architecture.

Tips:

This program uses two uncommon Chinese characters as separators (StringTokenizer), which are “眚” and “曩”, respectively. Therefore, don’t use these two characters in words and meanings.