THE BANK APP

2021-04-26

App Concept

The Bank App is a simulation of a banking system which allows working with bank accounts on behalf of the administrator (bank officer) and a regular bank client.

Two types of accounts

Checking Account

- Has no interest rate
- Unlimited free transactions
- Operations:

deposit, withdraw, transfer, pay bills

Saving Account

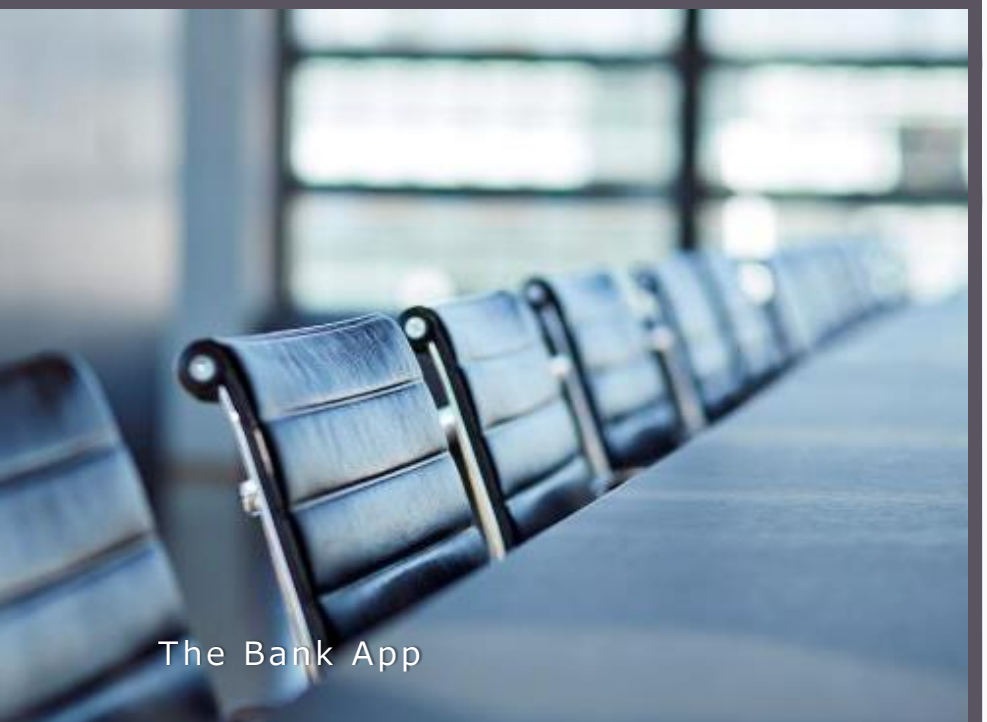
- Has interest rate
- Has limited free transactions
- Operations:

deposit, withdraw, transfer

- Free transactions between accounts of one clients
- Transaction fee for transactions over the limit

Some business rules

- Only bank officers can add new clients to the system and create accounts for them.
- Bank has only 2 types of accounts: saving and checking, but its properties (rate and number of free transactions) can be different.
- Once account is created, its properties can’t be changed (account number, rate, number of free transactions, date of opening).
- Only bank officers can delete (close) accounts and delete clients from the system.
- Clients can’t be deleted if they have an existing account.



Admin menu options

- Can register a new client in the system
- View information about clients and their accounts
- Change information about client: name and address
- Create accounts for clients
- Do operations with accounts:

