Name: Ruilin Liu
ID: 871076

COMP90015: Distributed Systems – Assignment 1

Multi-threaded Dictionary Server Report

Name: Ruilin Liu | ID: 871076

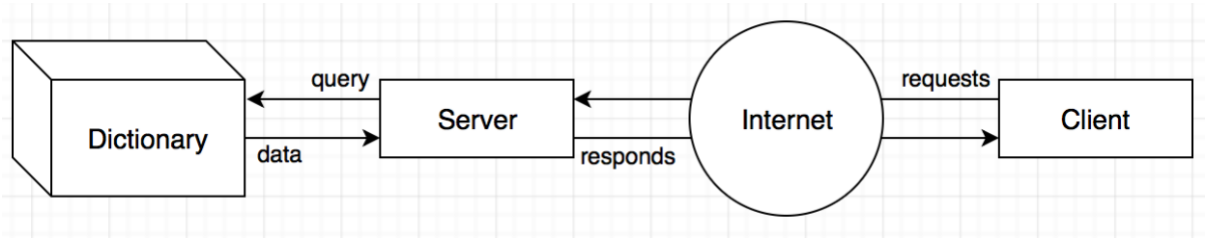
1. Context

This assignment is aimed to build a multi-threaded dictionary server which allows multiple clients to use. The dictionary should support several operations such as search, add and remove. It also requires to handle all failures which may happen and ensure all communication to be reliable.

A

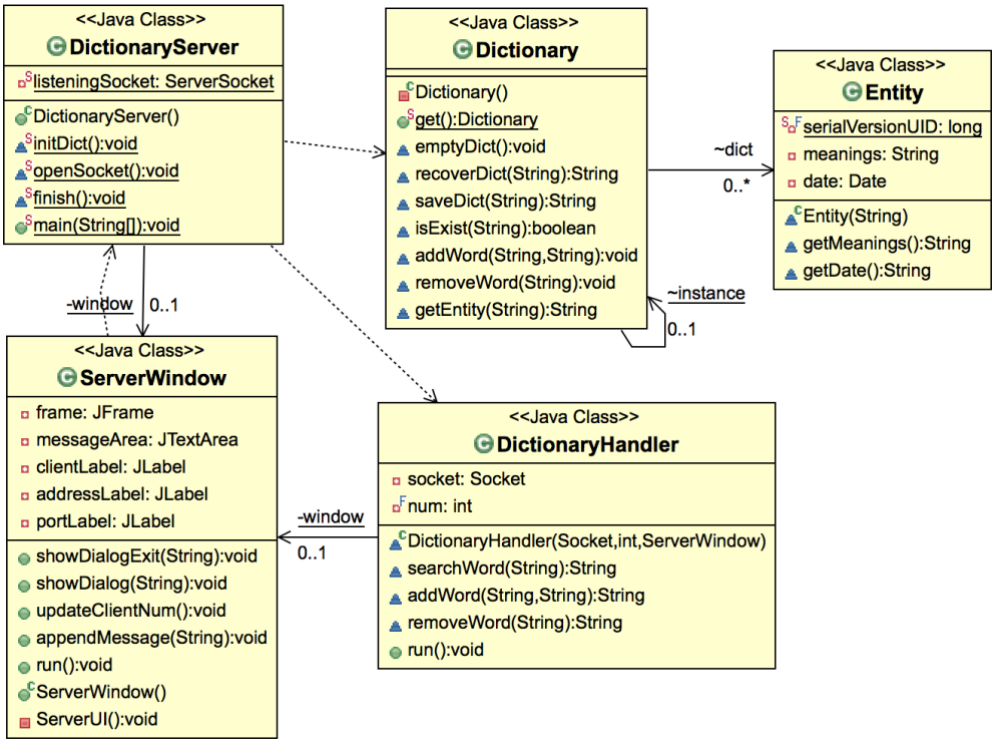
2. System Components

The application comprises three main components: Dictionary, Server and Client. Firstly, the Server opens Server Socket and keeps listening to connection requests. When there is a new Client making a request, the Server accepts the request and creates a thread to build the connection. Then the Client can send search, add or remove requests to the Server. The requests are represented by strings. After the Server received the requests, it queries relevant data from Dictionary, and sent it to the Client. Finally, when the client wants to disconnect, the Server removes the thread, which is thread-per-connection architecture.



