202135574 전시현 (소프트웨어 전공)

Execute Server class

Execute client1 class

Then server read the host and port number from the file.

Make a bank object.

Account class includes

Bank class has 5 accounts then set the 5 accounts through setnew()

Get into while loop{

Waiting for the client to contact

if it contacted, then it makes a ClientHandler object with the socket and bank info

make a new Thread

then execute the thread:

{

In clientHandler, it read line from the socket

Decides to call check/deposit/withdraw/transfer or execution message

If it decides to call check/deposit/withdraw/transfer, then call it in the bank

Gets string return of it

Write out message to socket

}

}

Account class

It has String firstname and balance. It has a method of returning firstname and a method of returning balance.

Bank class

It has 5 names of String, 5 Account objects and 5 methods.

setnew method : set the account objects with certain values

check method : check the customer’s balance and return the result

it find the customer’s object with getName(). If the object was found, it saves the balance and their name with certain protocol and change it into toUpperCase

if the object is not found, it shows exception with certain protocol

then, return it

deposit method : deposit given amount into the customer’s balance and return the result

it find the customer’s object with getName(). If the object was found, it changes the balance and saves their name and balance with certain protocol and change it into toUpperCase

if the object is not found, it shows exception with certain protocol

then, return it

withdraw method : withdraw given amount into the customer’s balance and return the result

it find the customer’s object with getName(). If the object was found, it changes the balance and saves their name and balance with certain protocol and change it into toUpperCase

if the object is not found, it shows exception with certain protocol

then, return it

transfer method : transfer given amount from the customer1’s balance into the customer2’s balance and return the result

it find the 2 customers’ objects with getName(). If bot of the object was found, it changes two balances and saves their names and balances with certain protocol and change it into toUpperCase

if one or two object is not found, it shows exception with certain protocol

then, return it

Client1 class

1.It gets host address and port number from the file

2.Get into while loop (stay in loop to wait for the message)

3.create client socket and connect to server

create output stream attacked to socket

create input stream attacked to socket

read from user

send line to server

read line from server

and shows the user the result(if the command was wrong or customer is not found, it shows you exception)

then close the socket

go back to 3

ClientHandler.java

It has private final Socket client;

private final Bank bank;

when calling ClientHandler with a client and bank then it saves into client and bank

run method

create input stream, attached to socket

read in line from the socket

counts the number of ' ' in clientSentence(given sentence) for future exception

split given sentence by " "

determines if they should causes Exception message or call method check/deposit/withdraw/transfer with bank object

then it does what was decided above

it makes the sentence to upper case and write out capitalizedSentence to socket

Server.java

read ip and port number from given file

then make a bank object and call setnew()

welcomeSocket was created above already, but assign new object into it with port number

while(true) loop

wait, on welcoming socket for contact by client

make a ClientHandler object

create a thread

start execution of the thread

protocol for command

CHECK "name"

DEPOSIT "name" amount

WITHDRAW "name" amount

TRANSFER "name1" amount "name2"

Protocol for Response

If amount is not an integer :

Exception: put an integer

If arguments are lack :

Exception: lack of arguments

If arguments are more than expected:

Exception: Wrong command put DEPOSIT "NAME" AMOUNT only

Exception: Wrong command put CHECK "NAME" only

Exception: Wrong command put WITHDRAW "NAME" AMOUNT only

Exception: wrong command : put TRANSFER "name1" amount "name2”

If command is right

* Check : “c0” +customer+ “ Account found" + ' ' + "Balance: " + customer’s balance
* Deposit : "c1 "+customer+" Accept" +' ' + "Balance: " + customer’s balance;
* Withdraw
  + When the balance goes down under 0 :

Exception: The balance doesn't go down under 0

When the balance doesn’t go down under 0

"c2 "+customer+" Accept" +' ' + "Balance: " + customer’s balance

* Transfer
  + When the balance goes down under 0:
    - Exception: The balnce can't go down under 0
  + When the balance doesn’t go down under 0
    - "C3 "+ cus1’s firstname+ "Accept " + cus1’s getbalance()+" " + cus2’s firstname+" Accept" + "Balance: " + cus’s balance();
* When the customer is not found :
  + EXCEPTION: name is not found for Check, Deposit, Withdraw method
  + Transfer : “Exception: No such info "+ customer + " is not found” for one customer

“Exception: No such info " + customer1 + ", "+ customer2 + " is not found” for two customers