deposit, withdraw, transfer

- Delete accounts from the system
- Delete clients from the system

Client menu options

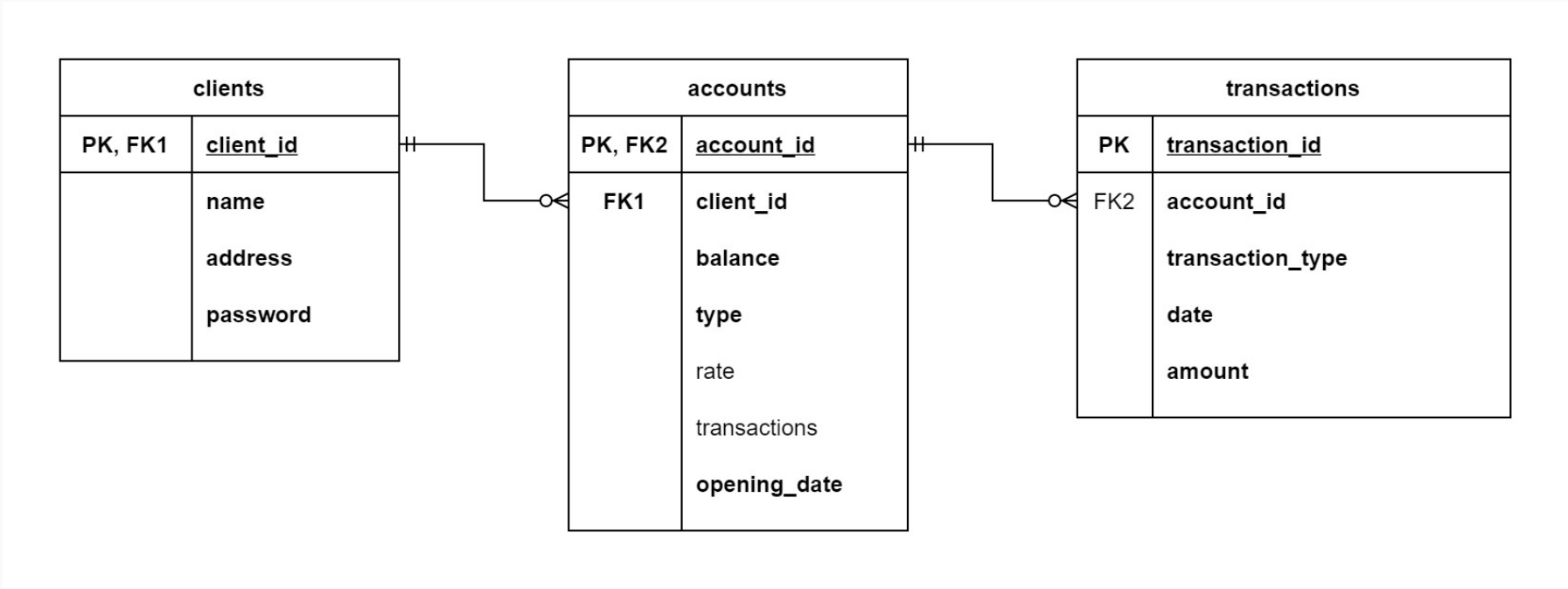
- View information about accounts
- Do operations with accounts:

withdraw, deposit, transfer, pay bills

- Change information about themselves: address and password
- Calculate future possible interest rate for saving accounts

Saving information about the clients

Information about clients and their accounts is saved to the MySQL database



Admin

clients : ArrayList<Client>
accounts : ArrayList<Account>

+ addClient(client : Client) : void
+ addAccount(account : Account) : void
+ findClient(name : string) : ArrayList<Client>
+ deleteAccount(account : Account) : void
+ deleteClient(client : Client) : void
+ showClients() : void
+ showAccounts() : void

Client

- clientID : int
- name : string
- address : string
- password : string
- accounts : ArrayList<Account>

+ Client(name : string, address : string) :
+ Client(id : int, name : string , address : string , password : string) :
+ getClientID() : int
+ setClientID(id : int) : void
+ getName() : string
+ setName(name : string) : void
+ getAddress() : string
+ setAddress(address : string) : void
+ getPassword() : string
+ setPassword(password : string) : void
+ addAccount(account : Account) : void
+ showAccounts() : void
+ selectAccount(input_no : int) : Account
+ deleteAccount(account : Account) : void
+ toString() : string
+ mainInfo() :string
- getRandomString : string

Account

accountNumber : int
accountHolder : int
balance : double
dateofOpening : LocalDate
- type : string

+ Account(accountNumber : int ,accountholder : int, balance : double):
+ setAccountnumber(id : int): void
+ getAccountNumber () : int
+ getAccountHolder() : int
+ getBalance() : double
+ getType() : string
+ getDateOfOpening() : LocalDate
+ deposit(amount : double) : void
+ withdraw(amount : double , statement : Statement) : void
+ transfer (amount : double , object : Account , statement :Statement) : void
+ toString() : string
+ mainInfo() : string

SavingAccount

- TYPE : string
- TRANSACTIONFEE : double
- rate : double
- numberOfFreeTransactions : int