2. Class Design

(1) Server and Dictionary:



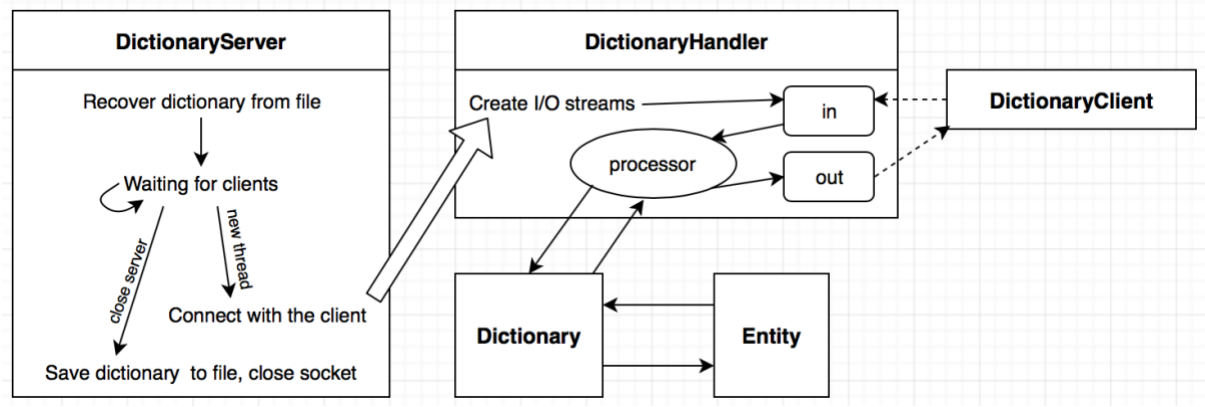
Name: Ruilin Liu
ID: 871076

The functions of **DictionaryServer** class are recovering the dictionary from the file (by call recoverDict method in Dictionary), opening the Server Socket, creating a new thread when there is a new client and keeping listening requests.

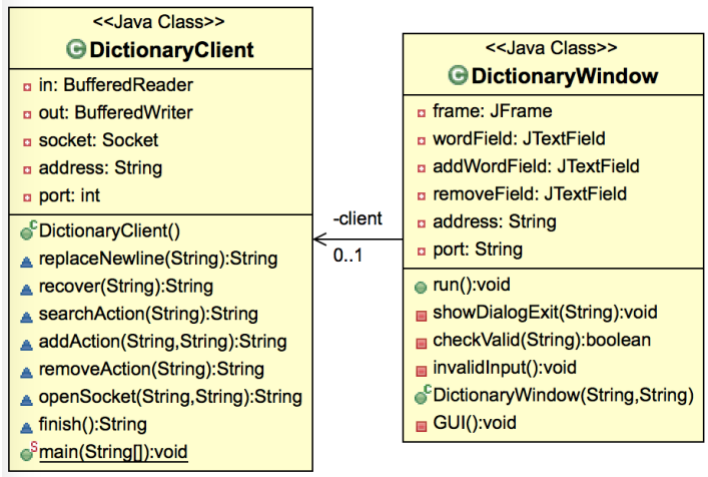
After each connection established, **DictionaryHandler** will take over the next step. It will create I/O streams for communicating with the Client, and processes Client requests, queries data from Dictionary and sends data to the Client.

ServerWindow is the GUI of Server, it can show Server logs and the current state. It also supports to request some operation such as empty the Dictionary or clear Server logs.

Dictionary class is used to control the dictionary. It implements several operations, for examples, empty the dictionary, recover from the file, save to the file, query a word is existing or not, add or remove a word. Each word in the dictionary is an **Entity**, which includes the word's meanings and its timestamp.



(2) Client:



DictionaryWindow is the GUI of Client, it listens to the user's actions and sends the corresponding messages to DictionaryClient.

DictionaryClient handles the connection and communications to the Server, and return Server's responds to DictionaryWindow.

Once there is a Client disconnect with the Server, the Server will back up the current version of Dictionary to the disk. A backup will also be created when the Server close.

