**Report**

1. **Design Details**
2. **Model:**

A screenshot of text

Description automatically generated

Fig: UML diagram of the project

The project tries to mimic MemeCached over TCP protocol. It consists of 2 main components: Server-which serves the requests & Client-which connects to server and sends command request(s). The Server acts as a centralized entity which serves multiple clients with Concurrency & Synchronization.

MainHandler class spawns number of clients based on the NO\_OF\_CLIENTS static class variable. Each request creates a thread which takes care of client execution, sending requests, processing requests & fetching the reply from the server. Each thread is temporarily stopped using sleep() command for random time up to 5 seconds.

The Server class binds the Server’s IP Address to the port so that it can listen client requests through the same. Later, it creates object of ServerHandler class which establishes Client socket connection, interprets input command, checks for exception (ie. validation), processes the commands based on the corresponding type and send back appropriate message(s) to the client.

1. **Data Structures & File Storage:**

The server uses ConcurrentHashMap as a data structure for storing <key, value> pairs with concurrent operations. Later this is stored onto the csv file for persistent storage. Every client creation initializes the map from the file and every data manipulation stores the map data back to file.

This project creates 2 underlying csv files:

1. data.csv

Stores <Key, Value> pair with comma-delimited format.

1. flags.csv

Stores <Key, Flag\_value> pair with comma-delimited format. This flag value is later used in cas operations.

1. **Commands:**

The project supports 3 types of commands:

1. Storage command:

* set
* add
* replace
* append
* prepend
* cas

1. Retrieval command:

* get
* gets

1. Deletion command:

* delete

The storage command accepts input in the format as below:

<command name> <key> <flags> <exptime> <bytes> [noreply] \r\n<value> \r\n

The cas command accepts input format as below:

cas <key> <flags> <exptime> <bytes> <cas unique> [noreply] \r\n<value> \r\n

The retrieval command accepts input format as below:

get <key>\*\r\n

gets <key>\*\r\n

The delete command accepts input format as below:

delete <key> [noreply]\r\n

**Note: This project supports <flags>, <exptime>, <cas unique> & <noreply> command fields. Although the server accepts <exptime>, the implementation of <exptime> is yet to be completed.**

1. **Execution Instructions:**

The project contains 2 files with main functions:

1. Server.java
2. MainHandler.java

For proper execution of project, first compile and run Server.java file & then run MainHandler.java file. The concurrent number of clients can be tweaked using class variable NO\_OF\_CLIENTS present in the MainHandler class.

1. **Experimental Evaluation**

Below are the benchmarks for which the existing server is tested for on the local machine:

1. No. of concurrent clients server can handle simultaneously: 1000+ (server successfully handled 1000 clients on local machine)
2. Response Time of server per request: Approx. 5 milliseconds (Depends on the Command type, server load & no of clients connected at that time)

For a closed system, λ = µ where λ = Arrival rate & µ = Service Rate.

The response time = 5milliseconds

λ = µ = throughput = 1/(5 \* 103) = 200 requests per second