+ SavingAccount(accountholder : int , balance : double, rate : double, numberOfFreeTransactions : int) :
+ SavingAccount(accountNumber : int , accountholder : int , balance : double, rate : double, numberOfFreeTransactions : int , dateOfopening : LocalDate) :
+ getType() : string
+ getTransactionFee() : double
+ getRate() : double
+ getNumberOfFreeTransactions() : int
+ applyFee(statement : Statement) : Boolean
+ withdraw(amount : double , statement : statement) : void
+ transfer(amount : double , object : Account , statement : Statement) : void
+ calculateInterest(months: int) : double
+ toString() : string

CheckingAccount

- TYPE : string

+ CheckingAccount(accountholder : int , balance : double) :
+ ChecingAccount(accountNumber : int , accountholder : int , balance : double , dateofopening : LocalDate) :
+ getType () : string
+ withdraw(amount : double, statement : Statement) : void
+ transfer(amount : double , object : Account , statement : Statement) : void
+ toString() : string

Admin class

addClient(Client client) – add the object **client** to the **clients** ArrayList

addAccount(Account account) – add the object **account** to the **accounts** ArrayList

findClient(String name) – search the clients by the name, put them in ArrayList<Client> and return ArrayList<Client> (it's easier to find by name, but may be several clients with the same name)

deleteAccount(Account account) – delete the object **account** from the **accounts** ArrayList

deleteClient(Client client) – delete the object **client** from the **clients** ArrayList

showClients() – display the number of clients and all the client's name

showAccounts() – display the number of accounts, list of all accounts with account_id, type and balance, the number of saving and checking accounts, and total balance

Client class

Each client has clientID, name, address, password, and accounts ArrayList

Client(String name, String address) – constructor for a new client
Client(Int id, String name, String address, String password) - constructor for getting client from the database
Getters: getClientID(), getName(), getAddress(), getPassword()
Setters: setClientID(Int id), setName(String name), setAddress(String address), setPassword(String password)
addAccount(Account account : Account) - add the object **account** to the **accounts** ArrayList
showAccounts() – displays info about client's accounts
selectAccount(Int number) – search the account by number and return Account object
deleteAccount(Account account) - delete the object **account** from the **accounts** ArrayList
toString() – display all info about the client including accounts info
mainInfo() - display username(client_ID), name and address of the client
getRandomString() – create random string (for creating password)

Account class – abstract class

Has fields: accountNumber, accountholder(client_ID), balance, date of opening, type

Account(Int accountNumber, int accountHolder, double balance) – constructor
Getters: getAccountNumber(), getAccountHolder(), getBalance(), getType(), getDateOfOpening()
Setters: setAccountnumber(int id)
deposit(double amount) – deposit money to the account
withdraw(double amount , Statement statement) – abstract method to inherit by child
transfer (double amount, Account account2 , Statement statement) – abstract method to inherit by child
mainInfo() – display account_id, type and balance of the account

SavingAccount class – inherit from Account

Has the fields: rate, number of free transactions, final type = “Saving” and final variable transactionFee

SavingAccount(int accountholder, double balance, double rate, int numberOfFreeTransactions) – constructor for a new saving account
SavingAccount(int accountNumber, int accountholder, double balance, double rate, int numberOfFreeTransactions, LocalDate dateOfopening) - constructor for getting saving account from the database
Getters: getType(), getRate, getNumberOfFreeTransactions(), getTransactionFee()
applyFee(statement statement) – check if the account has more or equals transactions than numberOfFreeTransactions, and if yes, return true
withdraw(double amount, Statement statement) – withdraw money from the account (with the fee if necessary)
transfer(double amount, Account account2 , Statemen statement) – transfer money to the account2 (with the fee if necessary)
calculateInterest(int months) – calculate possible future interest for the current balance
toString() – return the string with the account info

CheckingAccount class – inherit from Account

Has the fields: final type = “Checking”

CheckingAccount(int accountholder, double balance) – constructor for a new checking account
CheckingAccount(int accountNumber, int accountholder, double balance, LocalDate dateOfopening) - constructor for getting checking account from the database
Getters: getType()
applyFee(statement statement) – check if the account has more or equals transactions than numberOfFreeTransactions, and if yes, return true
withdraw(double amount, Statement statement) – withdraw money from the
transfer(double amount, Account account2 , Statemen statement) – transfer money to the account2
toString() – return the string with the account info