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

import java.io.\*;

import java.net.\*;

public class Server {

static ServerSocket welcomeSocket;

public static void main(String[] args) throws Exception {

Server s = new Server();

String host = "";

int port = 0;

BufferedReader br = new BufferedReader(new FileReader("./serverinfo.txt")); //read ip and port number

while(true) {

String line = br.readLine();

if (line==null) {

break;}

String[] twowords = line.split(" "); //split by ' '

host = twowords[0]; //ip number

port = Integer.parseInt(twowords[1]); //port number

}

Bank bank = new Bank();

bank.setnew();

welcomeSocket = new ServerSocket(port); //welcomeSocket at port

while(true) {

Socket client = welcomeSocket.accept(); //wait, on welcoming socket for contact by client

ClientHandler handler = new ClientHandler(client, bank); //make a ClientHandler object

Thread thread = new Thread(handler); //create a thread

thread.start();//start execution of the thread

}

}

}

import java.io.BufferedReader;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.UncheckedIOException;

import java.net.Socket;

public class ClientHandler implements Runnable{

private final Socket client;

private final Bank bank;

public ClientHandler(Socket client, Bank bank) {

this.client = client; //save client into this.client

this.bank = bank; //save bank into this.bank

}

@Override

public void run() {

try {

BufferedReader inFromClient = null;

DataOutputStream outToClient = null;

String clientSentence = null;

String capitalizedSentence = null;

inFromClient= new BufferedReader(new

InputStreamReader(client.getInputStream())); //create input stream, attached to socket

clientSentence= inFromClient.readLine(); //read in line from the socket

char findChar= ' ';

int chCount = 0;

for(int i=0; i<clientSentence.length(); i++) {

if(clientSentence.charAt(i) == findChar) {

chCount++;} //counts the number of ' ' in clientSentence

}

if(chCount == 0) {//no ' ' in the clientSentence

capitalizedSentence = "Exception: Wrong command"; //has Exception

}

outToClient= new DataOutputStream(client.getOutputStream()); //created output stream, attached to socket

String[] parts = clientSentence.split(" "); //split by " "

String cmd = parts[0];

String nameOfUser = null;

if(chCount!=0) {

nameOfUser = parts[1].replaceAll("\"", ""); //remove all "s

}

String num;

if(cmd.contains(" ")) { //contains two " "

capitalizedSentence = "Eception: Wrong command - too much spaces\r\n"; //exception message

}

if(cmd.equalsIgnoreCase("CHECK")){ //when the word of the message is CHECK

if(chCount>1) {

capitalizedSentence = "Exception: Wrong command put CHECK \"NAME\" only"; //exception message

}

else {

if(nameOfUser != null && !nameOfUser.isEmpty()) { //if nameOfUser is not null and empty

capitalizedSentence = bank.check(nameOfUser); //call check function

}

else {

capitalizedSentence = "Exception: lack of arguments\r\n"; //exception message

}

}

}

else if (cmd.equalsIgnoreCase("DEPOSIT")) { //when the word of the message is DEPOSIT

if(chCount>2) {

capitalizedSentence = "Exception: Wrong command put DEPOSIT \"NAME\" AMOUNT only"; //exception message

}

else if (chCount==1) {

capitalizedSentence = "Exception: lack of arguments\r\n"; //exception message

}

else {

num = parts[2];

if(nameOfUser != null && !nameOfUser.isEmpty() && num!=null && !num.isEmpty()) {

try {

int temp = Integer.parseInt(num); //change num into int and save it into temp

capitalizedSentence = bank.deposit(nameOfUser, Integer.parseInt(num)); //call deposit function

}

catch(NumberFormatException e) {

capitalizedSentence = "Exception: put an integer"; //exception message

}

}

else {

capitalizedSentence = "Exception: lack of arguments\r\n"; //exception message

}

}

}

else if (cmd.equalsIgnoreCase("WITHDRAW")) { //when the word of the message is WITHDRAW

if(chCount>2) {

capitalizedSentence = "Exception: Wrong command put WITHDRAW \"NAME\" AMOUNT only"; //exception message

}

else if (chCount==1) {

capitalizedSentence = "Exception: lack of arguments\r\n"; //exception message

}

else {

num = parts[2];

if(nameOfUser != null && !nameOfUser.isEmpty() && num!=null && !num.isEmpty()) {

try {

int temp = Integer.parseInt(num); //change num into int and save it into temp

capitalizedSentence = bank.withdraw(nameOfUser, Integer.parseInt(num)); //call withdraw function

