|  |  |
| --- | --- |
| A picture of a winding road and trees  TCP  TCP key – value store | Nikhil Badyal  Internet Technology |

**TCP**

**Transmission Control Protocol (TCP)** – a connection-oriented communications [protocol](https://www.sdxcentral.com/resources/glossary/protocol/) that facilitates the exchange of messages between computing devices in a [network](https://www.sdxcentral.com/resources/glossary/network/). It is the most common protocol in networks that use the [Internet Protocol (IP)](https://www.sdxcentral.com/resources/glossary/internet-protocol-ip/); together they are sometimes referred to as TCP/IP.

TCP takes messages from an [application](https://www.sdxcentral.com/resources/glossary/anti-virus/)/[server](https://www.sdxcentral.com/resources/glossary/server/) and divides them into [packets](https://www.sdxcentral.com/resources/glossary/packet/), which can then be forwarded by the devices in the network – [switches](https://www.sdxcentral.com/resources/glossary/switch/), [routers](https://www.sdxcentral.com/resources/glossary/routers/), security gateways – to the destination. TCP numbers each packet and reassembles them prior to handing them off to the application/server recipient. Because it is connection-oriented, it ensures a connection is established and maintained until the exchange between the application/servers sending and receiving the message is complete.

**Assignment 1**

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 ”

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.

**Solution**

APPROACH

The whole program can be broadly divided into 3 parts

* Client
* Server
* Models

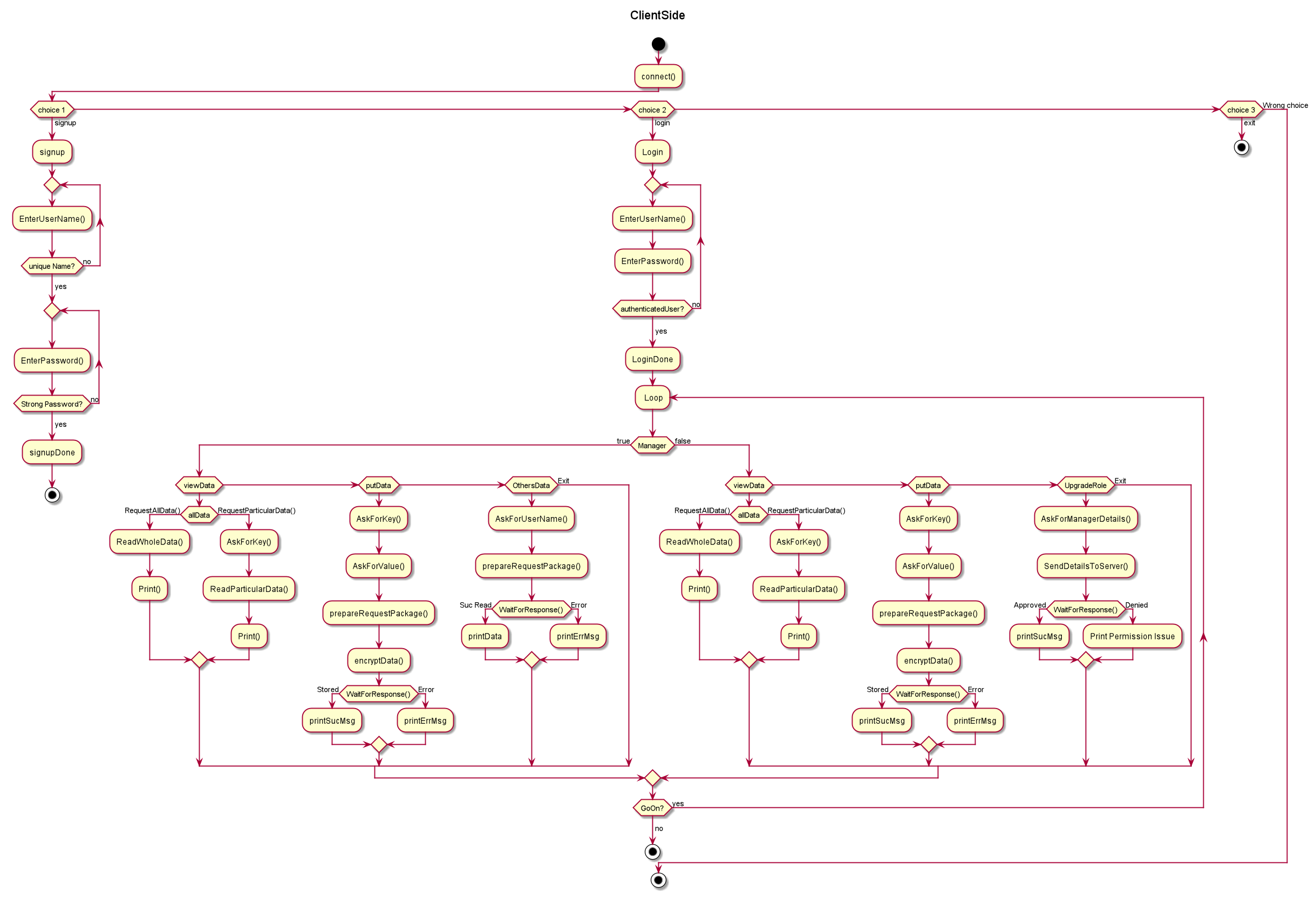
**Clients** – Client consist of all those parts to which users interact. Client side usually runs on a port provided by the connection otherwise the program can choose a default port 5555. Client can be run on any device but for assignment purpose it is running currently on localhost

**Server** – Server consist of all those parts which server needs to serve the client. Server can interact with database to read and write user data to the database.

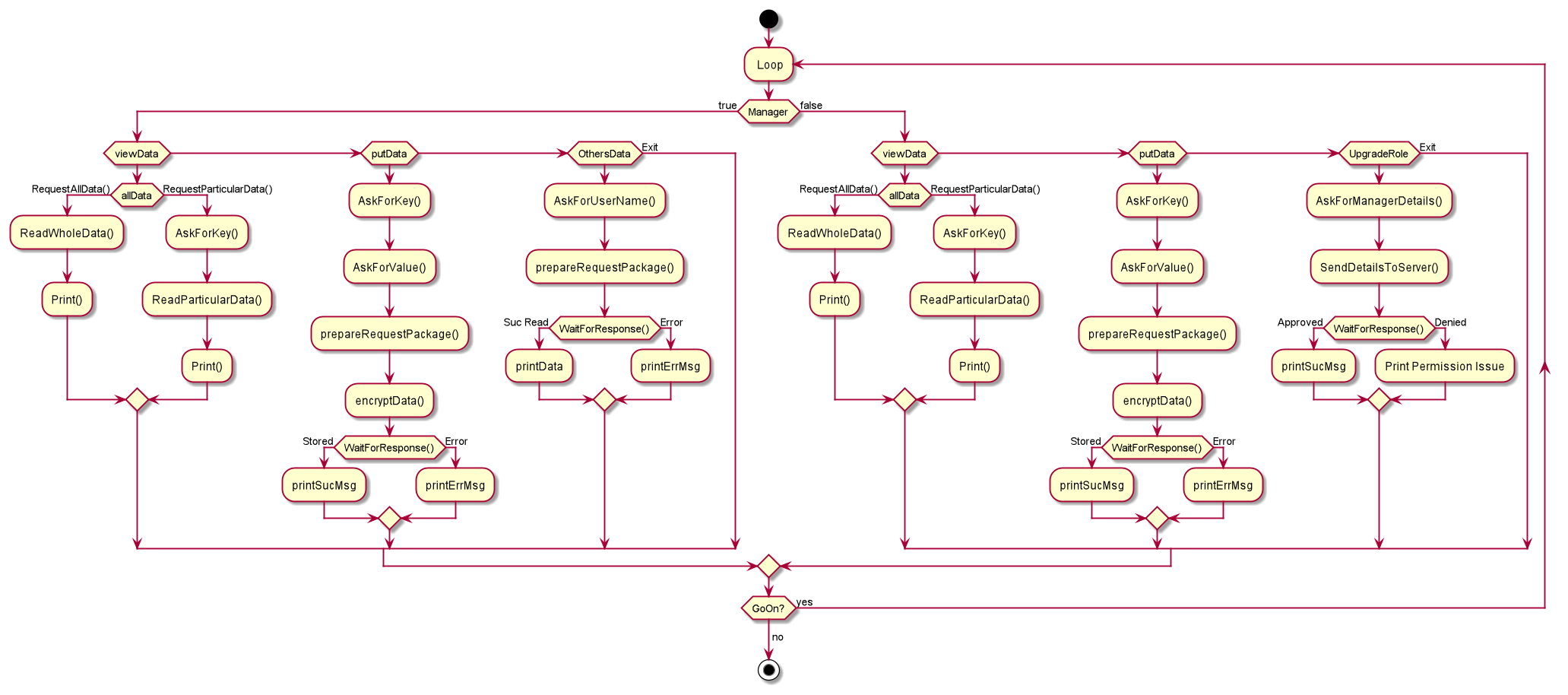
**Common Object**  – Serve as a common object or Protocol which both client and server use to communicate with each other.