Service static classes

Menu

+ readCommand(prompt : string) : int
+ readData(prompt : string) : string
+ readValue(prompt : string) : double
+ readInt(prompt : string) : int
+ adminOptions() : int
+ clientOptions() : int
+ workWithClient() : int
+ changeClientInfo() : int
+ createAccount(): int
+ workWithAccount() : int
+ selectAccount() : int
+ addNewClient() : client

Sql

+ saveClient(statement : Statement , client : Client) : int
+ saveSaving (statement : Statement , account : SavingAccount) : int
+ saveChecking (statement : Statement , account : CheckingAccount) : int
+ updateName(statement : Statement , client : Client , newInfo : string) : void
+ updateAddress(statement : Statement , client : Client , newInfo : string) : void
+ updatePassword(statement : Statement , client : Client , newInfo : string) : void
+ deleteAccount(statement : Statement , account : Account) : void
+ deleteClient(statement : Statement , client : Client) : void
+ updateAccountBalance(statement : Statement , account : Account) : void
+ accountTransactionsInMonth(statement : Statement , account : Account) : int
+ addTransaction(statement : Statement , account : Account , amount : double , transaction : string) : void
+ deleteTransaction(statement : Statement, account : Account) : void
+ returnPassword(statement : Statement, username : String) : String

Menu class

readCommand(String prompt) – read input user, validate it, and return the command number
readData(String prompt) - read input user, validate it, and return string data
readValue(String prompt) – read input user, validate it, and return the double value
readInt(String prompt) – read input user, validate it, and return the int value

adminOptions() – display admin menu and return the command number
clientOptions() - display client menu and return the command number
workWithClient() – display admin menu for working with selected client and return command number
changeClientInfo() – display admin menu for changing client info and return command number
createAccount() – display the menu for creating account with the choice: saving account or checking account and return the command number
selectAccount() – display the menu with operations for the selected account and return the command number
addNewClient() – menu to enter the data for creating the new client

Sql class – methods for working with db

saveClient(Statement statement, Client client) – save client to db and return its clientID

saveSaving (Statement statement, SavingAccount account) – insert saving account to db and return its account number

saveChecking (Statement statement, CheckingAccount account) – insert checking account to db and return its account number

updateName(Statement statement, Client client, String newInfo) – change the client's name in db

updateAddress (Statement statement, Client client, String newInfo) – change the client's address in db

updatePassword (Statement statement, Client client, String newInfo) – change the client's password in db

deleteAccount(Statement statement, Account account) – delete the account from db

deleteClient(Statement statement, Client client) – delete the client from db

updateAccountBalance(Statement statement, Account account) – change the balance of the account in db

accountTransactionsInMonth(Statement statement, Account account) – count and return the number of account's transaction for the current month

addTransaction(Statement statement, Account account, double amount, String transaction) – insert transaction into db

deleteTransaction(Statement statement, Account account) – delete all transactions for the account in db

returnPassword(Statement statement, String username) – select and return the password for the username from db

Main class

Main
+ main() : void + adminMenu(statement : Statement) : void + workWithClient(admin : Admin, client : Client, statement : Statement) : void + clientMenu(statement : Statement, username : String) : void + OperationsInAccount(admin : Admin, account : Account, client : Client, statement : Statement) :void

adminMenu(Statement statement) – do-while loop implementing the full admin menu
workWithClient(Admin admin , Client client, Statement statement) – do-while loop implementing working with the particular client part of admin menu
clientMenu(Statement statement, String username) – do-while loop implementing the full client menu for the particular client
OperationsInAccount(Admin admin, Account account, Client clietnt, Statement statement) - do-while loop implementing the full working with account menu for the particular client

The Bank App user manual

for bank officers and clients

Client Menu

1. Login into the application with the client ID and password.

2. After you successfully login with the correct client ID and its password, you will see the client menu with the eight options

```
Welcome to the Bank App!
You are in the login menu. Type 0 in username field to quit.
Username: John Smith
Password: fgrthy56
```

```
Welcome to the Bank App!
You are in the login menu. Type 0 in username field to quit.
Username: 1
Password: demo
--------------------------
1) Display balance
2) Deposit
3) Withdraw
4) Transfer
5) Change Address
6) Change Password
7) Pay bills
8) Quit
--------------------------
Enter the command number:
```

