Internet Technologies Lab Report
Assignment 1

Md Sahil
BCSE IV
Roll-001710501029

1 Problem Statement

Implement a TCP-based key-value store. The server implements the key-value store and clients make use of it. The server must accept clients' connections and serve their requests for 'get' and 'put' key value pairs. All key-value pairs should be stored by the server only in memory. Keys and values are strings. The client accepts a variable no of command line arguments where the first argument is the server hostname followed by port no. It should be followed by any sequence of
" get ⟨ key⟩ "
and/or
" put ⟨ key⟩ ⟨ value⟩ ".

```
./client 192.168.124.5 5555 put city Kolkata put country India get country get city get Institute
India
Kolkata
<blank>
```

The server should be running on a TCP port. The server should support multiple clients and maintain their key-value stores separately.

Implement authorization so that only few clients having the role "manager" can access other's key-value stores. A user is assigned the "guest" role by default. The server can upgrade a "guest" user to a "manager" user

2 Design & Implementation

The program is implemented using Java. The program is divided into two sections the *Server* and the *Client*.

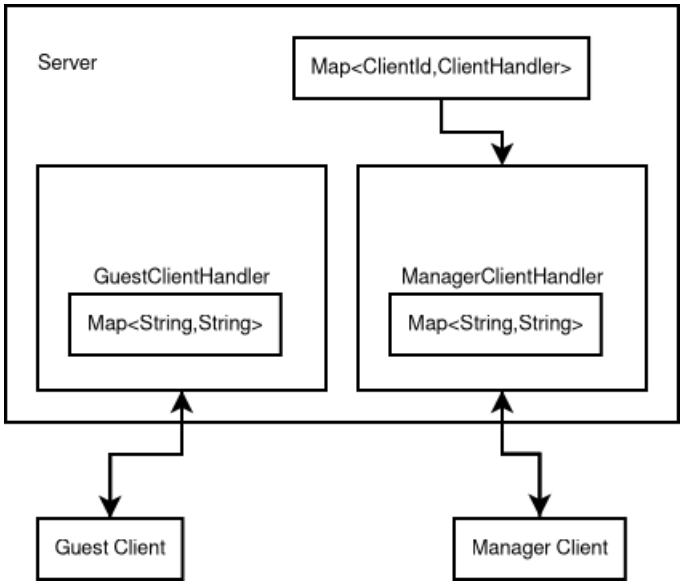

Figure 1: Flow Diagram

The Client program has 2 threads running. The main thread runs the input console. The console takes input from the user and sends it to the server accordingly.

The Server program has a main thread to run the console and a seperate thread for each client connected to it. The ClientHandler class (implements the Runnable interface) handle the communication between the assigned client and the server. When multiple clients are connected multiple ClientHandler instances are generated. The ClientHandler instances also store the client data in a HashMap⟨String,String⟩ data structure.

The Port number of each client is treated as the ID for the client. Client data is stored in a HashMap as a member variable of the ClientHandler interface.

The Server terminal can upgrade guest clients to managers or downgrade managers to guest.

3 Usage and Features

3.1 Directory structure

The directory structure of the program is as follows:

```
.
|-- Makefile
|-- bin
|   |-- client
|   |   |-- Client$Listener.class
|   |   |-- Client.class
|   |   `-- Main.class
|   `-- server
|       |-- ClientHandler.class
|       |-- GuestClientHandler.class
|       |-- Main.class
|       |-- ManagerClientHandler.class
|       |-- Server$Terminal.class
|       `-- Server.class
|-- run.sh
`-- src
    |-- client
    |   |-- Client.java
    |   `-- Main.java
    `-- server
        |-- ClientHandler.java
        |-- GuestClientHandler.java
        |-- Main.java
        |-- ManagerClientHandler.java
        `-- Server.java

6 directories, 18 files
```

3.2 Compilation

In order to compile the program just run make while at the root directoy.

```
$ make
```

To run the server use to following command.

```
$ java -cp bin/ server/Main <port>
```

To run clients use the following command. The port number is the port in which the server runs.

```
$ java -cp bin/ client/Main <port>
```

3.3 Usage

3.3.1 Server commands

- List connected clients

  ```
  # list
  ```

- Upgrade guest clients

  ```
  # upgrade <client_port>
  ```

- Downgrade guest clients

  ```
  # downgrade <client_port>
  ```

3.3.2 Client commands

- Put command

```
$ put <key> <value>
```

- Get command

```
$ get <key>
```

The get and put commands are concatinatable.

3.3.3 Manager commands

Along with all the client commands the manager also has some additional commands. All manager commands are accompanied by the prefix *mgr*.

- List connected clients

```
# mgr list
```

- Put or get command on client

```
# mgr <clienti_d> put <key> <value> get <key> ...
```

3.4 Features

- The program is an interative program. There are interative shells running on both server and client end points.
- Depending on the permissions of the terminal, the shell prompt changes. For manager clients and the server, the shell prompt is #. For regular clients the shell prompt is **$**.
- The program is multi-threaded. Multiple clients can stay connected to the server simultaneously.
- Multiple servers can be run on different ports on the machine. A client can be connected to a single server only.

3.5 Sample I/O

- Server I/O

```
^^IServer started at :127.0.1.1
^^I# Connected to :/127.0.0.1:39328
^^IConnected to :/127.0.0.1:39330
^^Ilist
^^IGuest:39328
^^IGuest:39330
^^I39330: put city Kolkata
^^I39330: get city
^^I# upgrade 39328
^^IUpgrading 39328
^^I# Connected to :/127.0.0.1:39328
^^I39328: mgr lisg
^^I39328: mgr list
^^I39328: mgr 39330 get city
^^I#
```

- Client 1 I/O

```
^^IConnection established with server!
^^I$ put city Kolkata
^^I$ get city
^^IKolkata
^^I$
```

- Client 2 I/O

```
^^IConnection established with server!
^^Iupgrade
^^I# mgr lisg
^^IInvalid command
^^I# mgr list
^^IManager:39328
^^IGuest:39330
^^I# mgr 39330 get city
^^IKolkata
^^I#
```