3. Protocols

The application uses Strings to communicate between Clients and the Server. The communications follow the protocols above:

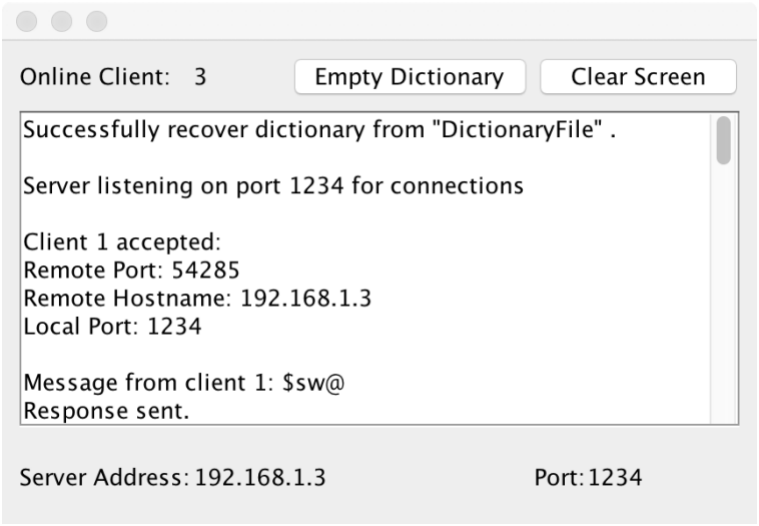
Name: Ruilin Liu
ID: 871076

Pattern	Meaning
$sw@word	Search word and need to return its meanings and timestamp.
$aw@word@meanings#meanings	Add word with several meanings and need to return success or failure. '#' needs to be replaced by '\n'
$rw@word	Remove word and need to return success or failure.
meanings#meanings@date	Successfully get data from Dictionary, '#' needs to be replaced by '\n'
$M@string	Show the string in a message dialog.
$E@string	Exception. Show the string in a message dialog and end the app.
$Q@	Exception. End the app without any dialog.
$success	The operation is successful.

4. GUI

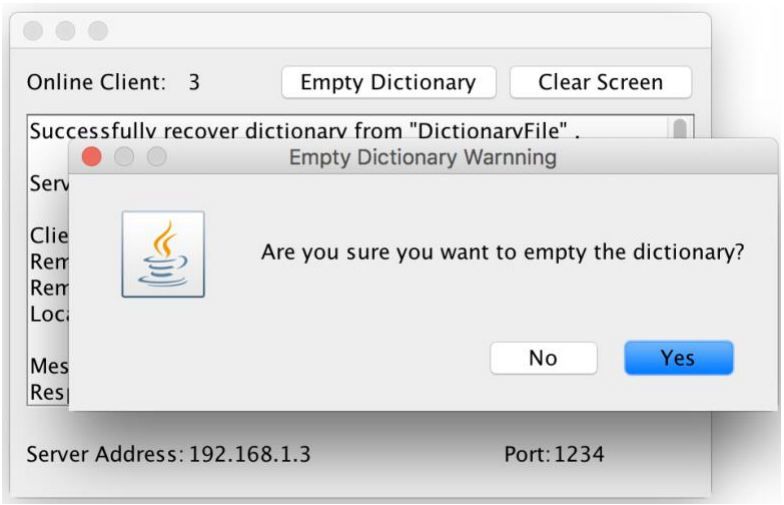
(1) Server

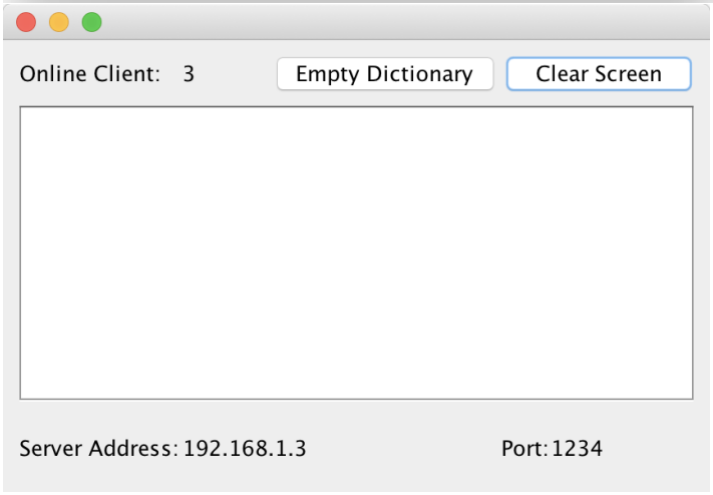
Server UI shows the number of online clients, the current Server address and the port. All server log will be recorded in the text area, the messages from clients followed the protocols above.



Server manager could empty the whole Dictionary by clicking the "Empty Dictionary" button. The server log could be cleared by clicking the "Clear Screen" button.

