

COMP90015: Distributed Systems - Assignment1 Report

Name: Yiming Zhang

Student id: 889262

Email: yimingz8@student.unimelb.edu.au

Tutor: Alisha Aneja

Date: 06 Sep, 2018

1. Introduction

In this report, firstly, the problem description of the multithreaded dictionary server will be discussed. Then, the components of the system and the class design will be explained in the report. Finally, the whole design will be critical analysis.

2. Problem Context

The requirement of the assignment 1 is to build a multithreaded dictionary server, which can allow concurrent clients to search the meaning of the word, add word and remove word.

3. System Components

The system has client-server architecture which uses TCP socket to transform message. In the code, two java files “Client.java” and “DictionaryServer.java” implement the client-server architecture. For the multithreading part, the system uses worker pool architecture to allow concurrent clients can connect to the server. The “MultiThreadServer.java” implements the worker pool architecture. The messages that transformed in the system are JSON data format.

The “JsonDictionary.java” implements the JSON dictionary. Each java file contains “try-catch” to catch exception and will print out the error message. For example, if the clientGUI starts without serverGUI, the error message will print as below in figure 2.

```
// catch error messages and print error messages.
} catch (UnknownHostException e) {
    //e.printStackTrace();
    System.out.println("Error! A host error occurred! ");
} catch (IOException e) {
    //e.printStackTrace();
    System.out.println("Error! A I/O error occurred! ");
} finally {
    // Close the socket
    if (socket != null) {
        try {
            socket.close();
        } catch (IOException e) {
            //e.printStackTrace();
            System.out.println("Error! A socket error occurred! ");
        }
    }
}
```

Information From Server:

Error! A I/O error occurred!

Figure1: “try-catch” error message

Figure2: ClientGUI error message

4. Class Design & Interaction Diagram

“JsonDictionary.java” implements the JSON dictionary, it uses “json-simple-1.1.1.jar” to build JSON object and each word in the dictionary is treated as a JSON object. The reason to choose JSON to store the dictionary is that it is efficient and reliable. There are three words in the default dictionary which are “java”, “python” and “c”.

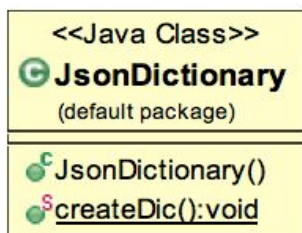


Diagram1: JsonDictionary UML

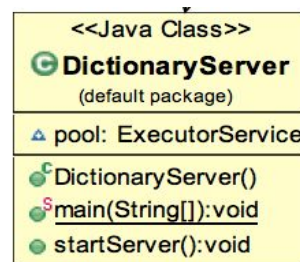


Diagram2: DictionaryServer UML

“DictionaryServer.java” implements the server in TCP server-client architecture. Java uses `ServerSocket` to implements it. In the code, the port number is set to 4444. In addition, it implements the worker pool architecture. Java has “`Executor`” and “`ExecutorService`” methods to help build it.

```
listeningSocket = new ServerSocket(4444);
```

Figure 3: code fragment from DictionaryServer.java

```
ExecutorService pool = null;    pool = Executors.newFixedThreadPool(5);
```

Figure 4: code fragment from DictionaryServer.java

“MultiThreadServer.java” implements the three functional requirements. It has three methods which are `query()`, `add()` and `remove()`. To check the meaning of one word in the dictionary, user should input “query, java” in the **console**. To add one new word to dictionary, user should input “add,html,a language used to build web.” in the **console**. To remove one existing word from the dictionary, user should input “remove,java” in the **console**.