Client menu

Option 1:

Display the account balances of the login client.

```
Enter the command number: 1
-------------------------------------
demo
Login: 1
Address: toronto
-------------------------------------
Accounts:
Account 1
Type: Saving
Date of opening: 2021-04-25
Rate: 2.0
FreeTransactions: 15
Balance: 1000.0
.....................................
Account 2
Type: Checking
Date of opening: 2021-04-25
Balance: 2000.0
.....................................
```

Option 2:

Choose an account of the current client,
and deposit some money into the account.

```
Enter the command number: 2
1  Saving 1000.0
2  Checking 2000.0
Enter the account number: 1
How much do you want to deposit: 10
-------------------------
```

You will see the account balances changed accordingly after your deposit.

```
Enter the command number: 1
-------------------------------------
demo
Login: 1
Address: toronto
-------------------------------------
Accounts:
Account 1
Type: Saving
Date of opening: 2021-04-25
Rate: 2.0
FreeTransactions: 15
Balance: 1010.0
.....................................
Account 2
Type: Checking
Date of opening: 2021-04-25
Balance: 2000.0
.....................................
```

```
------------------------
Enter the command number: 3
1  Saving 957.0
2  Checking 1950.0
Enter the account number: 1
How much do you want to withdraw: 10
You have done 11 transactions in this month.
Current transaction is free.
Type 1 to confirm 1
------------------------
```

```
Enter the command number: 3
1  Saving 947.0
2  Checking 1950.0
Enter the account number: 2
How much do you want to withdraw: 10
Transaction is free
Type 1 to confirm 1
------------------------
```

Client Menu

Option 3:

Withdraw money from one account of the current client.

If the account is Saving account, it will check if the free transaction times has been all used. If not, the transaction will be free.

If you are withdrawing money from a Checking account, then transaction will be free of charge.

Client Menu

Option 4:
Transfer money from one account to another account.

You have to choose one account from the current client to transfer money from for the receiving account.

As long as the account number exists in our system, it will work.

```
--------------------------
Enter the command number: 4
1  Saving 947.0
2  Checking 1940.0
Enter the account number you transfer money from: 2
Enter the receiving account number: 1
How much do you want to transfer: 50
Transaction is free
Type 1 to confirm 1
--------------------------
```

Client Menu

Option 5:

Update the address information of the current client.

Option 6:

Update the password of the current client.

After you changed the password, the next time you login for the same client, you need to use the new password.

```
--------------------------
Enter the command number: 5
Enter the new address: Shanghai
--------------------------
```

```
--------------------------
Enter the command number: 6
Enter the new password: demo1
--------------------------
```

```
Welcome to the Bank App!
You are in the login menu. Type 0 in username field to quit.
Username: 1
Password: demo1
------------------------
1) Display balance
2) Deposit
3) Withdraw
4) Transfer
5) Change Address
6) Change Password
7) Pay bills
8) Quit
------------------------
Enter the command number:
```

Client Menu

Option 7:

Pay bills, the client must choose a Checking account to pay bills.

```
--------------------------
Enter the command number: 7
1  Saving 997.0
2  Checking 1890.0
Choose the account number to pay the bill: 1
We suggest you use Checking account to pay your bills.
--------------------------
```

After the checking account is chosen, the client enter the bill number with the amount to pay. Then the bill be will paid. The relevant amount of money will be removed from the client's chosen account.

```
--------------------------
Enter the command number: 7
1  Saving 997.0
2  Checking 775.0
Choose the account number to pay the bill: 2
Please enter your bill number: 2209333
How much do you want to pay: 90
Transaction is free
Type 1 to confirm 1
Bill Number: 2209333 has been paid.
```

Client Menu

Option 8:

You can calculate your future possible interest for the saving account based on current balance

```
--------------------------
Enter the command number: 8
1  Saving 10070.0
Choose the account number to calculate interest rate: 1
How much months will you hold your deposit: 24
Your earned interest: 16.79032638888566
--------------------------
```

Option 9:

quit the client menu and go back to the login page.