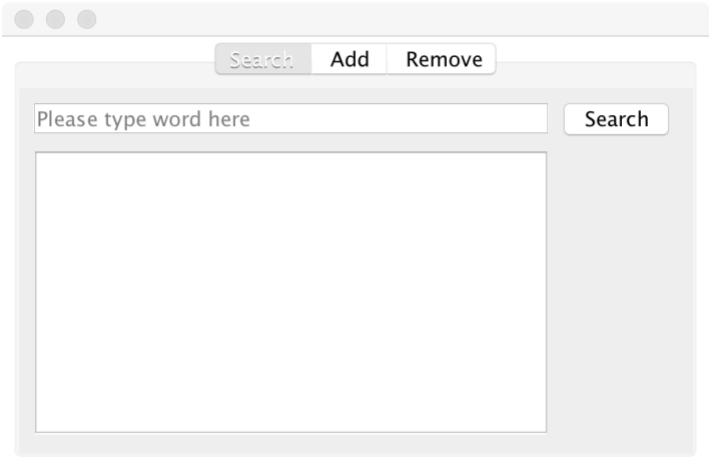
Name: Ruilin Liu
ID: 871076





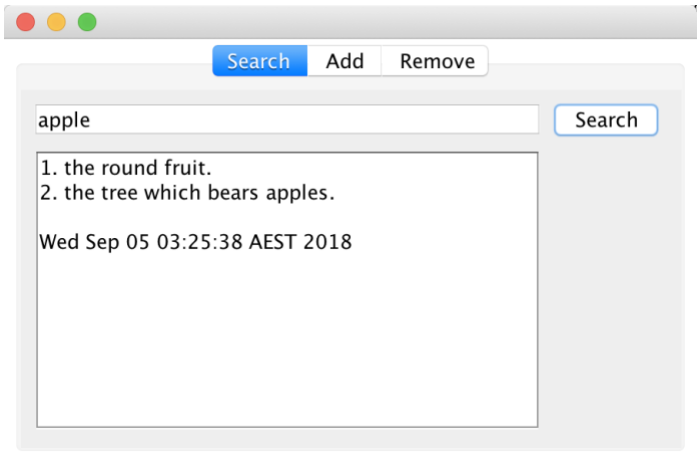
(2) Client

The Client UI includes three panels corresponding to the three operations. In the Search Panel, it will show “Please type word here” initially and the hint will disappear after clicking the field.

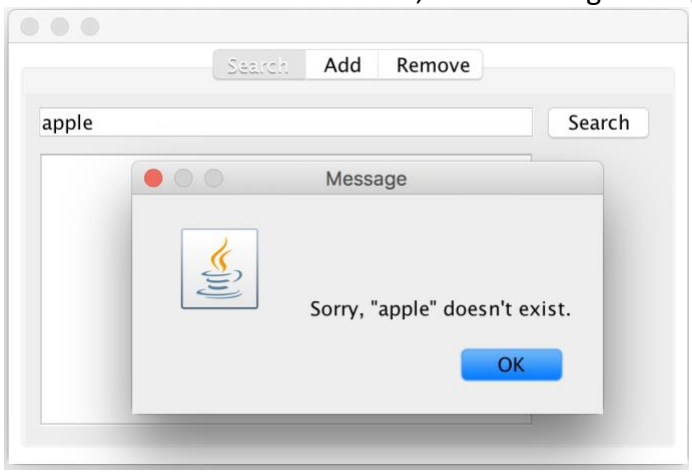


If the word exists in the Dictionary, the meanings of the word and the added time will show in the text area.

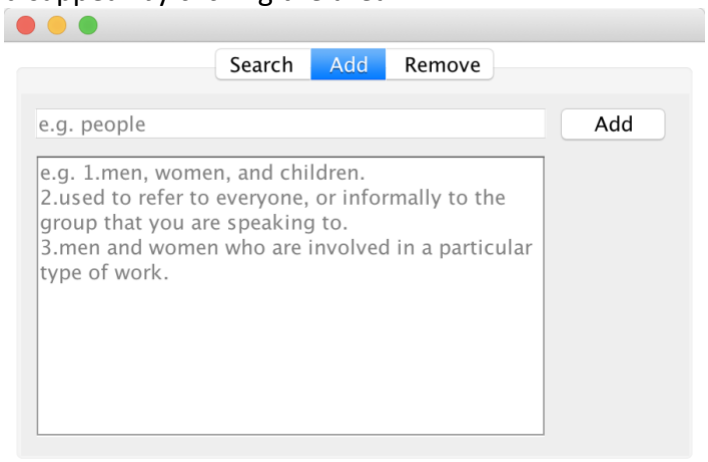
Name: Ruilin Liu
ID: 871076



If the word doesn't exist, the following message will show:

