Hui Yee, Tan, 1128168

Problem

In a distributed system, a host keeps a dictionary storing some words and their meanings. Then, other hosts make request for access to the dictionary to query the meaning(s) of word, add a new word, update and remove an existing word. Hence, a system is required to deliver requests and services among them. Besides, there are few non-functional requirements for the system: 1. **Concurrency:** Concurrent clients connect to the server and perform operations. 2. **Reliability:** The connection guarantees data delivery. 3. **Fault Tolerance:** Error should be handled properly and should not result in failure of whole system.

To fulfil the requirements, there are two main programs: DictionaryServer, a multi-threaded server which permits concurrent clients to perform operations and DictionaryClient, a client which provides proper interaction with the user. They communicate with each other through TCP socket.

System Components Description

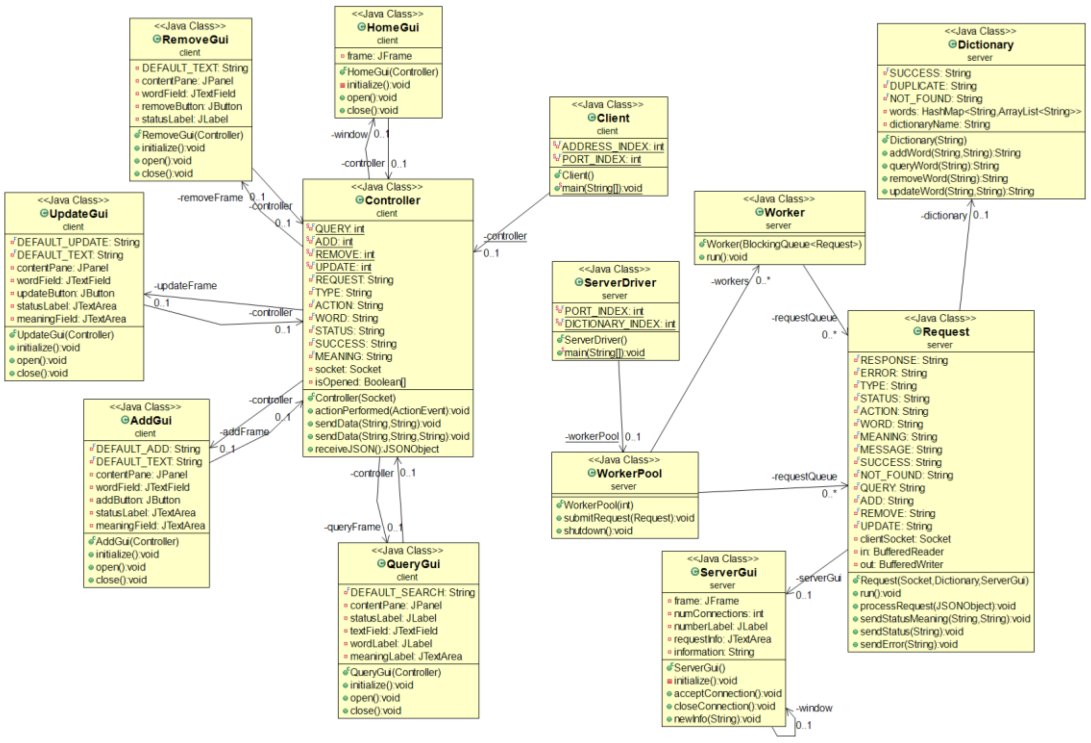

Figure 1: Class Diagram

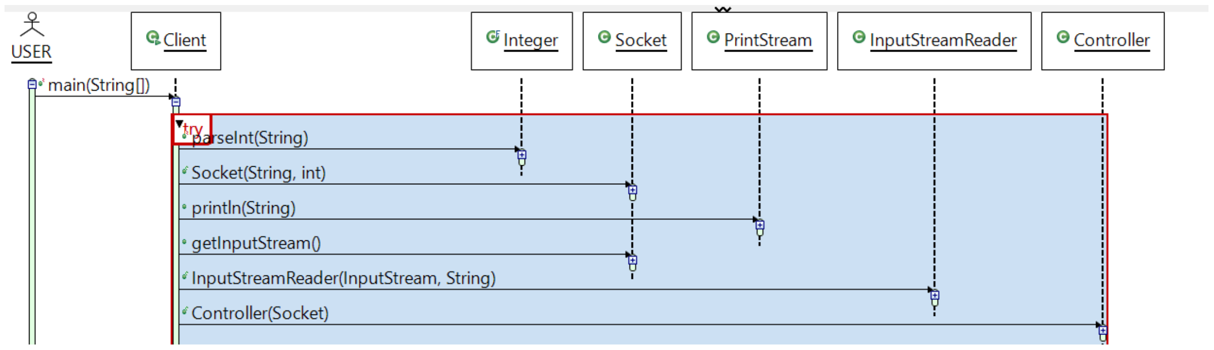

Figure 2: Interaction Diagram (Client)

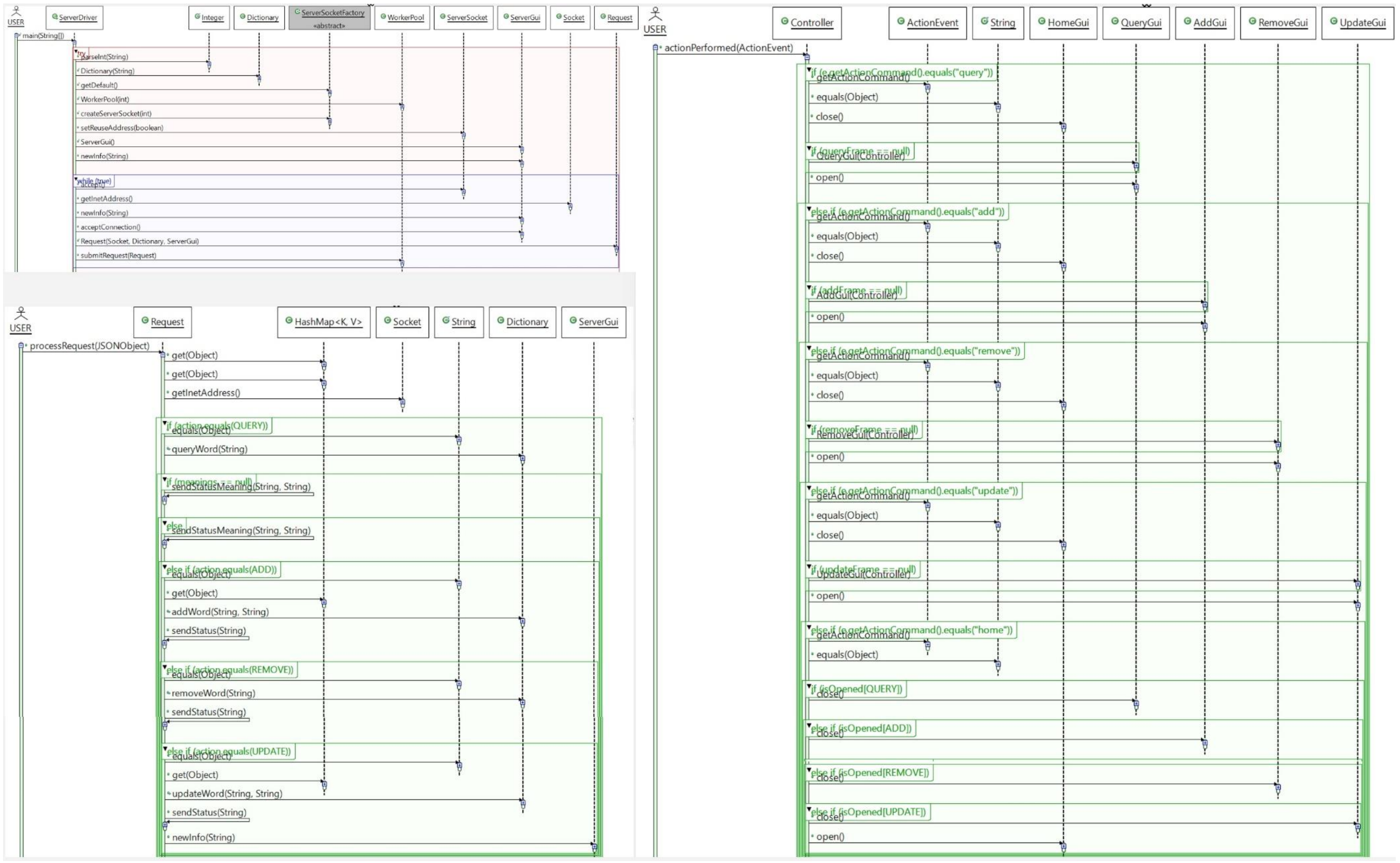

Figure 3: The two diagrams on left hand side show interaction diagram for client while the other diagram interaction diagram for server.