Admin Menu

```
Username: admin
Password: aDmIn
```

```
1) Create new client
2) Work with client
3) Show all clients
4) Show all accounts
5) Quit
----------------------------
Enter the command number: 1
```

1. Login into the application with the special username and password.

Default: admin - aDmIn

2. After you successfully login, you will see the admin menu with 4 options

Admin Menu

Option 1:
Create new client, by entering the name and address of the new client.

The login client ID and its initial password will be generated by the application.

It is necessary for the first entry from the client side. The client can change the password after the first entry to the app.

```
--------------------------
1) Create new client
2) Work with client
3) Show all clients
4) Show all accounts
5) Quit
--------------------------
Enter the command number: 1
Enter name: demo2
Enter address: Toronto
Login: 6
Password: fgww6UqE
--------------------------
```

Admin menu

Option 2:
Work with the client searching by the name.

If the client's name you entered does not exist in the system, it will not proceed..

```
1) Create new client
2) Work with client
3) Show all clients
4) Show all accounts
5) Quit
-------------------------
Enter the command number: 2
Enter the client's name: 2
No such client
-------------------------
```

If you have several clients with the same name, the system displays all of them and offer to select

```
Enter the command number: 2
Enter the client's name: Maria S
1)
Maria S
Login: 100007
Address: 5 H Road

2)
Maria S
Login: 100008
Address: 45 Green Road

Enter the number of client: 1
```

Once you have entered the correct client's name, the system will show basic info of the client, and a menu to further work with the client.

```
-------------------------
Enter the command number: 2
Enter the client's name: demo

demo
Login: 1
Address: Shanghai

1  Saving 998.0
2  Checking 584.0
-------------------------
1) Show information
2) Change Information
3) Create new account
4) Work with account
5) Delete account
6) Delete client
7) Back
-------------------------
```

Admin menu

Option 2 – Sub menu (Option 1)

Show the detailed info of the client.

```
-------------------------
1) Show information
2) Change Information
3) Create new account
4) Work with account
5) Delete account
6) Delete client
7) Back
-------------------------
Enter the command number: 1
-------------------------------------
demo
Login: 1
Address: Shanghai
-------------------------------------
Accounts:
Account 1
Type: Saving
Date of opening: 2021-04-25
Rate: 2.0
FreeTransactions: 15
Balance: 998.0
.....................................
Account 2
Type: Checking
Date of opening: 2021-04-25
Balance: 584.0
.....................................
```

Option 2 – Sub menu (Option 2)

Change the name or address of the current client

```
--------------------------
Enter the command number: 2
--------------------------
1) Change name
2) Change address
3) Back
--------------------------
Enter the command number: 1
Enter the new name: demo

demo
Login: 1
Address: Shanghai

1  Saving 998.0
2  Checking 584.0
--------------------------
```

```
--------------------------
Enter the command number: 2
--------------------------
1) Change name
2) Change address
3) Back
--------------------------
Enter the command number: 2
Enter the new address: Toronto

demo
Login: 1
Address: Toronto

1  Saving 998.0
2  Checking 584.0
--------------------------
```

Admin Menu

Option 2 – Sub menu (Option 3):

Create accounts for the current client.

Option 2 – Sub menu (Option 4):

Choose one account of the current client and you can withdraw from/deposit to/transfer from this account.

Option 2 – Sub menu (Option 5):

Delete one account of the current client’s accounts, if the account number entered is wrong, no account will be changed.

```
--------------------------
Enter the command number: 3
--------------------------
1) Saving Account
2) Checking Account
3) Back
--------------------------
```

```
--------------------------
Enter the command number: 4
1  Saving 998.0
2  Checking 584.0
Enter the account number: 1
1  Saving 998.0
--------------------------
1) Withdraw
2) Deposit
3) Transfer
4) Back
--------------------------
```

```
--------------------------
Enter the command number: 5
1  Saving 998.0
2  Checking 584.0
Enter the account number: 3
No such account! Please try again.

demo
Login: 1
Address: Toronto

1  Saving 998.0
2  Checking 584.0
--------------------------
```

Admin Menu

Option 2 – Sub menu (Option 6)

