TCP BASED KEY VALUE STORE

An in-memory key value store using java socket programming

Abstract

A key-value database, or key-value store, is a data storage paradigm designed for storing, retrieving, and managing associative arrays, a data structure more commonly known today as a dictionary or hash table. Some key value stores maintain data in-memory, while others use solid-state drives or rotating disks. Among the above mentioned two, in-memory key value store provides fast and efficient processing. In this document, we present a java based key value store for storing and retrieving data.

Amartya Choudhury
BCSE -IV
Roll-001610501026
amartyachowdhury98@gmail.com
Internet Technology Laboratory

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 serverhostname 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>
```

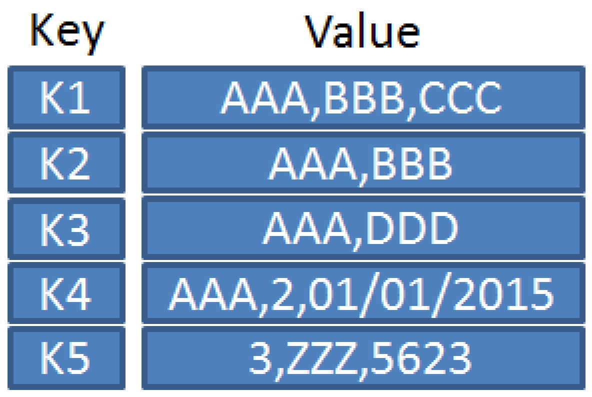
The server should be running on a TCP port. The server should support multiple clients and maintain their key-value stores separately. Implement authorization so that only few clients having the role "manager" can access other's key-value

stores. A user is assigned the "guest" role by default. The server can upgrade a "guest" user to a "manager" user.

Introduction:

Businesses must ensure that their data is durable by keeping it safe on disk. However, in-memory databases are also useful in some instances, such as the following:

- **High-speed caching:** To remove read workloads from the database of record, to reduce the cost of hardware and software licenses, and to prevent distributed denial-of-service (DDoS) attacks from affecting a live system.
- **Transient data holding:** Data that isn't very important and that has a known lifespan. A typical web application's details on users' sessions are good examples.