}

catch(NumberFormatException e) {

capitalizedSentence = "Exception: put an integer"; //exception message

}

}

else {

capitalizedSentence = "Exception: lack of arguments\r\n"; //exception message

}

}

}

else if (cmd.equalsIgnoreCase("TRANSFER")) { //when the word of the message is TRANSFER

if(chCount<=2) {

capitalizedSentence = "Exception: lack of arguments"; //exception message

}

else {

String part3 = parts[3].replaceAll("\"", ""); //remove all " in the word

num = parts[2];

if(nameOfUser != null && !nameOfUser.isEmpty() && num!=null && !num.isEmpty()&&part3 != null && !part3.isEmpty()) { //when nameOfUser, num and parts3 are not null and empty

try {

int temp = Integer.parseInt(num);

capitalizedSentence = bank.transfer(nameOfUser, Integer.parseInt(num), part3); //call transfer function

}

catch(NumberFormatException e) {

capitalizedSentence = "Exception: put an integer"; //exception message

}

}

else {

capitalizedSentence = "Exception: wrong command : put TRANSFER \"name1\" amount \"name2\"\r\n"; //exception message

}

}

}

else {

capitalizedSentence = "Exception: Too many arguments or wrong commands\r\n"; //exception message

}

capitalizedSentence = capitalizedSentence.toUpperCase() + "\r\n"; //change capitalizedSentence into upper cases

outToClient.writeBytes(capitalizedSentence); //write out capitalizedSentence to socket

}

catch (IOException e) {

throw new UncheckedIOException(e);

}

}

}

import java.io.BufferedReader;

import java.io.DataOutputStream;

import java.io.FileReader;

import java.io.InputStreamReader;

import java.net.\*;

public class Client1 {

static Socket clientSocket;

public static void main(String argv[]) throws Exception{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CHECK "name"

DEPOSIT "name" amount

WITHDRAW "name" amount

TRANSFER "name1" amount "name2"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));

//create input stream

String host = "";

int port = 0;

BufferedReader br = new BufferedReader(new FileReader("./serverinfo.txt")); //read serverinfo.txt

while(true) {

String hostport = br.readLine();

if (hostport==null) break; //nothing read

String[] linebuffer = hostport.split(" "); //split by " " and save in linebuffer

host = linebuffer[0]; //host has linebuffer[0]

port = Integer.parseInt(linebuffer[1]); //port has linebuffer[1] in int

}

// read host and port from the file

/\*stay in loop so that it sends given data by \*/

while(true) {

clientSocket = new Socket(host, port); //create client socket and connect to server

DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());

//create output stream attacked to socket

BufferedReader inFromServer = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

//create input stream attacked to socket

String sentence = inFromUser.readLine(); //read from user]

if(sentence.equalsIgnoreCase("q")) {

System.exit(0);

}

outToServer.writeBytes(sentence + "\n"); //send line to server

String modifiedSentence = inFromServer.readLine(); //read line from server

modifiedSentence = modifiedSentence.toString();

if(modifiedSentence.contains("EXCEPTION")) { //if modifiedSentence contains EXCEPTION

modifiedSentence = modifiedSentence.replaceAll("EXCEPTION: ", "Error message : ");

}

else if (modifiedSentence.contains("C1")) { //if modifiedSentence contains C1

modifiedSentence = modifiedSentence.replaceAll("C1 ", "");

modifiedSentence = modifiedSentence.replaceAll("ACCEPT", "NOW HAS ");

}

else if (modifiedSentence.contains("C0")) { //if modifiedSentence contains C0

modifiedSentence = modifiedSentence.replace("C0 ", "");

modifiedSentence = modifiedSentence.replace("ACCOUNT FOUND", "HAS ");

}

else if (modifiedSentence.contains("C2")) { //if modifiedSentence contains C2

modifiedSentence = modifiedSentence.replaceAll("C2 ", "");

modifiedSentence = modifiedSentence.replaceAll("ACCEPT", "NOW HAS ");

}

else if (modifiedSentence.contains("C3")) { //if modifiedSentence contains C3

modifiedSentence = modifiedSentence.replaceAll("C3 ", "");

modifiedSentence = modifiedSentence.replaceAll("ACCEPT", " NOW HAS ");