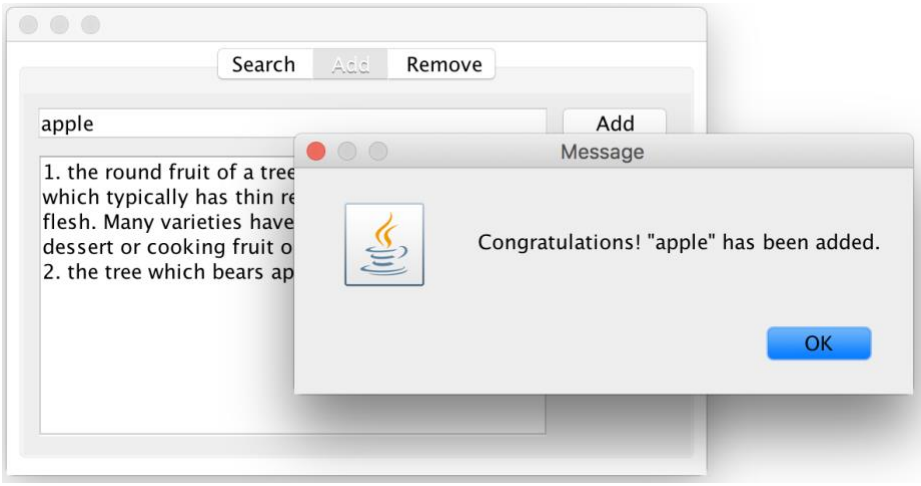


There is an example format shows initially in the add panel. The hint also will disappear by clicking the area.

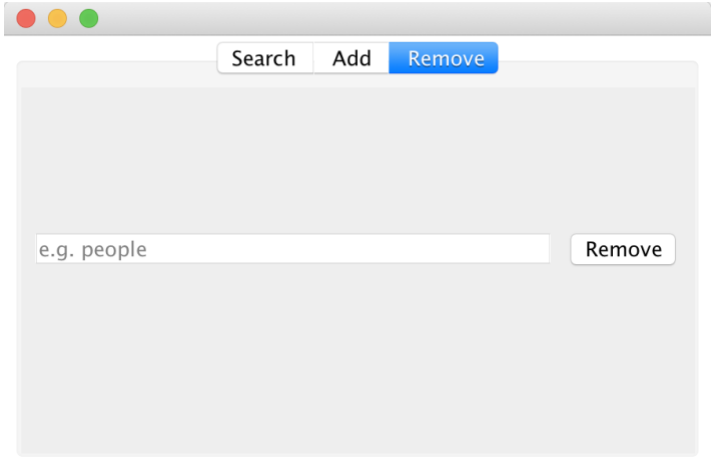


If the word is not in the Dictionary, the following message dialog will pop up. Otherwise, the message will be "Sorry, "apple" has already been added".

Name: Ruilin Liu
ID: 871076

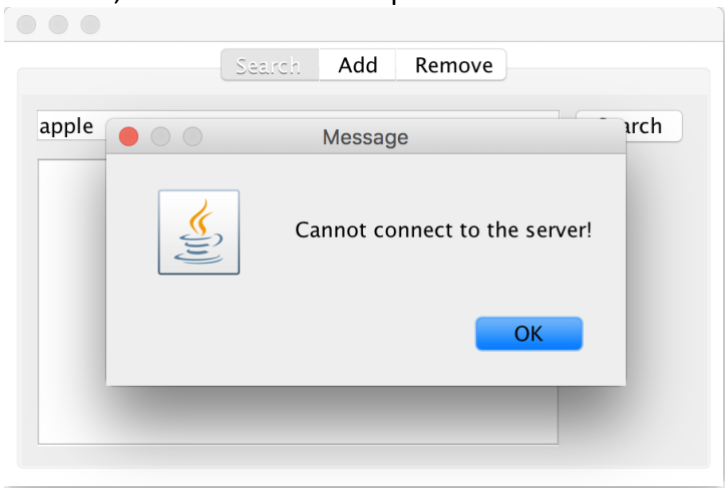


The remove panel only includes one text field, and the dialog will show “Sorry, “apple” doesn’t exist.” or “Congratulations! “apple” has been removed.”