Delete the current client.

```
-------------------------
1) Show information
2) Change Information
3) Create new account
4) Work with account
5) Delete account
6) Delete client
7) Back
-------------------------
Enter the command number: 6
Client deleted
-------------------------
```

Option 2 – Sub menu (Option 7)

Back to the previous menu

The possible errors and exceptions

and their resolutions

Failing connecting with database

If connecting with the database fails the app is going to crash.

```
Welcome to the Bank App!
Exception in thread "main" java.sql.SQLException Create breakpoint : Access denied for user 'yoot'@'localhost' (using password: YES)
    at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:129)
    at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:122)
    at com.mysql.cj.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:833)
    at com.mysql.cj.jdbc.ConnectionImpl.<init>(ConnectionImpl.java:453)
    at com.mysql.cj.jdbc.ConnectionImpl.getInstance(ConnectionImpl.java:246)
    at com.mysql.cj.jdbc.NonRegisteringDriver.connect(NonRegisteringDriver.java:198)
    at java.sql.DriverManager.getConnection(DriverManager.java:664)
    at java.sql.DriverManager.getConnection(DriverManager.java:247)
    at com.company.Main.main(Main.java:13)

Process finished with exit code 1
```

Failing connecting with database

We added try-catch block to handle this exception.

```
try {
    System.out.println("Welcome to the Bank App!");
    Class.forName("com.mysql.cj.jdbc.Driver").getDeclaredConstructor().newInstance();
    Connection conn = DriverManager.getConnection( url: "jdbc:mysql://localhost/bank_app",  user: "root",  password: "tatiana");
    Statement statement = conn.createStatement();


    do {...} while (true);

    conn.close();
    System.out.println("Thank you for using the Bank App");
}
catch (Exception e){
    System.out.println("Oops! Something went wrong. Please try again later.");
}
```

Now it doesn’t crash, but shows the error message.

Input Validation

```java
public static double readValue(String prompt){
    double value;
    do {
        System.out.print(prompt);
        while (!scanner.hasNextDouble()) {
            System.out.println("Invalid input! Please try again");
            scanner.next();
        }
        value = scanner.nextDouble();
    } while (value <=0);
    return value;
}

public static int readInt(String prompt){
    int value;
    do {
        System.out.print(prompt);
        while (!scanner.hasNextInt()) {
            System.out.println("Invalid input! Please try again");
            scanner.next();
        }
        value = scanner.nextInt();
    } while (value <=0);
    return value;
}
```

To avoid crashes related to invalid user's input, we include input validation to our methods.

For example, for double and int values we check if the input has the valid variable's type and ask for a positive number.

No sufficient fund condition

When creating withdraw method we needed to check the withdraw amount to continue the operation.

Firstly, we used if-else statement. However, it wasn’t the good decision, because it was crucial to stop the operation if there was no sufficient fund for it.

So we went for the throwing the exception in the method.

In the main class we handled this exception. The program shows the error message and ask the user to try again.

```java
@Override
public void withdraw(double amount, Statement statement) throws Exception {
    // check for fee
    if (applyFee(statement)){
        if (amount + transactionFee > balance){
            throw new Exception("No sufficient fund");
        }
        else{
                balance -= (amount + transactionFee);
            }
    }
    else {
        if (amount > balance){
            throw new Exception("No sufficient fund");
        } else{
                balance -= amount;
        }
    }
}
```

Admin menu

Option 3:

Show all clients in the system.

```
1) Create new client
2) Work with client
3) Show all clients
4) Show all accounts
5) Quit
--------------------------
Enter the command number: 3

Clients: 2

demo
demo2
--------------------------
```

Option 4:

Show all accounts in the system.

```
1) Create new client
2) Work with client
3) Show all clients
4) Show all accounts
5) Quit
--------------------------
Enter the command number: 4

Accounts: 2

1  Saving 998.0
2  Checking 584.0

Saving accounts: 1
Checking accounts: 1
Total balance: 1582.0
--------------------------
```

Option 5:

Quit the admin menu.