**Database** – This class represent my database which contains the data of all the client.

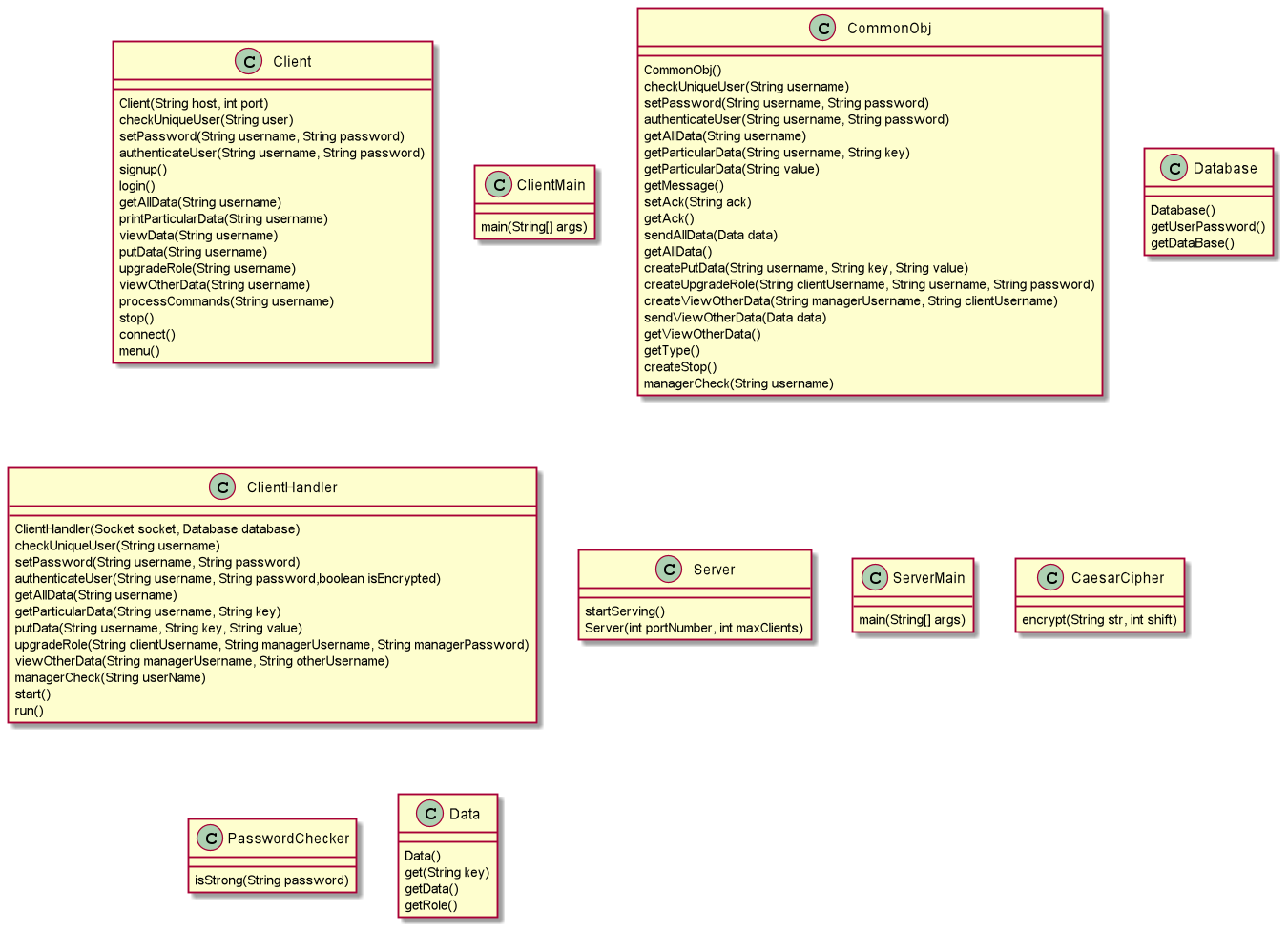
**MODELS**

**CLIENT**

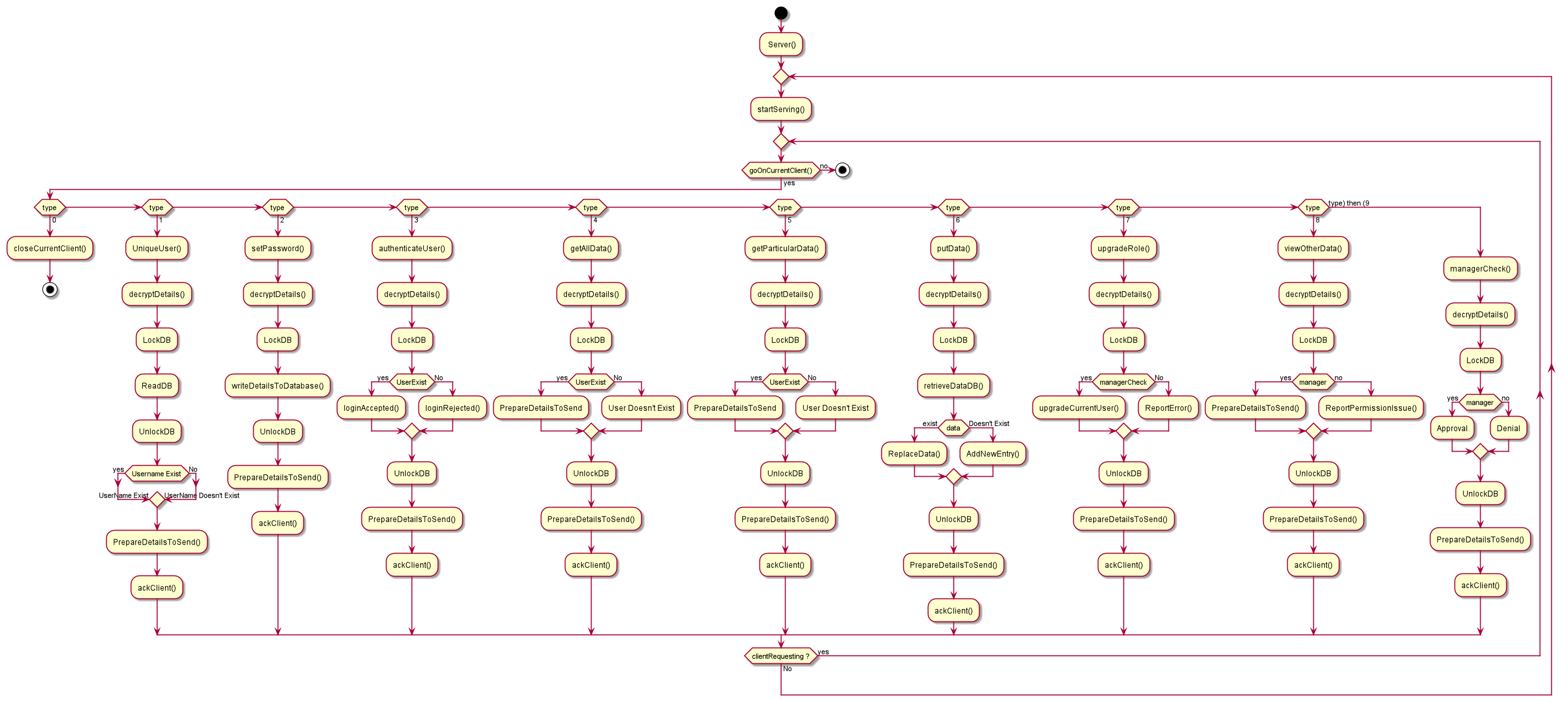
**CLIENT REQUESTS**



**SERVER CONFIG**

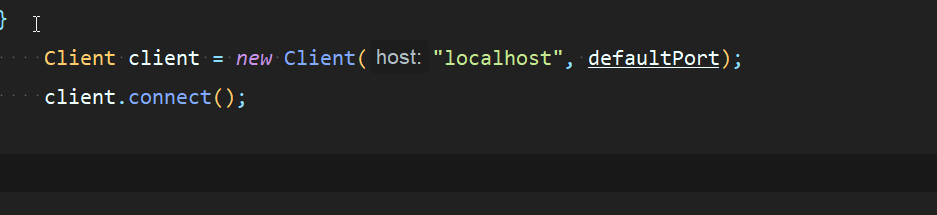