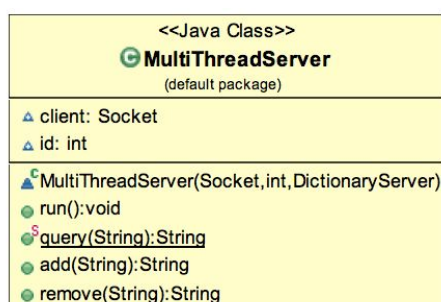
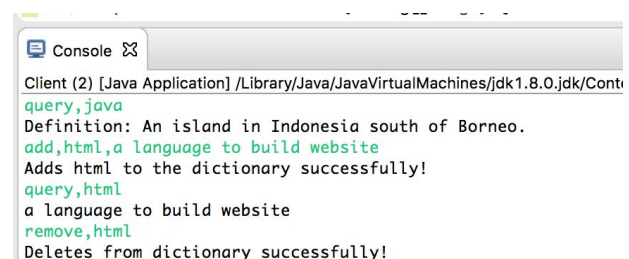


Diagram 3: MultiThreadServer UML



“Client.java” implements the client part of the TCP client-server architecture. Java uses “socket” to implement it. In order to connect to the server, the port number is set to 4444 too. The port number has to be same with the port number in the ServerSocket.

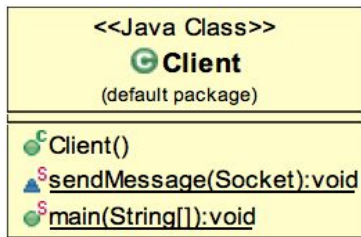


Diagram 4: Client UML

```

socket = new Socket("localhost", 4444);
  
```

Figure 6: Clinet socket

“ServerGUI.java” implements a graphical user interface for the server. It has two buttons which are “Show Client Number” and the other one is “Show Details”. If the user clicks the “Show Client Number” button, it will show how many clients have connected to the server. If the user clicked “Show Details” button, it will show the remote number that the client has just connected. The screenshot of “ServerGUI.java” is shown below at figure 7.

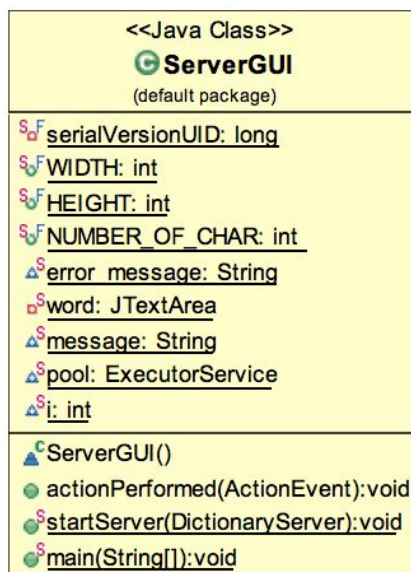


Diagram 5: ServerGUI UML

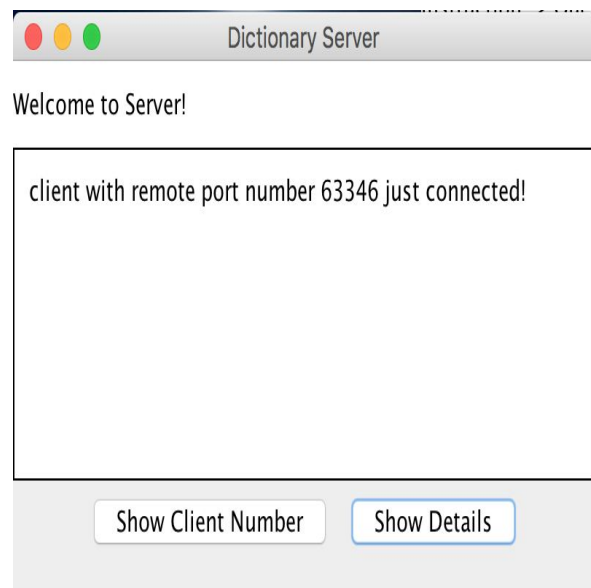


Figure 7: DictionaryServer GUI

“ClientGUI.java” implements a graphical user interface for the client. If the client connects the server successfully, it will show the connect message. The user can query a word or remove a word by just enter the word in the box and then clicks the “query” or “remove” button. However, it has to be noticed that if the user want to add a new word, the new definition has to be followed the word with “,”, such as “html,a language to build website.”, then the user can click the add button and the message from the server will show in the GUI frame.