The diagrams shows all the components of the system and the relationship among them. The Client owns a Controller which controls a number of frames and switches between different frames depending on user's operations. The ServerDriver owns a WorkerPool which contains a fixed number of Worker and a list of Request (connection request from client). The Client and ServerDriver do not have direct relationship since they interact through socket.

Critical Analysis

The system has the following advantages and disadvantages due to the decisions made:

	Design Decisions	**Advantages**	**Disadvantages**
1	Client Server Architecture	Scalability, security, consistency	Unavailability, expensive to increase system scale
2	Worker Pool Architecture	Low memory usage, enhance performance	-
3	TCP	Reliability, congestion control	Lower efficiency, slower data delivery, higher resource consumption
4	JSON	Enhance performance, readability and maintainability, simplify data representation and visualization	-
5	Read the dictionary file in memory and search there	Efficient searching	High memory usage
6	Concurrency	Synchronization	-

More detailed explanations of why those design decisions are made and how the system could be improved given different design decisions are illustrated below:

1. Client Server Architecture
   The system utilizes a client server model, a centralized server provides services to its clients. The model provides few benefits: 1. Scalability: new servers and clients could be added to the system easily, creating a larger and network. 2. Security: all data are kept in a centralized location, simplifying user authorization and authentication control. 3. Consistency: Due to data centralization, all the clients access to the exact same dictionary to search for a word and perform modification, ensuring there is no data discrepancy.

   However, the system may be further improved by using a peer to peer model (P2P) which contains a decentralized collection of hosts which exchange information with each other or specific users only. There are few advantages: 1. Availability: In the client server architecture, the centralized server must be online all the time to provide services to it client while in P2P, since every client is the server, when the centralized server is not available, other clients could act as servers and provide services to others. 2. Scalability: In the client server model, increased scale comes with massive capital expense but in the P2P, since each host takes part in data delivery and brings its infrastructure to the system, increased scale comes at no cost. Nonetheless, P2P may compromise system security and data consistency, hence a careful planning should be made before the implementation.

2. Worker Pool Architecture
   The multi-threaded server implements a worker pool architecture. This is because the server produces a fixed number of worker threads and recycles threads to fulfil requests from clients instead of generating new threads every time a request is satisfied. This lowers the memory usage and helps enhance the performance of the system. Alternatives such as thread-per-request and thread-per-connection which are simpler to set up are not considered in this case due to higher resources consumption and slower execution.

3. TCP
   All communication between clients and servers takes place via TCP socket. TCP socket ensures reliable transmission through its ability to retransmit lost packets, perform in-order delivery, error detection and recovery. Moreover, TCP offers congestion control to prevent congestion when networks are overloaded. Nevertheless, compared to its substitute, UDP, TCP is less efficient and slower, as well as consumes more resources because of its complexity to guarantee reliability.

   Communication taking place through UDP socket could help advance the system since it provides faster delivery of data and increases efficiency. Though, this comes at a cost of compromising reliability, hence extra effort is needed to develop an infrastructure that delivers reliable communication over UDP.

4. JSON
   The system makes use of data format JSON since it is more succinct and compact, consuming lesser time to parse and produce data such as different types of requests by clients and responses by server. Subsequently, this decreases the size and bandwidth of data transfers, speeding up the performance and efficiency of data processing. Furthermore, it is easy to read and write in JSON data format, supporting data readability and maintainability. It also supports the data types used in the system such as string and null, contributing to simple data representation and validation.

   Other data formats such as XML and Java Object Serialization are considered less suitable. XML supports other data types, including binary data and comments, offering greater flexibility than JSON. Nonetheless, the message exchange protocol does not apply these data types, therefore it does not need the greater data flexibility from XML. While, Java Object Serialization is less memory efficient compared to JSON.