**SERVER ACTIVITES**



**SOCKET CONNECTION**

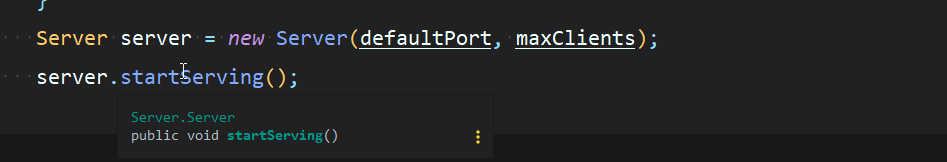
Every time a Client is connected. A new Client Object is made.



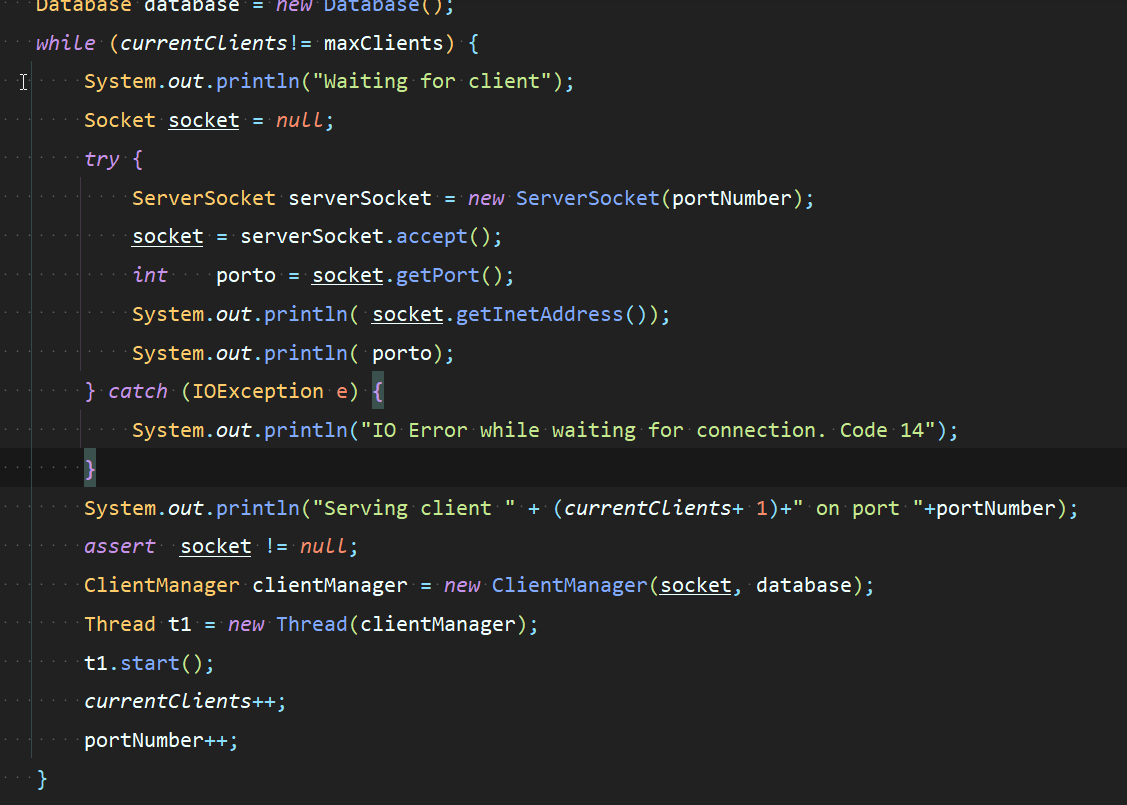
This is handler which handle client and present a menu to the client.