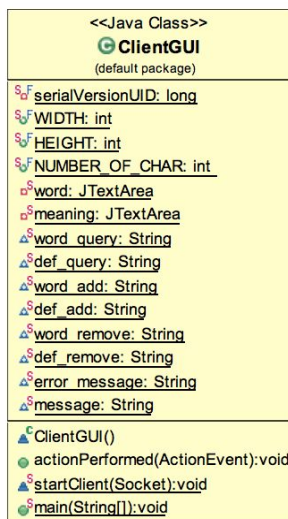


Diagram 6: ClientGUI UML

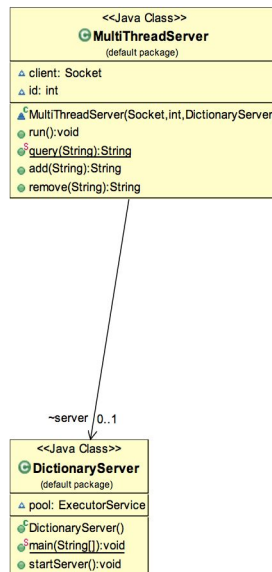


Diagram 7: Client- Server Relationship UML

Information From Server:

Connected to the server successfully!

Figure 8: ClientGUI shows connection information

●
●
●

Dictionary

Welcome to dictionary! Please enter word in the box:)

Instruction-> Query:word Add:word,definition Remove:word

java

Information From Server:

Definition: An island in Indonesia south of Borneo.

Query

Add

Remove

Figure 9: ClientGUI Query a Word Definition

5. Critical Analysis

5.1 Architecture

The system chooses TCP client-server and worker pool architecture to implement the dictionary server. Transmission Control Protocol (TCP) provides an abstraction for a two-way stream, and the data sent by the producer are queued until the consumer is ready to receive them.[1] In a worker pool architecture, the server creates a fixed pool of worker threads to process requests.[2]

5.2 Reason to Choose the Architecture

There are mainly three reasons to choose TCP. Firstly, TCP establishes a connection before communication. Next, it uses an acknowledgment scheme which is more reliable than UDP. Finally, it is easier to implement in Java. The reason to choose worker pool architecture is that it is more practical to apply it in the real world. The server just needs to create a fixed number of threads which are worker pool and when requests arrive at the server, they are put into a queue and from there assigned to the next available worker thread. For instance, if 100000 users request the server at the same time, if we choose the thread-per-request architecture, then the server will have to deal with 100000 requests and create 100000 threads, the server will probably crash. However, if we apply worker pool architecture, the server will handle it well.

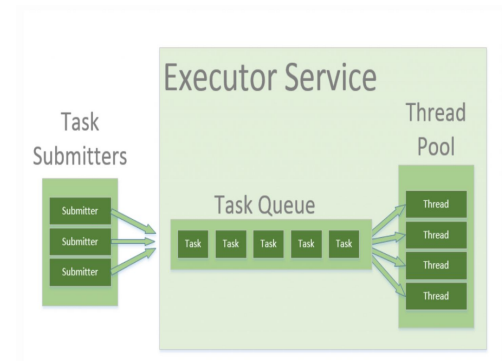


Figure 10: worker pool architecture

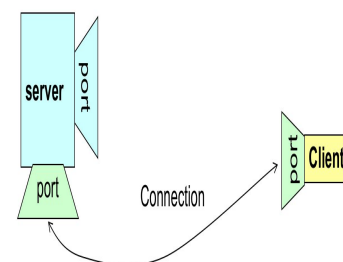


Figure 11: Socket Connection

5.3 Further Improvements

The system is not perfect there are still so many things can be improved if more time is given. For example, the GUI can be better and the serverGUI can have more functions. Also, as the design is very easy and basic, it does not require user to input IP or port number, the port number is set as port 4444 as default and the IP address is localhost(127.0.0.1). If more time is given, the GUI for inputting IP and port number will be implemented.