- **Analysis before storage:** For example, large memory systems are used by scientists to analyze stellar observations. Most of the time, telescopes are looking at a blank bit of sky — no need to store that data! Scientists analyze the data quickly in memory, and store only what's useful.



Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Fig 1: A table showing different formatted data values associated with different keys

Because of their uncomplicated nature, many in-memory databases are also key-value stores. In-memory use also lends itself to high-speed applications. Retrieving a record using its unique key is the quickest way to retrieve data, so key-value stores and in-memory databases are a natural fit.

- Redis began as an in-memory database. Indeed, Redis can still be used in this way. Redis does operate as a single process, though, so you need to run multiple instances of Redis on each server in order to get full utilization of its resources.
- Hazelcast is an in-memory NoSQL database that replicates its data to other Hazelcast nodes in the cluster. It is an open-source product, but it's also offers a commercial product (Hazelcast Enterprise) with more features. Hazelcast is used within the commercial version of the OrientDB triple store. OrientDB uses Hazelcast in order to provide high availability. Hazelcast effectively provides replication of OrientDB's data structures

Design:

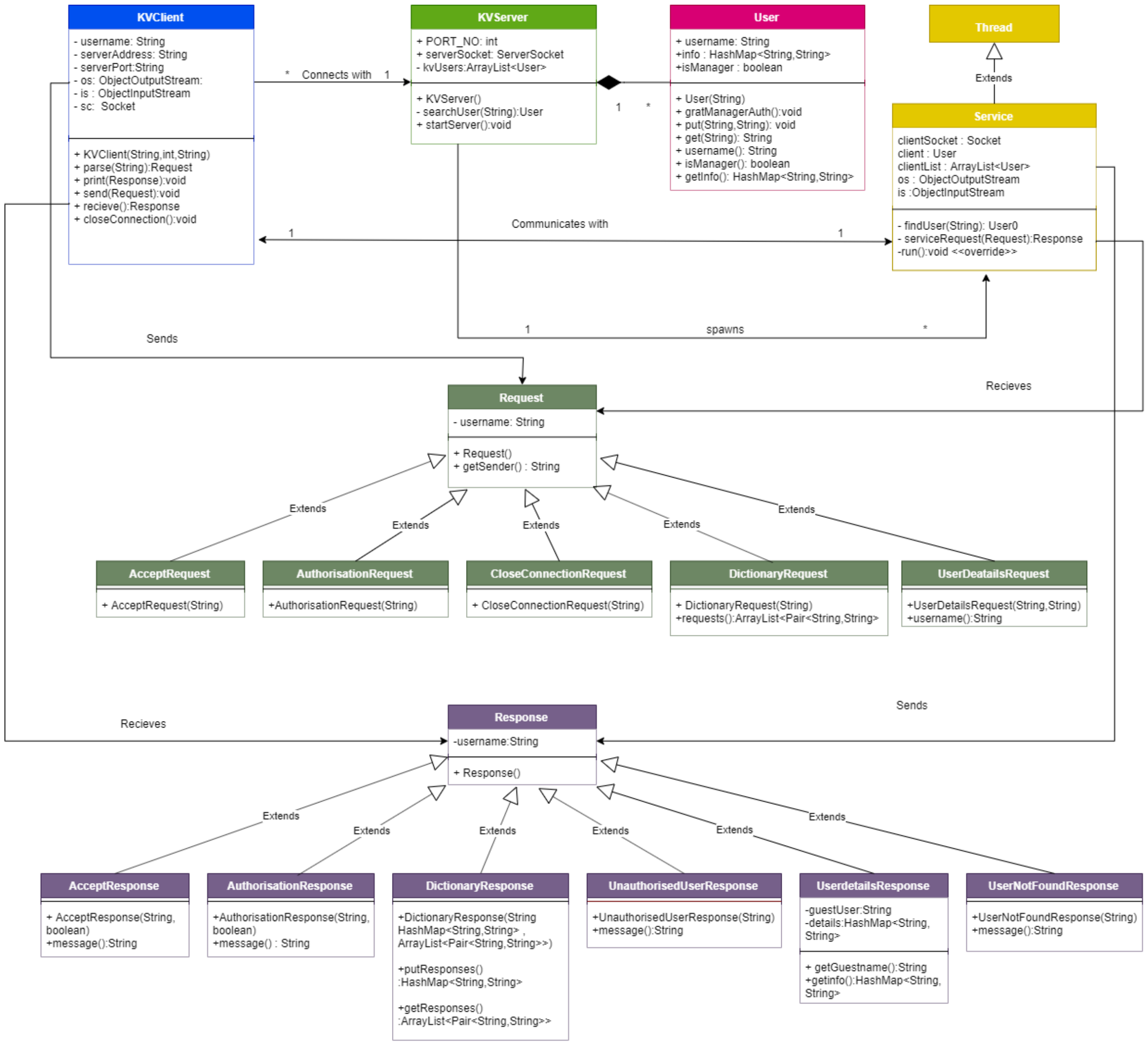


Fig 2 :Class Diagram of Key Value store

1. The Key Value Server listens on port number 8000 for incoming connections for clients
2. It maintains an in-memory dynamic list of User objects .
3. A User has a userId , his / her own dictionary , and whether he / she is a manager or not .