   Message Exchange Protocol using JSON:

   i. Message Exchange Flow
      a. *Client sends request:* The client prepares a JSON request message containing "action" of request and other information, serializes and sends it to server.
      b. *Server receives request:* The server receives the JSON message, deserializes and extract the "action" of request to process to the request accordingly.
      c. *Server sends response:* The client prepares a JSON response message consisting of related data and possibly error message , serializes and sends it to client.
      d. *Client receives response:* The client receives the JSON message, deserializes and process the data appropriately.

ii. Message Type
   a. *Request Message*
      Type: To indicate whether it is a request or a response message e.g., "REQUEST"
      Action: The type of operations requested by client e.g., "QUERY", "ADD", "REMOVE", "UPDATE"
      Word: The word related to the action performed. A valid word consists only characters and it could be capital letters or small letters.
      Meaning: The meaning of the word. This is an optional field, where only "ADD" and "UPDATE" message will have.
   b. *Response Message*
      Type: To indicate whether it is a request or a response message e.g., "REPONSE"
      Status: The status of the operation performed e.g., "Success", "Duplicate", "Not found"
      Meaning: The meaning of the word. This is an optional field, where only response to "QUERY" message will have

5. Read the dictionary file in memory and search there
   The server reads the dictionary file in memory and searches every line to determine whether it contains the target word as prefix. The article "Java – Search in File as fast as possible" compares different methods to search a word in file containing 6 million records and this method ranks third among of 16 methods. Also, given that the server only searches first few characters instead of the whole line, this method is able to search for the target word quite efficiently.

   Besides, since this method keeps the whole file in memory, it gives rise to higher memory usage. Yet, The Oxford English Dictionary estimates that there are 174,000 words in current use, hence keeping all records in the file should not consume too much memory.

6. Concurrency
   Since multiple clients could connect to the server and perform operations concurrently, they access and manipulate the shared resource, dictionary file. To prevent inconsistent state, access to the do the dictionary file is synchronized through "synchronized" method.

Excellence

1. Proper interaction with users
   The system provides a clear and simple interface which could be easily understood and navigated by the users. Once the host runs the client program successfully, an application window with four buttons will pop up. These four buttons represent four different operations which could be performed by the users: searching the meaning of a word, adding a new word, removing and updating an existing word. Then, the user just simply clicks on the button which represents the operation he would like to perform and follow the instructions to type the information in the fields next page.

Furthermore, the GUI notifies the users any errors occurred in the system, helping them to understand what went wrong and probably suggestions to fix them. I aggregate the errors into two different types: 1. Connection errors: errors which happen before the connection is established successfully; 2. Input errors: errors which happen after the connection is established successfully and are likely to be related with the inputs. If connection error occurs in the system, the system terminates the program and describe the errors occur in the console. The connection errors mainly occur because of incorrect command line arguments given and the user could simply run the program with correct command line arguments in the terminal. On the other hand, when input error occurs, the system gives description of the problem in GUI, helping the suers to identify and fix the errors.

Creativity

1. GUI for managing the server
   The system provides a GUI for the server side. The GUI gives the users useful information such as the number of connections and the interaction between the server and client. Specifically, the GUI will provide information like the address of client connected, the request received from the client and so on.

2. Thread pool
   I implement my own implementation of a thread pool for the system using thread-per-connection concept. The server has a fixed number of threads to process client connections. It associates a thread with a new client connection and recycles the thread for other connections when client closes the connection.

   There are three Java class: Request, Worker and WorkerPool in my program. The Request class represents a client connection while the Worker class represents a worker thread. The WorkerPool contains a fixed number of worker thread as well as a queue of client connections waiting to be associated. When a client tries to connect to the server, a new Request with client information will be created and submitted to the queue. Then, the worker thread gets a task from the queue and starts to process requests from the client.

   The queue containing all client connections is a shared resources among all the threads. Hence, I used BlockingQueue to ensure thread safety when putting a connection into and getting a connection from the queue.

Conclusion

In summary, a solution which fulfils the functional and non-functional requirements has been developed. The decision choices made for the program like client-server architecture, worker pool architecture and so on, along with its justification have been provided as well. Further improvement to the solution could be made by following the suggestions.