6. Conclusion

To conclude, the system that we built in the code uses a TCP client-server and worker pool multithreading architecture. We explained the reason why we choose this architecture, as well as how we implement in Java. And some improvements can be done if more time is given.

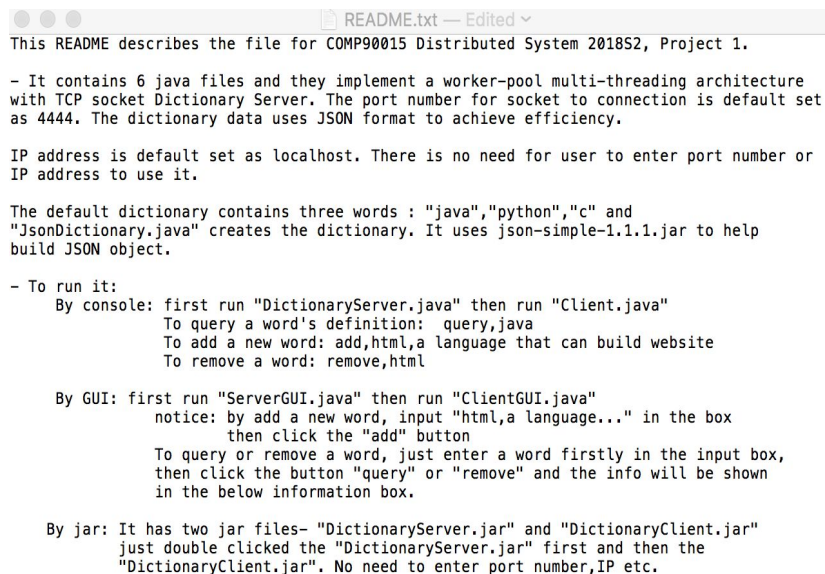
Reference:

[1] R. Buyya, COMP90015. Class Lecture, Topic: "Inter-Process Communication: Network Programming using TCP Java Sockets" Department of Computing and Information System, The University of Melbourne, Melbourne, VIC, July. 31, 2018.

[2] R. Buyya, COMP90015. Class Lecture, Topic: "Multithreaded Programming using Java Threads" Department of Computing and Information System, The University of Melbourne, Melbourne, VIC, Aug. 8, 2018.

Appendix:

readme screenshot:



```
● ● ● README.txt — Edited v
This README describes the file for COMP90015 Distributed System 2018S2, Project 1.

- It contains 6 java files and they implement a worker-pool multi-threading architecture
with TCP socket Dictionary Server. The port number for socket to connection is default set
as 4444. The dictionary data uses JSON format to achieve efficiency.

IP address is default set as localhost. There is no need for user to enter port number or
IP address to use it.

The default dictionary contains three words : "java","python","c" and
"JsonDictionary.java" creates the dictionary. It uses json-simple-1.1.1.jar to help
build JSON object.

- To run it:
  By console: first run "DictionaryServer.java" then run "Client.java"
               To query a word's definition: query,java
               To add a new word: add,html,a language that can build website
               To remove a word: remove,html

  By GUI: first run "ServerGUI.java" then run "ClientGUI.java"
            notice: by add a new word, input "html,a language..." in the box
                   then click the "add" button
            To query or remove a word, just enter a word firstly in the input box,
            then click the button "query" or "remove" and the info will be shown
            in the below information box.

By jar: It has two jar files- "DictionaryServer.jar" and "DictionaryClient.jar"
        just double clicked the "DictionaryServer.jar" first and then the
        "DictionaryClient.jar". No need to enter port number,IP etc.
```