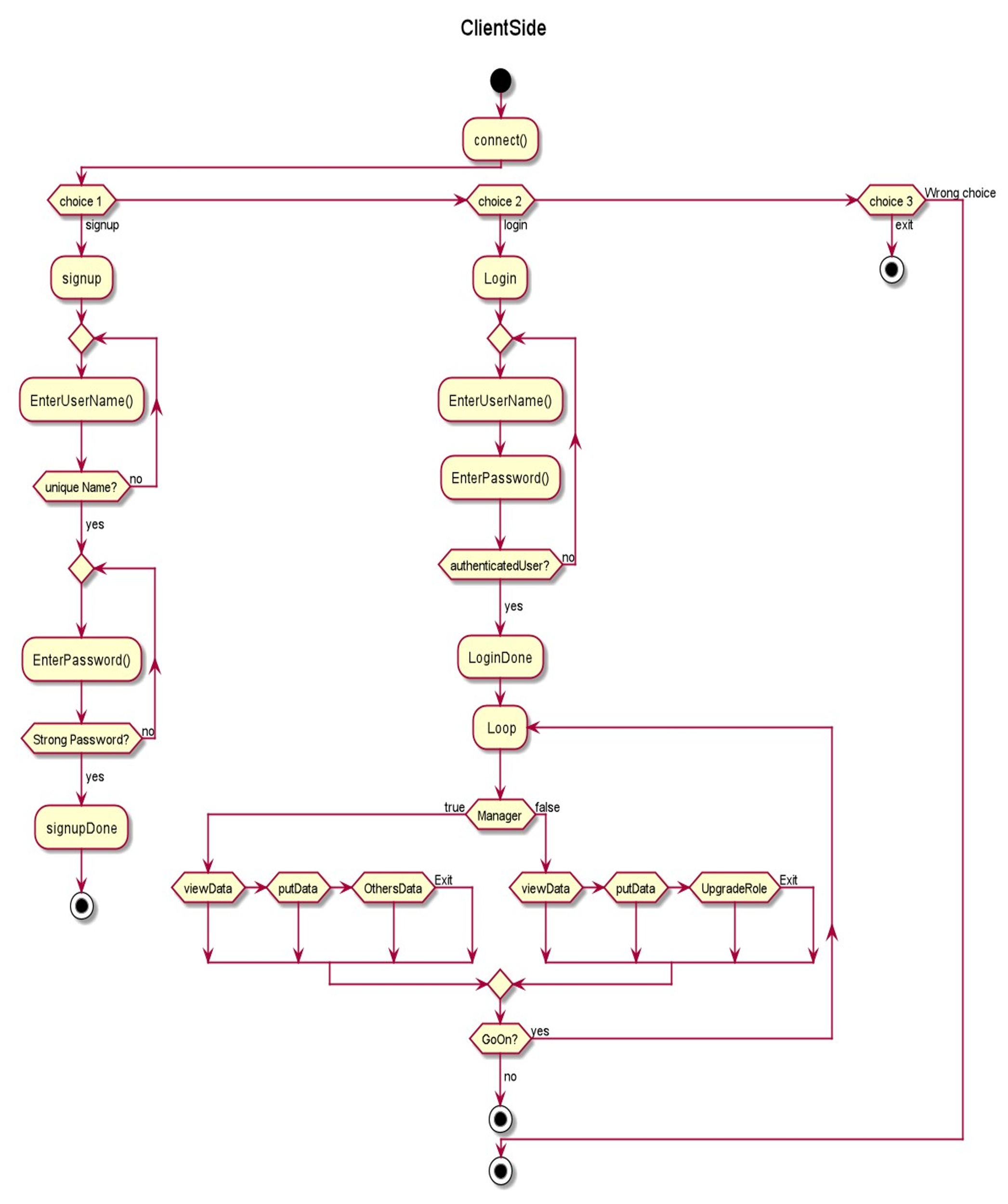
As soon as we turn on server. We create a single server object and it starts serving ASAP.



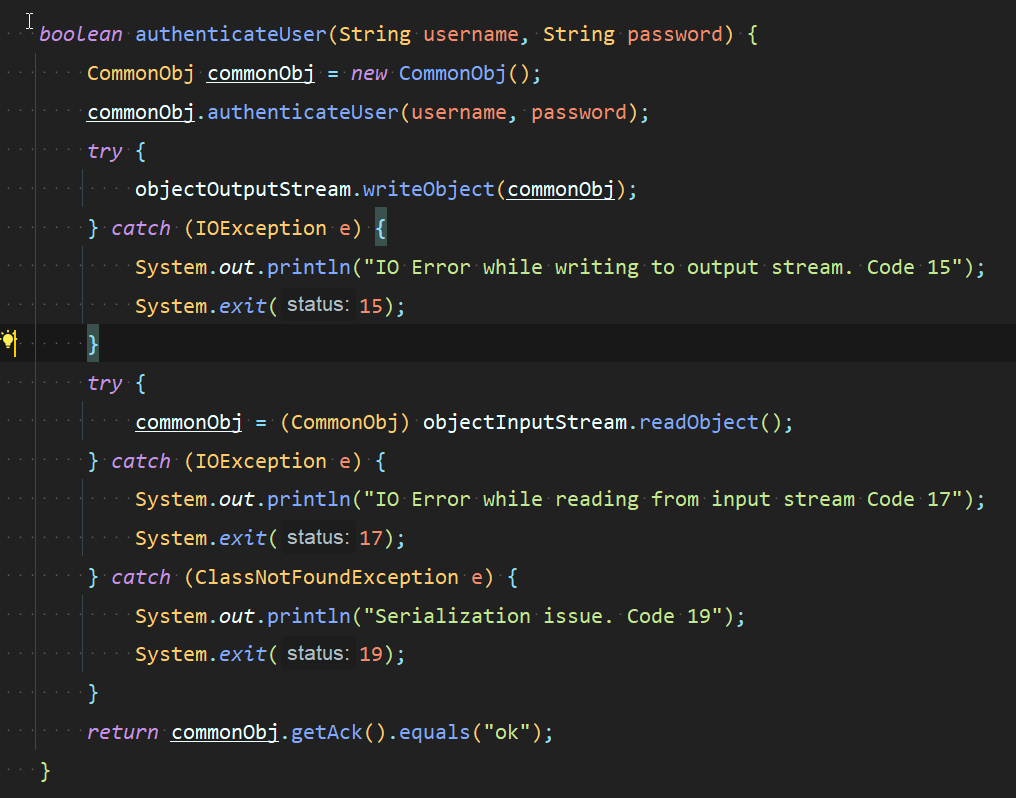
Server then waits for new client as soon client request for connection. Server dispatches it to a new thread.



**Client Login Handler**

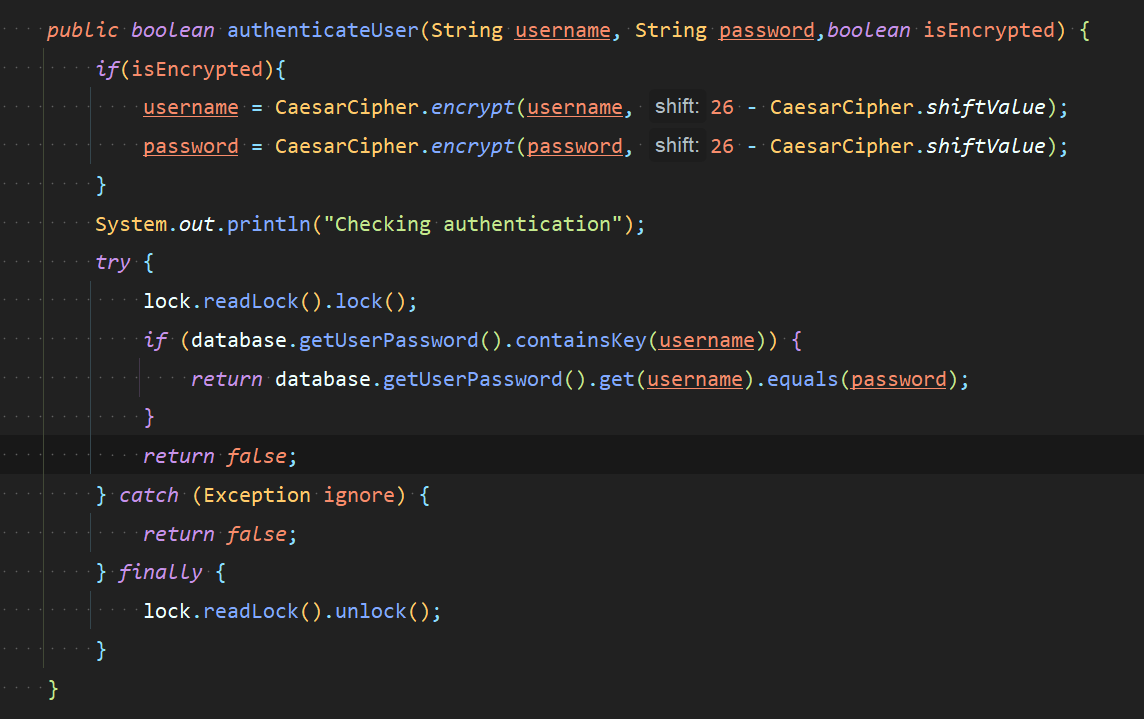


For Auth client makes an Auth Package. Which is then written to the objectOutputStream.