4. A key value client knows the socket address of the server , it connects to it
5. The server detects an incoming connection and spawns a new service thread to serve the client
6. The client and the service communicate over a TCP connection.The client sends requests , the server sends back responses.
7. Requests are of the following types and are handled in a separate manner by the service thread –
    - AcceptRequest(Client requests the server to accept its connection)
    - AuthorisationRequest(Client requests server to promote to manager Role)
    - UserDetailsRequest(Client requests dictionary of another user , invalid request if client is not a manager)
    - DictionaryRequest(Client sends a number of put / get commands to modify / read his own dictionary)
    - CloseConnectionRequest(Client Server to close the connection)
8. The service threads sends back “Responses” which are of the following types –
    - AcceptResponse(Different responses if the client is new , or an existing client wanting to reconnect to his / her store)

- AuthorisationResponse(Conveys , if not already a manger that the request was granted)
- DictionaryResponse(Returns the values for the keys in the request , if they exist)
- UnauthorisedUserResponse(Sent , if a user who is not a manager asks for dictionary of another User)
- UserDetailsResponse(Dictionary of another user , sent to a manager)
- UserNotFoundResponse(Sent to a manager if the user whose details he/she requests doesn't exist)

Code Snippets:

- client execution:

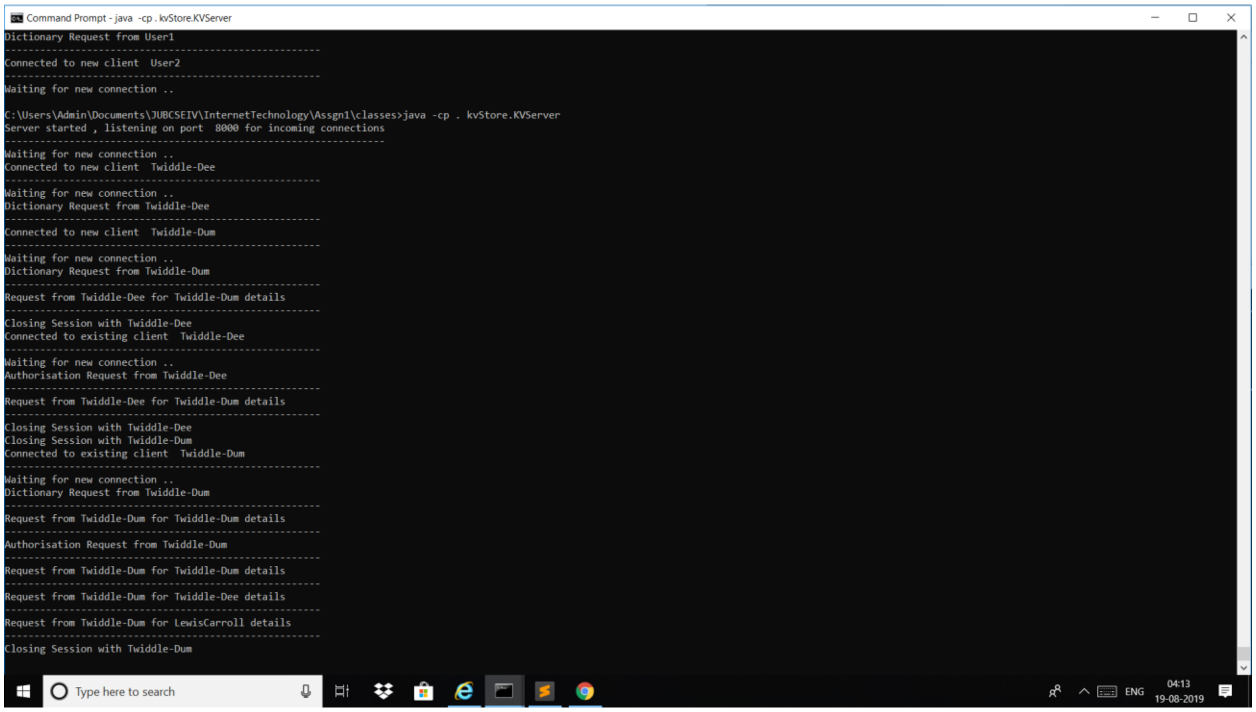
```
	public static void main(String args[]){
			//fetch server credentials
			String serverInetAddress = args[0];
			int serverPort =Integer.parseInt(args[1]);
			//fetch client username
			String clientUsername = args[2];