}

System.out.println(modifiedSentence); //shows the user modifiedSentence

clientSocket.close(); //close socket

}

}

}

public class Bank {

String[] name = new String[] {"Kim", "Choi", "Jo", "Lee", "Park"};

private Account accountObject[] = new Account[5];

public void setnew() {

accountObject[0] = new Account(name[0], 500);

accountObject[1] = new Account(name[1], 700);

accountObject[2] = new Account(name[2], 200);

accountObject[3] = new Account(name[3], 300);

accountObject[4] = new Account(name[4], 400); //made object array with Account class

}

public synchronized String check(String customer){

String clientSentence;

String capitalizedSentence = null;

/\*when account is found shows the balance information\*/

if(customer.equalsIgnoreCase(accountObject[0].getName())) { //exact match of the name is found

clientSentence = "c0 " +customer+ " Account found" + ' ' + "Balance: " + accountObject[0].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n'; //change the clientSentence to upper case

}

else if(customer.equalsIgnoreCase(accountObject[1].getName())) {//exact match of the name is found

clientSentence = "c0 "+customer+" Account found" + ' ' + "Balance: " + accountObject[1].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else if(customer.equalsIgnoreCase(accountObject[2].getName())) {//exact match of the name is found

clientSentence = "c0 "+customer+" Account found" + ' ' + "Balance: " + accountObject[2].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else if(customer.equalsIgnoreCase(accountObject[3].getName())) {//exact match of the name is found

clientSentence = "c0 "+customer+" Account found" + ' ' + "Balance: " + accountObject[3].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else if(customer.equalsIgnoreCase(accountObject[4].getName())) {//exact match of the name is found

clientSentence = "c0 "+customer+" Account found" + ' ' + "Balance: " + accountObject[4].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else {/\*when name is not found\*/

clientSentence = "EXCEPTION: name is not found"; //exception message

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

return capitalizedSentence;

}

public synchronized String deposit(String customer, int amount) {

String clientSentence;

String capitalizedSentence = null;

/\*when account is found shows the balance information after adding amount\*/

if(customer.equalsIgnoreCase(accountObject[0].getName())) {//exact match of the name is found

this.accountObject[0].balance+=amount;

clientSentence = "c1 "+customer+" Accept" +' ' + "Balance: " + accountObject[0].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n'; //change the clientSentence to upper case

}

else if(customer.equalsIgnoreCase(accountObject[1].getName())) {//exact match of the name is found

this.accountObject[1].balance+=amount;

clientSentence = "c1 "+customer+" Accept" +' ' + "Balance: " + accountObject[1].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else if(customer.equalsIgnoreCase(accountObject[2].getName())) {//exact match of the name is found

this.accountObject[2].balance+=amount;

clientSentence = "c1 "+customer+ " Accept" +' ' + "Balance: " + accountObject[2].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else if(customer.equalsIgnoreCase(accountObject[3].getName())) {//exact match of the name is found

this.accountObject[3].balance+=amount;

clientSentence = "c1 "+customer+" Accept" +' ' + "Balance: " + accountObject[3].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else if(customer.equalsIgnoreCase(accountObject[4].getName())) {//exact match of the name is found

this.accountObject[4].balance+=amount;

clientSentence = "c1 "+customer+" Accept" +' ' + "Balance: " + accountObject[4].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else {/\*customer name is not found\*/

capitalizedSentence = "Exception: "+customer+" is not found"; //exception message

}

return capitalizedSentence; //return capitalizedSentence

}

public synchronized String withdraw(String customer, int amount) {

String clientSentence;

String capitalizedSentence = null;

/\*when account is found shows the balance information after withdrawing amount\*/

if(customer.equalsIgnoreCase(accountObject[0].getName())) {//exact match of the name is found

if(this.accountObject[0].balance-amount<0) { //amount can't be withdrawn

clientSentence = "Exception: The balnce can't go down under 0"; //exception message

capitalizedSentence= clientSentence.toUpperCase() + '\n'; //change the clientSentence to upper case

}

else {//amount can be withdrawn

this.accountObject[0].balance-=amount;

clientSentence = "c2 "+customer+" Accept" +' ' + "Balance: " + accountObject[0].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

}

else if(customer.equalsIgnoreCase(accountObject[1].getName())) {//exact match of the name is found

if(this.accountObject[1].balance-amount<0) { //amount can't be withdrawn

clientSentence = "Exception: The balnce can't go down under 0"; //exception message

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else {//amount can be withdrawn

this.accountObject[1].balance-=amount;

clientSentence = "c2 "+customer+" Accept" +' ' + "Balance: " + accountObject[1].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