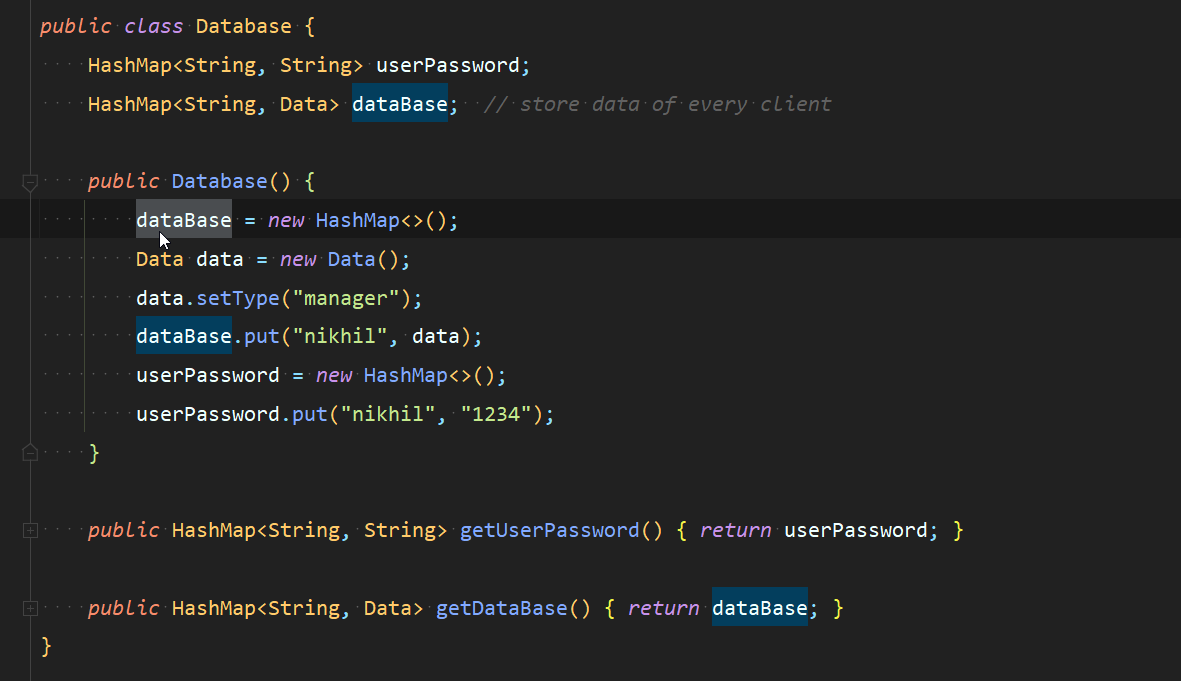


**SERVER**

Server – As server receives the details. The server first checks whether data is encrypted. If it is he decrypt it first and read the database for Auth.



**Database**



**CLIENT MANAGER**

*package* Server.ClientManager;  
  
*import* Models.CommonObj;  
*import* Models.Data;  
*import* Models.Database;  
*import* Server.Server;  
*import* Util.CaesarCipher;  
*import* java.io.IOException;  
*import* java.io.ObjectInputStream;  
*import* java.io.ObjectOutputStream;  
*import* java.net.Socket;  
*import* java.util.HashMap;  
*import* java.util.concurrent.locks.ReentrantReadWriteLock;  
  
*public class* ClientManager *implements Runnable* {  
 ReentrantReadWriteLock lock = *new* ReentrantReadWriteLock();  
 Socket socket;  
 Database database;  
 ObjectOutputStream objectOutputStream;  
 ObjectInputStream objectInputStream;  
 *boolean* donaldTrump = *false*;  
  
 *public* ClientManager(Socket socket, Database database) {  
 *this*.socket = socket;  
 *this*.database = database;  
 *try* {  
 objectOutputStream = *new* ObjectOutputStream(socket.getOutputStream());  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while creating output Stream. Code 10");  
*// System.exit(10);* }  
 *try* {  
 objectInputStream = *new* ObjectInputStream(socket.getInputStream());  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while creating input Stream. Code 11");  
*// System.exit(11);* }  
 }  
  
 *public boolean* checkUniqueUser(String username) {  
 username = CaesarCipher.*encrypt*(username, 26 - CaesarCipher.*shiftValue*);  
 System.*out*.println("Checking user " + username);  
 *boolean* bool;  
 *try* {  
 lock.readLock().lock();  
 bool = !database.getUserPassword().containsKey(username);  
 } *finally* {  
 lock.readLock().unlock();  
 }  
 System.*out*.println(bool);  
 *return* bool;  
 }  
  
 *public boolean* setPassword(String username, String password) {  
 username = CaesarCipher.*encrypt*(username, 26 - CaesarCipher.*shiftValue*);  
 password = CaesarCipher.*encrypt*(password, 26 - CaesarCipher.*shiftValue*);  
 System.*out*.println("Setting password");  
 *try* {  
 lock.writeLock().lock();  
 database.getUserPassword().put(username, password);  
 database.getDataBase().put(username, *new* Data());  
 *return true*;  
 } *catch* (Exception e) {  
 *return false*;  
 } *finally* {  
 lock.writeLock().unlock();  
 }  
 }  
  
 *public boolean* authenticateUser(String username, String password,*boolean* isEncrypted) {  
 *if*(isEncrypted){  
 username = CaesarCipher.*encrypt*(username, 26 - CaesarCipher.*shiftValue*);  
 password = CaesarCipher.*encrypt*(password, 26 - CaesarCipher.*shiftValue*);  
 }  
 System.*out*.println("Checking authentication");  
 *try* {  
 lock.readLock().lock();  
 *if* (database.getUserPassword().containsKey(username)) {  
 *return* database.getUserPassword().get(username).equals(password);  
 }  
 *return false*;  
 } *catch* (Exception ignore) {  
 *return false*;  
 } *finally* {  
 lock.readLock().unlock();  
 }  
 }  
  
 *public void* getAllData(String username) {  
 username = CaesarCipher.*encrypt*(username, 26 - CaesarCipher.*shiftValue*);  
 System.*out*.println("Request for printing all data of (" + username + ")");  
 CommonObj commonObj = *new* CommonObj();  
 *boolean* isPresent;  
  
 *try* {  
 lock.readLock().lock();  
 isPresent = database.getDataBase().containsKey(username);  
 } *finally* {  
 lock.readLock().unlock();  
 }  
  
 *if* (!isPresent) {  
 System.*out*.println("Username (" + username + ") does not exist");  
 commonObj.setAck("notOk");  
 } *else* {  
 System.*out*.println("Sending ack(ok)");  
 commonObj.setAck("ok");  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 donaldTrump = *true*;  
 *return*;  
*// System.exit(15);* }  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
 *return*;  
*// System.exit(16);* }  
 System.*out*.println("Sending all data of username (" + username + ")");  
 commonObj.sendAllData(database.getDataBase().get(username));  
 }  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 donaldTrump = *true*;  
 *return*;  
*// System.exit(15);* }  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
*// System.exit(16);* }  
 }  
  