			KVClient client =null;

			//create new client
			try{
				client = new
	KVClient(serverInetAddress,serverPort,clientUsername);
			}
			catch(UsernameExistsException e){
				e.printStackTrace();
				return ;
				}
			Scanner sc = new Scanner(System.in);
			//until connection closed
			for(;;){
				String input = sc.nextLine();
				if(input.equals("Exit")){
					client.closeConnection();
					break;
```

```
                        }
                        //parse input
                        Request rq = null;
                        try{
                                rq = client.parse(input);

                        }
                        catch(InvalidInputException e){
                                e.printStackTrace();
                        }
                        catch(NoSuchElementException e){
                                e.printStackTrace();
                        }
                        //send request
                        client.send(rq);
                        //print response
                        client.print(client.recieve());
                        System.out.println("------------------------------------------
      ---------------");

                }
        }
```

-service Thread execution:

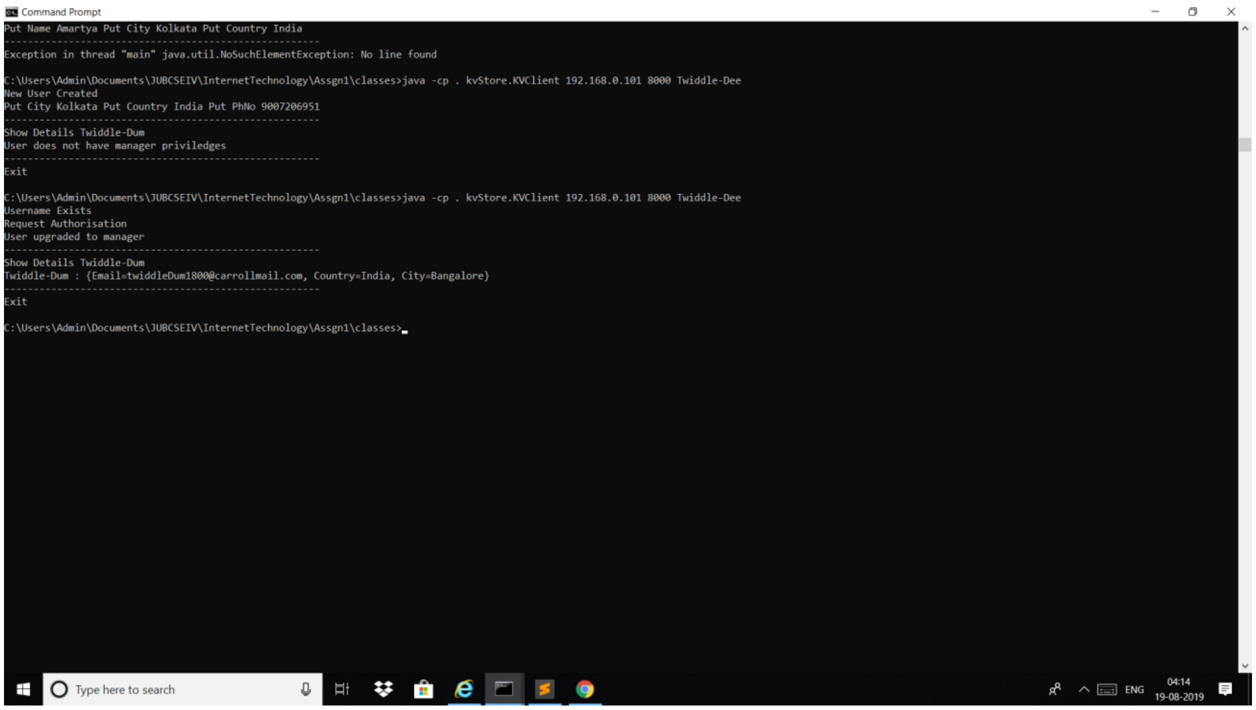
```
public void run(){
                for(;;){
                        try{
                        Request rq = (Request)is.readObject();
                        if(rq instanceof CloseConnectionRequest){
                                clientSocket.close();
                                System.out.println("Closing Session with " +
client.username());
                                break;
                        }
                        Response rs = serviceRequest(rq);
                        os.writeObject(rs);
                        }
                        catch(IOException e){
                                e.printStackTrace();
                        }
                        catch(ClassNotFoundException e){
                                e.printStackTrace();
                        }
                        System.out.println("------------------------------------------
--------------");
                }
        }
```

Output:



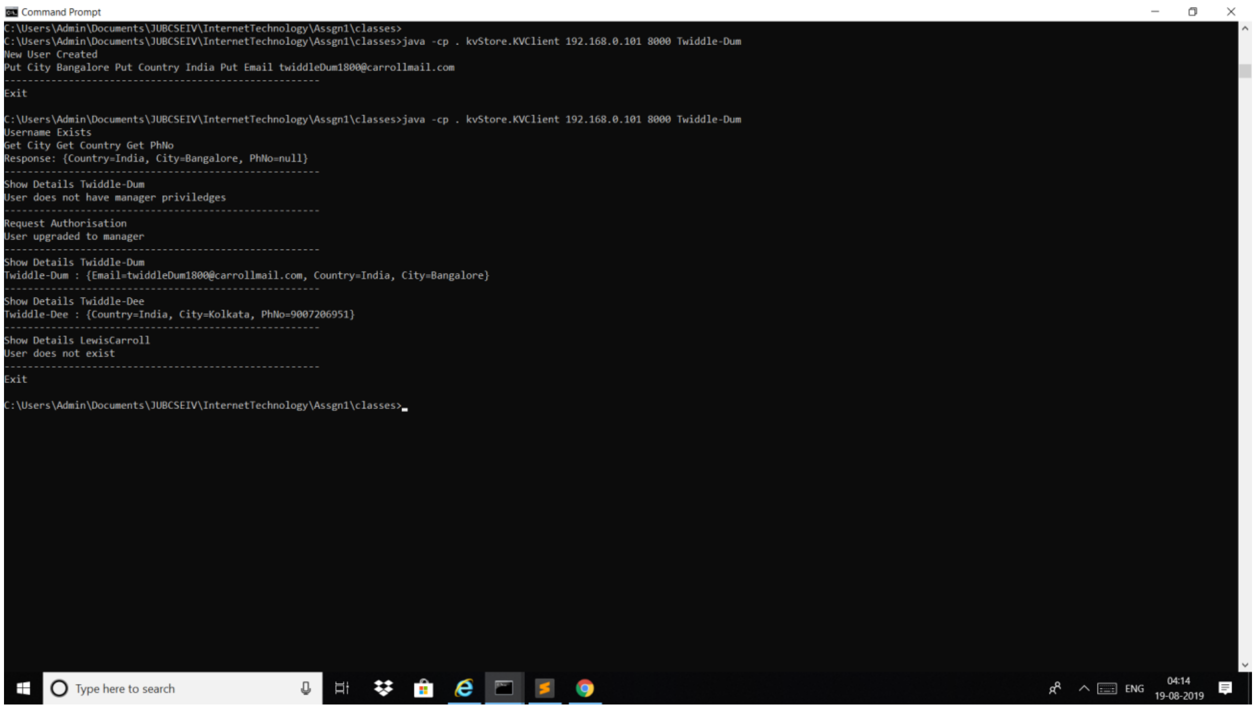
```
Command Prompt - java -cp . kvStore.KVServer
Dictionary Request from User1
------------------------------------------------------
Connected to new client  User2
------------------------------------------------------
Waiting for new connection ..