}

else if(customer.equalsIgnoreCase(accountObject[2].getName())) {//exact match of the name is found

if(this.accountObject[2].balance-amount<0) { //amount can't be withdrawn

clientSentence = "Exception: The balnce can't go down under 0";

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else {//amount can be withdrawn

this.accountObject[2].balance-=amount;

clientSentence = "c2 "+customer+" Accept" +' ' + "Balance: " + accountObject[2].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

}

else if(customer.equalsIgnoreCase(accountObject[3].getName())) {//exact match of the name is found

if(this.accountObject[3].balance-amount<0) { //amount can't be withdrawn

clientSentence = "Exception: The balnce can't go down under 0";

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else {//amount can be withdrawn

this.accountObject[3].balance-=amount;

clientSentence = "c2 "+customer+" Accept" +' ' + "Balance: " + accountObject[3].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

}

else if(customer.equalsIgnoreCase(accountObject[4].getName())) {//exact match of the name is found

if(this.accountObject[4].balance-amount<0) { //amount can't be withdrawn

clientSentence = "Exception: The balnce can't go down under 0";

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else {//amount can be withdrawn

this.accountObject[4].balance-=amount;

clientSentence = "c2 "+customer+" Accept" +' ' + "Balance: " + accountObject[4].getbalance();

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

}

else {/\*customer name is not found\*/

clientSentence = "EXCEPTION: name is not found"; //exception message

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

return capitalizedSentence; //return capitalizedSentence

}

public synchronized String transfer(String customer1, int amount, String customer2) {

String clientSentence = null;

String capitalizedSentence = null;

System.***out***.println(customer1 + customer2);

int cus1 = -1;

int right = 0;

/\*find customer1's account object\*/

if(customer1.equalsIgnoreCase(customer2)){

capitalizedSentence = "Exception: You can't transfer to your account";

}

if(customer1.equalsIgnoreCase(accountObject[0].getName())) {

cus1 = 0;

}

else if(customer1.equalsIgnoreCase(accountObject[1].getName())) {

cus1 = 1;

}

else if(customer1.equalsIgnoreCase(accountObject[2].getName())) {

cus1 = 2;

}

else if(customer1.equalsIgnoreCase(accountObject[3].getName())) {

cus1 = 3;

}

else if(customer1.equalsIgnoreCase(accountObject[4].getName())) {

cus1 = 4;

}

int cus2 = -1;

/\*find customer2's account object\*/

if(customer2.equalsIgnoreCase(accountObject[0].getName())) {

cus2 = 0;

}

else if(customer2.equalsIgnoreCase(accountObject[1].getName())) {

cus2 = 1;

}

else if(customer2.equalsIgnoreCase(accountObject[2].getName())) {

cus2 = 2;

}

else if(customer2.equalsIgnoreCase(accountObject[3].getName())) {

cus2 = 3;

}

else if(customer2.equalsIgnoreCase(accountObject[4].getName())) {

cus2 = 4;

}

if (cus1 == -1 && cus2 == -1) { //none of them was found

right = -1;

clientSentence = "Exception: No such info " + customer1 + ", "+ customer2 + " is not found";

}

else if (cus1 == -1) {//only customer2 is found

right = -1;

clientSentence = "Exception: No such info " + customer1 + " is not found";

}

else if (cus2 == -1 ) {//only customer1 is found

right = -1;

clientSentence = "Exception: No such info "+ customer2 + " is not found";;

}

if (right == 0) {/\*found both of them. when account is found shows the balance information after computing\*/

if(this.accountObject[cus1].balance-amount<0) {

clientSentence = "Exception: The balnce can't go down under 0"; //exception message

capitalizedSentence= clientSentence.toUpperCase() + '\n';

}

else {/\*it's able to transfer customer1's balance into customer2's balance\*/

this.accountObject[cus1].balance-=amount;

this.accountObject[cus2].balance+=amount;

clientSentence = "C3 "+ accountObject[cus1].firstname+ "Accept " + accountObject[cus1].getbalance()+" " +accountObject[cus2].firstname+" Accept" + "Balance: "

+ " " + accountObject[cus2].getbalance();

}

}

capitalizedSentence= clientSentence.toUpperCase() + "\r\n"; //change clientSentence into upper cases

return capitalizedSentence;

}

}

public class Account {

public String firstname;

public int balance;

Account(String name, int initialValue){

this.firstname = name; //saves name to this.firstname

this.balance = initialValue; //saves modifiedSentence to this.balance

}

public String getName() {

return firstname;

}

public int getbalance() {

return balance;

}

}