 *public void* getParticularData(String username, String key) {  
 username = CaesarCipher.*encrypt*(username, 26 - CaesarCipher.*shiftValue*);  
 key = CaesarCipher.*encrypt*(key, 26 - CaesarCipher.*shiftValue*);  
 System.*out*.println("Request for printing a particular data of (" + username + ")");  
 CommonObj commonObj = *new* CommonObj();  
 *boolean* isPresent;  
 *try* {  
 lock.readLock().lock();  
 isPresent = database.getDataBase().containsKey(username);  
 } *finally* {  
 lock.readLock().unlock();  
 }  
  
 *if* (!isPresent) {  
 System.*out*.println("Username (" + username + ") does not exist");  
 commonObj.setAck("notOk");  
 } *else* {  
 System.*out*.println("Sending ack(ok)");  
 commonObj.setAck("ok");  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writting to output stream. Code 15");  
 donaldTrump = *true*;  
 *return*;  
*// System.exit(15);* }  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
 *return*;  
*// System.exit(16);* }  
 System.*out*.println("Sending value of (" + key + ")" + "of username(" + username + ")");  
 HashMap<String, String> data;  
 *try* {  
 lock.readLock().lock();  
 data = database.getDataBase().get(username).getData();  
 } *finally* {  
 lock.readLock().unlock();  
 }  
 commonObj.getParticularData(data.getOrDefault(key, "null"));  
 }  
 System.*out*.println(commonObj.getPrintParticularMessage());  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 donaldTrump = *true*;  
 *return*;  
*// System.exit(15);* }  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
*// System.exit(16);* }  
 }  
  
 *public void* putData(String username, String key, String value) {  
 username = CaesarCipher.*encrypt*(username, 26 - CaesarCipher.*shiftValue*);  
 key = CaesarCipher.*encrypt*(key, 26 - CaesarCipher.*shiftValue*);  
 value = CaesarCipher.*encrypt*(value, 26 - CaesarCipher.*shiftValue*);  
 System.*out*.println("Request for putting data "+ key +" "+ value + "with username (" + username + ")");  
 HashMap<String, String> data;  
 *boolean* isPresent;  
 *try* {  
 lock.readLock().lock();  
 data = database.getDataBase().get(username).getData();  
 isPresent = data.containsKey(key);  
 } *finally* {  
 lock.readLock().unlock();  
 }  
 *if* (isPresent) {  
 System.*out*.println("replaced");  
 data.replace(key, value);  
 } *else* {  
 System.*out*.println("added");  
 data.put(key, value);  
 }  
  
 CommonObj commonObj = *new* CommonObj();  
 commonObj.setAck("ok");  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 donaldTrump = *true*;  
 *return*;  
*// System.exit(15);* }  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
*// System.exit(16);* }  
 }  
  
 *public void* upgradeRole(String clientUsername, String managerUsername, String managerPassword) {  
 clientUsername = CaesarCipher.*encrypt*(clientUsername, 26 - CaesarCipher.*shiftValue*);  
 managerUsername = CaesarCipher.*encrypt*(managerUsername, 26 - CaesarCipher.*shiftValue*);  
 managerPassword = CaesarCipher.*encrypt*(managerPassword, 26 - CaesarCipher.*shiftValue*);  
 System.*out*.println("Request to upgrade role of (" + clientUsername + ")");  
 CommonObj commonObj = *new* CommonObj();  
  
 *boolean* isPresent;  
 *try* {  
 lock.readLock().lock();  
 isPresent = database.getDataBase().containsKey(managerUsername);  
 } *finally* {  
 lock.readLock().unlock();  
 }  
 *if* (isPresent) {  
 *boolean* isManager;  
 *try* {  
 lock.readLock().lock();  
 isManager = database.getDataBase().get(managerUsername).getRole().equals("manager");  
 } *finally* {  
 lock.readLock().unlock();  
 }  
  
 *if* (isManager) {  
  
 *boolean* auth;  
 *try* {  
 lock.readLock().lock();  
 auth = authenticateUser(managerUsername, managerPassword,*false*);  
 } *finally* {  
 lock.readLock().unlock();  
 }  
 *if* (auth) {  
 System.*out*.println("Upgrading client (" + clientUsername + ") to manager");  
 database.getDataBase().get(clientUsername).setType("manager");  
 commonObj.setAck("ok");  
 } *else* {  
 System.*out*.println("Wrong username or password of manager");  
 commonObj.setAck("Wrong username or password");  
 }  
 } *else* {  
 commonObj.setAck("(" + managerUsername + ") has no authority to upgrade any user");  
 System.*out*.println(managerUsername + "has no authority to upgrade any user");  
 }  
 } *else* {  
 commonObj.setAck("Manager does not exist");  
 System.*out*.println("Manager with username (" + managerUsername + ") does not exist");  
 }  
 System.*out*.println("writing " + commonObj.getAck());  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 donaldTrump = *true*;  
 *return*;  
*// System.exit(15);* }  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
*// System.exit(16);* }  
 }  
  
 *public void* viewOtherData(String managerUsername, String otherUsername) {  
 managerUsername = CaesarCipher.*encrypt*(managerUsername, 26 - CaesarCipher.*shiftValue*);  
 otherUsername = CaesarCipher.*encrypt*(otherUsername, 26 - CaesarCipher.*shiftValue*);  
 System.*out*.println("Request for views data of (" + otherUsername + ") by (" + managerUsername + ")");  
 CommonObj commonObj = *new* CommonObj();  
  
 *boolean* isManager;  
 *try* {  
 lock.readLock().lock();  
 isManager = database.getDataBase().get(managerUsername).getRole().equals("manager");  
 } *finally* {  
 lock.readLock().unlock();  
 }  
  
 *if* (isManager) {  
 *boolean* isPresent;  
 *try* {  
 lock.readLock().lock();  
 isPresent = database.getDataBase().containsKey(otherUsername);  
 } *finally* {  
 lock.readLock().unlock();  
 }  
 *if* (isPresent) {  
 commonObj.setAck("ok");  
 System.*out*.println("writing " + commonObj.getAck());  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 donaldTrump = *true*;  
*// System.exit(15);* }  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
*// System.exit(16);* }  
 Data data;  
 *try* {  
 lock.readLock().lock();  
 data = database.getDataBase().get(otherUsername);  
 } *finally* {  
 lock.readLock().unlock();  
 }  
 commonObj.sendViewOtherData(data);  
 System.*out*.println("Sending data of (" + otherUsername + ") to (" + managerUsername + ")");  
 } *else* {  
 commonObj.setAck("Client (" + otherUsername + ") does not exist");  
 System.*out*.println("writing " + commonObj.getAck());  
 }  
 } *else* {  
 commonObj.setAck("(" + managerUsername + ")has not authority");  
 System.*out*.println("writing " + commonObj.getAck());  
 }  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 donaldTrump = *true*;  
*// System.exit(15);* }  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
*// System.exit(16);* }  
 }  
  
 *public void* managerCheck(String userName) {  
 userName = CaesarCipher.*encrypt*(userName, 26 - CaesarCipher.*shiftValue*);  
 System.*out*.println("Request for manager check");  
 CommonObj commonObj = *new* CommonObj();  
 Data data = database.getDataBase().get(userName);  
 *if* (data.getRole().equals("manager")) {  
 commonObj.setAck("ok");  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
*// System.exit(16);* }  
 } *else* {  
 commonObj.setAck("notOk");  
 System.*out*.println("writing " + commonObj.getAck());  
 }  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 donaldTrump = *true*;  
*// System.exit(15);* }  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
*// System.exit(16);* }  
 }  
  
