COMP90015 DISTRIBUTED SYSTEMS

Multi-threaded Dictionary Server

Name: JIAQI WANG ID: 908406

1 Introduction

With the power of internet, nowadays users can have access to services anywhere from the world. In this context, a distributed system is superior to a single machine to serve these needs. Distributed systems have advantages such as functional separation, inherent distribution, power imbalance and load variation, reliability and economies.

In this assignment, I will explore the distributed system by designing and implementing a multi-threaded dictionary using Java to allow concurrent clients searching the meanings of a word, adding a new word, and removing an exisiting word.

1.1 Architecture

From the top, the system will follow client-server architecture. A single multi-threaded sever is running and listening for connections from clients. Concurrency in the server is carefully handled by the use of ConcurrentHashMap.

1.2 Interaction

All communication will take place via TCP sockets. In addition, JSON is used to be the message exchange protocol.

1.3 Failure Model

Errors handling in server side includes printing stack-trace messages and assigning default values. In client side it includes showing dialog windows and requiring user to fix the inputs.

2 System Components

The server program, the client program and the client GUI are `Server.java`, `Client.java` and `ClientWindow` correspondingly.

2.1 `server.java`

The server is invoked through the `main()`. The port number is read from `args[0]`. In case when any error occurred, the default port number 3005 will be assigned. The dictionary is a `ConcurrentHashMap` instance, constructed using the provided dictionary JSON file. Then the server tries to instantiate a `ServerSocket` with the port number. In the while loop, the `serverSocket` blocks until a connect is accepted. It passes the client's socket to a new thread for processing all following requests. The loop then repeats listening connection and so on.

The `serveClient()` handles requests from a client. It first prints which client is connect. Then it tries to `readUTF()` from `DataInputStream` in a while loop. A request is a JSON String with internal structure {command, word, meaning}. Once a request message is received, it is deserialized and processed accordingly. The results and errors are put into an `ObjectNode`, which is serialized at the end and written in `DataOutputStream`. When an `EOFException` is thrown in reading input stream, it means a client has closed GUI window, so the server prints a message saying that the client is disconnected.