C:\Users\Admin\Documents\JUBCSEIV\InternetTechnology\Assgn1\classes>java -cp . kvStore.KVServer
Server started , listening on port  8000 for incoming connections
-----------------------------------------------------------------
Waiting for new connection ..
Connected to new client  Twiddle-Dee
------------------------------------------------------
Waiting for new connection ..
Dictionary Request from Twiddle-Dee
------------------------------------------------------
Connected to new client  Twiddle-Dum
------------------------------------------------------
Waiting for new connection ..
Dictionary Request from Twiddle-Dum
------------------------------------------------------
Request from Twiddle-Dee for Twiddle-Dum details
------------------------------------------------------
Closing Session with Twiddle-Dee
Connected to existing client  Twiddle-Dee
------------------------------------------------------
Waiting for new connection ..
Authorisation Request from Twiddle-Dee
------------------------------------------------------
Request from Twiddle-Dee for Twiddle-Dum details
------------------------------------------------------
Closing Session with Twiddle-Dee
Closing Session with Twiddle-Dum
Connected to existing client  Twiddle-Dum
------------------------------------------------------
Waiting for new connection ..
Dictionary Request from Twiddle-Dum
------------------------------------------------------
Request from Twiddle-Dum for Twiddle-Dum details
------------------------------------------------------
Authorisation Request from Twiddle-Dum
------------------------------------------------------
Request from Twiddle-Dum for Twiddle-Dum details
------------------------------------------------------
Request from Twiddle-Dum for Twiddle-Dee details
------------------------------------------------------
Request from Twiddle-Dum for LewisCarroll details
------------------------------------------------------
Closing Session with Twiddle-Dum
```

Fig 3: KVServer



```
Command Prompt
Put Name Amartya Put City Kolkata Put Country India
------------------------------------------------------
Exception in thread "main" java.util.NoSuchElementException: No line found

C:\Users\Admin\Documents\JUBCSEIV\InternetTechnology\Assgn1\classes>java -cp . kvStore.KVClient 192.168.0.101 8000 Twiddle-Dee
New User Created
Put City Kolkata Put Country India Put PhNo 9007206951
------------------------------------------------------
Show Details Twiddle-Dum
User does not have manager priviledges
------------------------------------------------------
Exit

C:\Users\Admin\Documents\JUBCSEIV\InternetTechnology\Assgn1\classes>java -cp . kvStore.KVClient 192.168.0.101 8000 Twiddle-Dee
Username Exists
Request Authorisation
User upgraded to manager
------------------------------------------------------
Show Details Twiddle-Dum
Twiddle-Dum : {Email=twiddleDum1800@carrollmail.com, Country=India, City=Bangalore}
------------------------------------------------------
Exit

C:\Users\Admin\Documents\JUBCSEIV\InternetTechnology\Assgn1\classes>_
```

Fig 4: Client 1



```
C:\Users\Admin\Documents\JUBCSEIV\InternetTechnology\Assgn1\classes>
C:\Users\Admin\Documents\JUBCSEIV\InternetTechnology\Assgn1\classes>java -cp . kvStore.KVClient 192.168.0.101 8000 Twiddle-Dum
New User Created
Put City Bangalore Put Country India Put Email twiddleDum1800@carrollmail.com
-----------------------------------------------------
Exit

C:\Users\Admin\Documents\JUBCSEIV\InternetTechnology\Assgn1\classes>java -cp . kvStore.KVClient 192.168.0.101 8000 Twiddle-Dum
Username Exists
Get City Get Country Get PhNo
Response: {Country=India, City=Bangalore, PhNo=null}
-----------------------------------------------------
Show Details Twiddle-Dum
User does not have manager priviledges
-----------------------------------------------------
Request Authorisation
User upgraded to manager
-----------------------------------------------------
Show Details Twiddle-Dum
Twiddle-Dum : {Email=twiddleDum1800@carrollmail.com, Country=India, City=Bangalore}
-----------------------------------------------------
Show Details Twiddle-Dee
Twiddle-Dee : {Country=India, City=Kolkata, PhNo=9007206951}
-----------------------------------------------------
Show Details LewisCarroll
User does not exist
-----------------------------------------------------
Exit

C:\Users\Admin\Documents\JUBCSEIV\InternetTechnology\Assgn1\classes>_
```

Fig 5: Client 2

References:

1. https://en.wikipedia.org/wiki/Key-value_database
2. https://www.dummies.com/programming/big-data/nosql/in-memory-key-value-stores-in-nosql-databases/