  
 *public void* start() {  
 *while* (!donaldTrump) {  
 CommonObj commonObj = *null*;  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while reading from input stream. Code 17");  
 donaldTrump = *true*;  
 *return*;  
*// System.exit(17);* } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 donaldTrump = *true*;  
*// System.exit(19);* }  
 *assert* commonObj != *null*;  
 *switch* (commonObj.getType()) {  
 *case* 0 -> donaldTrump = *true*;  
 *case* 1 -> {  
 *if* (checkUniqueUser(commonObj.getMessage().get(0))) {  
 commonObj.setAck("ok");  
 } *else* {  
 commonObj.setAck("notOk");  
 }  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 donaldTrump = *true*;  
*// System.exit(15);* }  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
*// System.exit(16);* }  
 }  
 *case* 2 -> {  
 *if* (setPassword(commonObj.getMessage().get(0), commonObj.getMessage().get(1))) {  
 commonObj.setAck("ok");  
 } *else* {  
 commonObj.setAck("notOk");  
 }  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 donaldTrump = *true*;  
*// System.exit(15);* }  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
*// System.exit(16);* }  
 }  
 *case* 3 -> {  
 *if* (authenticateUser(commonObj.getMessage().get(0), commonObj.getMessage().get(1),*true*)) {  
 commonObj.setAck("ok");  
 } *else* {  
 commonObj.setAck("notOk");  
 }  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 donaldTrump = *true*;  
*// System.exit(15);* }  
 *try* {  
 objectOutputStream.reset();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while resetting the output stream. Code 16");  
 donaldTrump = *true*;  
*// System.exit(16);* }  
 }  
 *case* 4 -> getAllData(commonObj.getMessage().get(0));  
 *case* 5 -> getParticularData(commonObj.getMessage().get(0), commonObj.getMessage().get(1));  
 *case* 6 -> putData(commonObj.getMessage().get(0), commonObj.getMessage().get(1), commonObj.getMessage().get(2));  
 *case* 7 -> upgradeRole(commonObj.getMessage().get(0), commonObj.getMessage().get(1), commonObj.getMessage().get(2));  
 *case* 8 -> viewOtherData(commonObj.getMessage().get(0), commonObj.getMessage().get(1));  
 *case* 9 -> managerCheck(commonObj.getMessage().get(0));  
 }  
  
 }  
 }  
  
 @Override  
 *public void* run() {  
 *try*{  
 start();  
 }*finally*{  
 System.*out*.println("One client disconnected");  
 Server.*currentClients*--;  
 }  
  
 }  
}

**CLIENT**

*package* Client;  
  
*import* Models.CommonObj;  
*import* Models.Data;  
*import* Util.PasswordChecker;  
  
*import* java.io.IOException;  
*import* java.io.ObjectInputStream;  
*import* java.io.ObjectOutputStream;  
*import* java.net.Socket;  
*import* java.util.Scanner;  
  
*public class* Client {  
 Socket socket;  
 *int* port;  
 ObjectOutputStream objectOutputStream;  
 ObjectInputStream objectInputStream;  
  
 *public* Client(String host, *int* port) {  
 *try* {  
 *this*.port = port;  
 socket = *new* Socket(host, port);  
 System.*out*.println(socket.getInetAddress());  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while creating socket. Code 18");  
 System.*exit*(15);  
 }  
  
 *try* {  
 objectOutputStream = *new* ObjectOutputStream(socket.getOutputStream());  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while creating output Stream. Code 10");  
 System.*exit*(10);  
 }  
 *try* {  
 objectInputStream = *new* ObjectInputStream(socket.getInputStream());  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while creating input Stream. Code 11");  
 System.*exit*(11);  
 }  
 }  
  
 *public boolean* checkUniqueUser(String user) {  
 CommonObj commonObj = *new* CommonObj();  
 commonObj.checkUniqueUser(user);  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*exit*(10);  
 }  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 System.*exit*(11);  
  
 } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 System.*exit*(19);  
 }  
 *return* commonObj.getAck().equals("ok");  
 */\* }else{  
 System.out.println("Server dead");  
 return false;  
 }\*/* }  
  
 */\*private boolean isServerAlive() {  
 try {  
 return !(socket.getInputStream().read() == -1);  
 } catch (SocketException e) {  
 System.out.println("Server died because of you.");  
 System.exit(1);  
 } catch (IOException e) {  
 e.printStackTrace();  
 System.exit(1);  
 }  
 return false;  
 }\*/  
  
  
 public void* setPassword(String username, String password) {  
 CommonObj commonObj = *new* CommonObj();  
 commonObj.setPassword(username, password);  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 System.*exit*(15);  
 }  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while reading from input stream Code 17");  
 System.*exit*(17);  
 } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 System.*exit*(19);  
 }  
 *if* (!commonObj.getAck().equals("ok")) {  
 System.*out*.println("Ops ! Something occurred on our side. Please report.");  
 }  
 }  
  
 *boolean* authenticateUser(String username, String password) {  
 CommonObj commonObj = *new* CommonObj();  
 commonObj.authenticateUser(username, password);  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 System.*exit*(15);  
 }  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while reading from input stream Code 17");  
 System.*exit*(17);  
 } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 System.*exit*(19);  
 }  
 *return* commonObj.getAck().equals("ok");  
 }  
  
 *void* printPasswordConditions() {  
 System.*out*.println("""  
 Password must have\s  
 1. Upper Case  
 2. Special Char  
 3. Digits  
 4. Small case  
 5. Minimum length of 8  
 """);  
 }  
 *public void* signup() {  
 String username = getUsername();  
 *while* (!checkUniqueUser(username)) {  
 System.*out*.println("Username exists.Enter new username\n");  
 username = getUsername();  
 *if* (username.equals("-1")) {  
 *return*;  
 }  
 }  
 System.*out*.print("Enter password : \n");  
 printPasswordConditions();  
 String password = getPassword();  
 *while* (!PasswordChecker.*isStrong*(password)) {  
 System.*out*.println("Password not strong enough.Enter again\n");  
 password = getPassword();  
 }  
 setPassword(username, password);  
 System.*out*.println("Registered Successful\n");  
 }  
  
 String getUsername() {  
 System.*out*.print("Enter username : ");  
 Scanner scanner = *new* Scanner(System.*in*);  
 *return* scanner.nextLine();  
 }  
  
 String getPassword() {  
 System.*out*.print("Enter password : ");  
 Scanner scanner = *new* Scanner(System.*in*);  
 String password = scanner.nextLine();  
 *return* scanner.nextLine();  
 }  
  
 *public* String login() {  
 String username = getUsername();  
 String password = getPassword();  
 *if* (!authenticateUser(username, password)) {  
 System.*out*.println("Wrong username or password\n");  
 *return* "-1";  
 }  
 System.*out*.println("Successfully logged in\n");  
 *return* username;  
 }  
  
 *public void* getAllData(String username) {  
 CommonObj commonObj = *new* CommonObj();  
 commonObj.getAllData(username);  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 System.*exit*(15);  
 }  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while reading from input stream Code 17");  
 System.*exit*(17);  
 } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 System.*exit*(19);  
 }  
 *if* (commonObj.getAck().equals("ok")) {  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while reading from input stream Code 17");  
 System.*exit*(17);  
 } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 System.*exit*(19);  
 }  
 Data data = commonObj.getAllData();  
 System.*out*.println("Printing all data of User(" + username + ")");  
 *for* (String key : data.getData().keySet()) {  
 System.*out*.println(key + " - " + data.get(key));  
 }  
 System.*out*.println();  
 }  
 }  
  