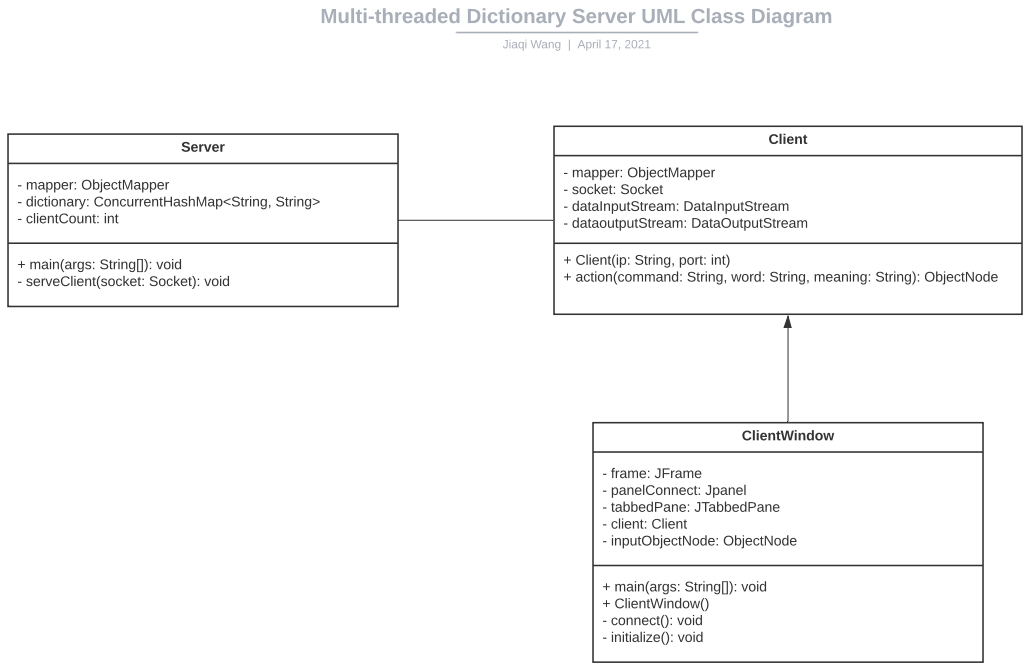
2.2 `Client.java`

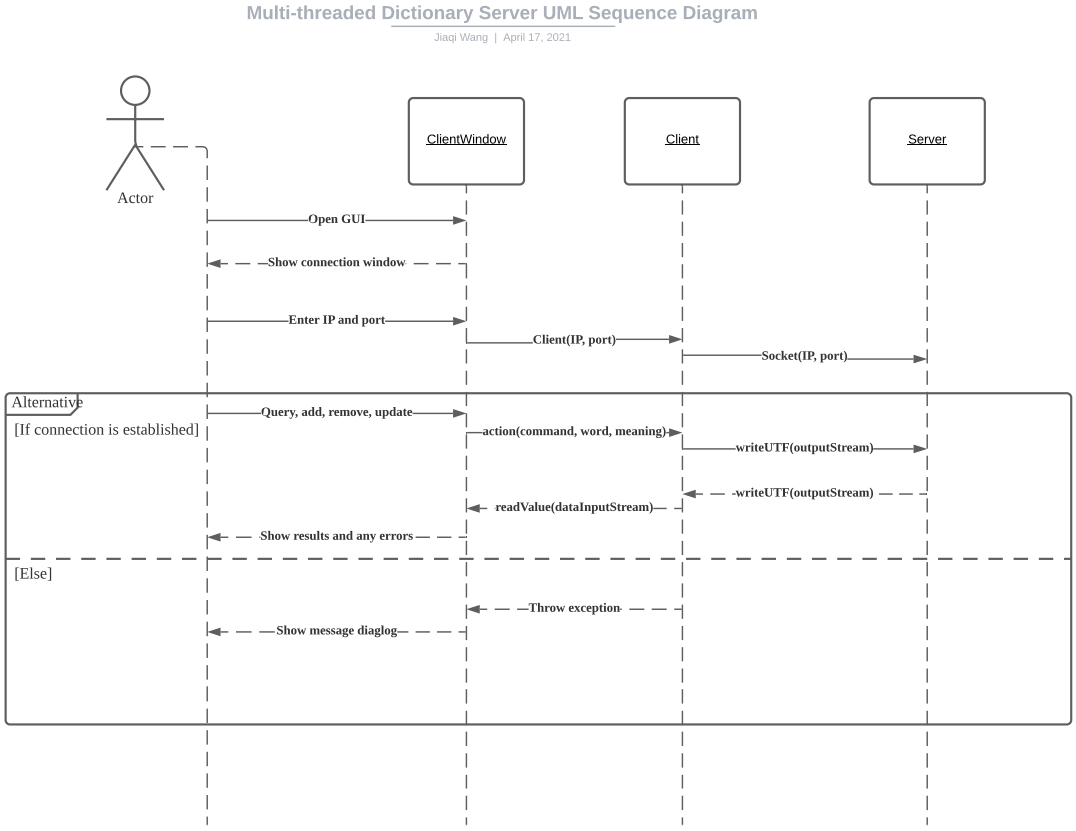
This `Client` class has constructor which takes two arguments ip and port and creates corresponding socket and data streams. The method `action()` is serializing a command sent to the server and returning an deserialized `ObjectNode` result.

2.3 `ClientWindow`

`ClientWindow()` creates a frame and a connection panel is shown by `connect()`. After a user input IP and port for connection, `initialize()` removes the connection panel and adds a tabbed content panel for different functionalities.

3 Class Design





4 Evaluation

The design choices might have following advantages and disadvantages.

- For the thread architecture, a thread-per-connection approach is chosen since it is more intuitively to have one thread serves one client and requests from one client at a time can be easily handled by one thread in this dictionary application. However, the client might need to wait for the thread-creation comparing to worker-pool architecture.
- TCP sockets provide reliable communication since it has taken care of all kinds of network issues in the transport layer. But it's speed is slower than UDP. The message passed between the server and the clients are all tiny, so it doesn't impact much in this dictionary application.
- The message exchange protocol JSON has advantages of being light-weighted and human-readable. But it requires external library Jackson to process, which is not very portable and might brings performance issues.

4.1 Excellence Elements

- Notification of errors are shown in a dialog window, which really help user to understand what went wrong.
- A user can use GUI to enter IP address and port for the server, which makes the GUI more user-friendly. If connection is refused for specific IP and port, an message dialog window will appear. A user checks to see the cause and then re-connects again.

4.2 Creativity Elements

- A tabbed pane is designed in client GUI to seperate different functionalities. It can effectively prevent any operation mistake and make GUI more intuitive.
- In the server, a `ClientCount` is kept. It helps to keep track of how many clients have connected and identify which client is currently connecting and disconnecting.