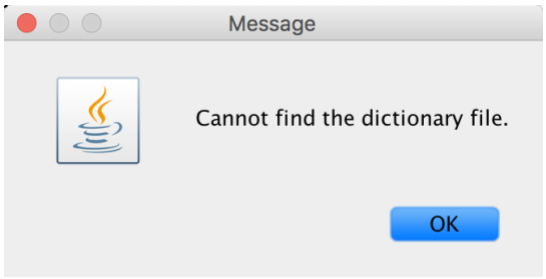
4. Failure Handling

All known exceptions are handled by message dialogs. For example, when the Server is closed, it will throw an exception in Client UI.

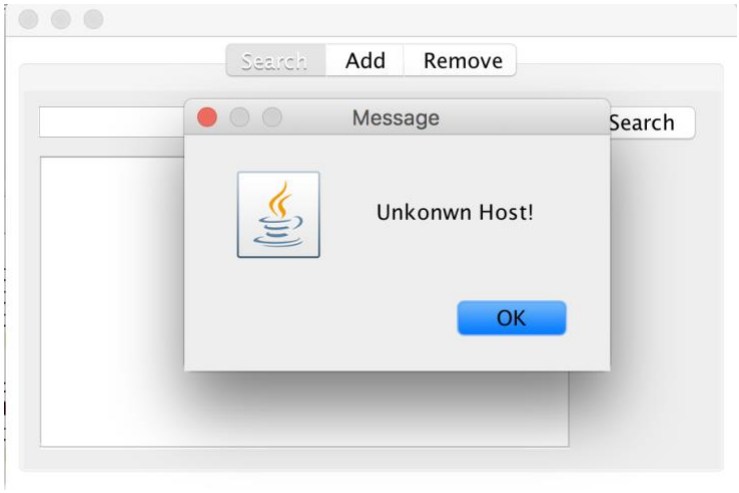


If the dictionary file cannot find when trying to open the Server, it will show a message and the Server UI won’t open.

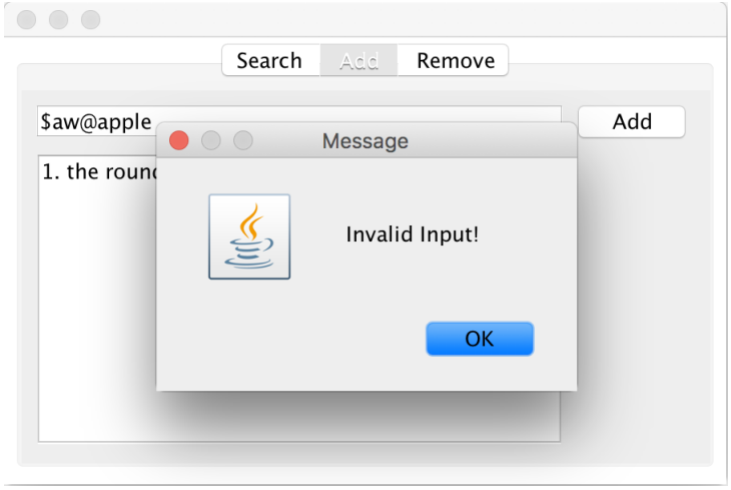
Name: Ruilin Liu
ID: 871076



If the Server address is invalid when trying to open the Client, it will show “Unknown Host!” and the Client UI will be closed.



To ensure the availability of the communication protocols, all user input includes ‘$’ or ‘@’ will be considered as invalid input.



5. Analysis

The application uses thread-per-connection architecture, which is the most suitable method in my personal opinion. As a Dictionary Server, most clients may make requests multiple times, which means Thread-per-request may waste more time. The worker-pool architecture could respond to each request immediately, which is also suitable for a Dictionary Server. It could be more effective than thread-per-connection. However, in this assignment, it is difficult to estimate the appropriate number of workers due to lack of the scale requirement. As a result, the application implements thread-per-connection.

It uses the TCP sockets due to the reliable requirement. The communication between Clients and Server uses “BufferedReader” and “BufferedWriter” to exchange JAVA

Strings. XML and JSON may be special-purpose to exchange messages, while it is almost impossible to exchange too complicated messages in a simple dictionary application. If more features are needed in the future, it would be better to use JSON.

The dictionary uses JAVA object to save and recover from the file, which is an acceptable method, but some error may happen if use different code version. In this case, it is reliable.

6. Conclusion

The application uses thread and socket technologies to implement a multi-threaded dictionary Server and Client. The application also handles exceptions and offers corresponding notifications to users. It chooses thread-per-connection and JAVA object which has some pros and cons, while all choices are reasonable. However, it would be better to use JSON if there is a more stringent format of the dictionary.