 *public void* printParticularData(String username) {  
 System.*out*.println("Enter key : ");  
 Scanner scanner = *new* Scanner(System.*in*);  
 String key = scanner.nextLine();  
 CommonObj commonObj = *new* CommonObj();  
 commonObj.getParticularData(username, key);  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 System.*exit*(15);  
 }  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while reading from input stream Code 17");  
 System.*exit*(17);  
 } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 System.*exit*(19);  
 }  
 *if* (commonObj.getAck().equals("ok")) {  
 System.*out*.println("Printing a particular data of user(" + username + ") key(" + key + ")");  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while reading from input stream Code 17");  
 System.*exit*(17);  
 } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 System.*exit*(19);  
 }  
 System.*out*.println(commonObj.getPrintParticularMessage());  
 } *else* {  
 System.*out*.println("Wrong key");  
 }  
 }  
  
 *public void* viewData(String username) {  
 System.*out*.println();  
 Scanner scanner = *new* Scanner(System.*in*);  
 System.*out*.println("1. View all data ");  
 System.*out*.println("2. View a particular data");  
 System.*out*.print("Enter Choice : ");  
 *int* choice = scanner.nextInt();  
 *switch* (choice) {  
 *case* 1 -> getAllData(username);  
 *case* 2 -> printParticularData(username);  
 *default* -> System.*out*.println("wrong choice");  
 }  
 }  
  
 *public void* putData(String username) {  
 System.*out*.println();  
 System.*out*.println("Enter info");  
 System.*out*.print("Enter key : ");  
 Scanner scanner = *new* Scanner(System.*in*);  
 String key = scanner.nextLine();  
 System.*out*.print("Enter value : ");  
 String value = scanner.nextLine();  
  
 CommonObj commonObj = *new* CommonObj();  
 commonObj.createPutData(username, key, value);  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 System.*exit*(15);  
 }  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while reading from input stream Code 17");  
 System.*exit*(17);  
 } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 System.*exit*(19);  
 }  
 *if* (commonObj.getAck().equals("ok")) {  
 System.*out*.println("\nSuccessfully data stored key(" + key + ") -> value(" + value + ")\n");  
 } *else* {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 System.*exit*(15);  
 System.*out*.println("Data cannot be stored");  
 }  
 }  
  
 *public void* upgradeRole(String username) {  
 System.*out*.println();  
 System.*out*.print("Enter username of Manager : ");  
 Scanner scanner = *new* Scanner(System.*in*);  
 String usernameManager = scanner.nextLine();  
 System.*out*.print("Enter Password : ");  
 String password = scanner.nextLine();  
  
 CommonObj commonObj = *new* CommonObj();  
 commonObj.createUpgradeRole(username, usernameManager, password);  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 }  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while reading from input stream Code 17");  
 System.*exit*(17);  
 } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 System.*exit*(19);  
 }  
 *if* (commonObj.getAck().equals("ok")) {  
 System.*out*.println();  
 System.*out*.println("Client(" + username + ") has been promoted to Manager");  
 System.*out*.println();  
 } *else* {  
 System.*out*.println("\nOperation unsuccessful");  
 System.*out*.println(commonObj.getAck());  
 System.*out*.println();  
 }  
 }  
  
 *public void* viewOtherData(String username) {  
 System.*out*.println();  
 System.*out*.print("Enter the username : ");  
 Scanner scanner = *new* Scanner(System.*in*);  
 String otherUsername = scanner.nextLine();  
  
 CommonObj commonObj = *new* CommonObj();  
 commonObj.createViewOtherData(username, otherUsername);  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 System.*exit*(15);  
 }  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while reading from input stream Code 17");  
 System.*exit*(17);  
 } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 System.*exit*(19);  
 }  
 *if* (commonObj.getAck().equals("ok")) {  
 System.*out*.println();  
 System.*out*.println("Printing all data of User(" + otherUsername + ")");  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while reading from input stream Code 17");  
 System.*exit*(17);  
 } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 System.*exit*(19);  
 }  
 Data data = commonObj.getViewOtherData();  
 *for* (String key : data.getData().keySet()) {  
 System.*out*.println(key + " - " + data.get(key));  
 }  
 } *else* {  
 System.*out*.println(commonObj.getAck());  
 }  
 System.*out*.println();  
 }  
  
 *public void* processCommands(String username) {  
 *boolean* goOn = *true*;  
 *while* (goOn) {  
 CommonObj commonObj = *new* CommonObj();  
 commonObj.managerCheck(username);  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 System.*exit*(15);  
 }  
 *try* {  
 commonObj = (CommonObj) objectInputStream.readObject();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while reading from input stream Code 17");  
 System.*exit*(17);  
 } *catch* (ClassNotFoundException e) {  
 System.*out*.println("Serialization issue. Code 19");  
 System.*exit*(19);  
 }  
  
 System.*out*.println("1. View data");  
 System.*out*.println("2. Put data");  
 *if* (commonObj.getAck().equals("ok")) {  
 System.*out*.println("3. View other's data");  
 System.*out*.println("4. Exit ");  
 Scanner scanner = *new* Scanner(System.*in*);  
 *int* choice = scanner.nextInt();  
 *switch* (choice) {  
 *case* 1 -> viewData(username);  
 *case* 2 -> putData(username);  
 *case* 3 -> viewOtherData(username);  
 *case* 4 -> goOn = *false*;  
 *default* -> System.*out*.println("Wrong choice");  
 }  
 } *else* {  
 System.*out*.println("3. Upgrade role");  
 System.*out*.println("4. Exit ");  
 Scanner scanner = *new* Scanner(System.*in*);  
 *int* choice = scanner.nextInt();  
 *switch* (choice) {  
 *case* 1 -> viewData(username);  
 *case* 2 -> putData(username);  
 *case* 3 -> upgradeRole(username);  
 *case* 4 -> goOn = *false*;  
 *default* -> System.*out*.println("Wrong choice");  
 }  
 }  
 }  
 }  
  
 *public void* stop() {  
 CommonObj commonObj = *new* CommonObj();  
 commonObj.createStop();  
 *try* {  
 objectOutputStream.writeObject(commonObj);  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while writing to output stream. Code 15");  
 System.*exit*(15);  
 }  
 }  
  
 *public void* connect() {  
 *boolean* end = *false*;  
 *while* (!end) {  
 String choice = menu();  
 *switch* (choice) {  
 *case* "1" -> signup();  
 *case* "2" -> {  
 String username = login();  
 *if* (!username.equals("-1")) {  
 processCommands(username);  
 }  
 }  
 *case* "3" -> {  
 stop();  
 end = *true*;  
 }  
 *default* -> System.*out*.println("Wrong choice");  
 }  
  
 }  
 *try* {  
 socket.close();  
 } *catch* (IOException e) {  
 System.*out*.println("IO Error while closing socket. Code 19");  
 System.*exit*(19);  
 }  
 }  
  
 *public* String menu() {  
 System.*out*.println("1. Sign Up");  
 System.*out*.println("2. login");  
 System.*out*.println("3. Exit");  
 Scanner scanner = *new* Scanner(System.*in*);  
 *return* scanner.nextLine();  
 }  
}