Screenshots of app working

without errors

WORKING AS ADMIN

```
Welcome to the Bank App!
You are in the login menu. Type 0 in username field to quit.
Username: admin
Password: aDmIn
-------------------------
1) Create new client
2) Work with client
3) Show all clients
4) Show all accounts
5) Quit
-------------------------
Enter the command number: 1
Enter name: Maria S
Enter address: 45 Green Road
Login: 100008
Password: IYgE6wyB
-------------------------
1) Create new client
2) Work with client
3) Show all clients
4) Show all accounts
5) Quit
-------------------------
```

```
1) Withdraw
2) Deposit
3) Transfer
4) Back
-------------------------
Enter the command number: 1
How much do you want to withdraw: 2000
Transaction is free
Type 1 to confirm 1
No sufficient fund
3  Saving 1500.0
-------------------------
1) Withdraw
2) Deposit
3) Transfer
4) Back
-------------------------
Enter the command number: 4

Maria S
Login: 100007
Address: 5 H Road

3  Saving 1500.0
```

```
Enter the command number: 3
-------------------------
1) Saving Account
2) Checking Account
3) Back
-------------------------
Enter the command number: 1
Enter start balance: 1500.00
Enter rate: 0.01
Enter the number of free transactions: 3
Saving account created

Maria S
Login: 100007
Address: 5 H Road

3  Saving 1500.0
-------------------------
```

```
Enter the command number: 2
Enter the client's name: Maria S
1)
Maria S
Login: 100007
Address: 5 H Road

2)
Maria S
Login: 100008
Address: 45 Green Road

Enter the number of client: 1

Maria S
Login: 100007
Address: 5 H Road
```

WORKING AS CLIENT

```
Username: 100007
Password: FRdfEbDY
--------------------------
1) Display balance
2) Deposit
3) Withdraw
4) Transfer
5) Change Address
6) Change Password
7) Pay bills
8) Calculate interest rate
9) Quit
--------------------------
Enter the command number: 1
---------------------------------------
Maria S
Login: 100007
Address: 5 H Road
---------------------------------------
Accounts:
Account 3
Type: Saving
Date of opening: 2021-04-25
Rate: 0.01
FreeTransactions: 3
Balance: 1500.0
.......................................
```

```
1) Display balance
2) Deposit
3) Withdraw
4) Transfer
5) Change Address
6) Change Password
7) Pay bills
8) Calculate interest rate
9) Quit
--------------------------
Enter the command number: 6
Enter the new password: qwerty
--------------------------
```

```
--------------------------
1) Display balance
2) Deposit
3) Withdraw
4) Transfer
5) Change Address
6) Change Password
7) Pay bills
8) Calculate interest rate
9) Quit
--------------------------
Enter the command number: 7
3  Saving 1500.0
Choose the account number to pay the bill: 3
We suggest you use Checking account to pay your bills.
--------------------------
```

Script for creating MySQL database

which we use in our app

```
CREATE TABLE `bank`.`clients` (
  `client_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `address` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`client_id`));

CREATE TABLE `bank`.`accounts` (
  `account_id` INT NOT NULL AUTO_INCREMENT,
  `client_id` INT NOT NULL,
  `balance` DECIMAL(9,2) NOT NULL DEFAULT 0.00,
  `type` INT NOT NULL,
  `rate` DECIMAL(5,2) NULL,
  `transactions` INT NULL,
  `opening_date` DATE NULL,
  PRIMARY KEY (`account_id`),
  INDEX `clients_accounts_idx` (`client_id` ASC) VISIBLE,
  CONSTRAINT `clients_accounts`
    FOREIGN KEY (`client_id`)
    REFERENCES `bank`.`clients` (`client_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

```
CREATE TABLE `bank`.`accounts` (
  `account_id` INT NOT NULL AUTO_INCREMENT,
  `client_id` INT NOT NULL,
  `balance` DECIMAL(9,2) NOT NULL DEFAULT 0.00,
  `type` INT NOT NULL,
  `rate` DECIMAL(5,2) NULL,
  `transactions` INT NULL,
  `opening_date` DATE NULL,
  PRIMARY KEY (`account_id`),
  INDEX `clients_accounts_idx` (`client_id` ASC) VISIBLE,
  CONSTRAINT `clients_accounts`
    FOREIGN KEY (`client_id`)
    REFERENCES `bank`.`clients` (`client_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

```
CREATE TABLE `bank`.`transactions` (
  `transaction_id` INT NOT NULL AUTO_INCREMENT,
  `account_id` INT NOT NULL,
  `transaction_type` VARCHAR(45) NOT NULL,
  `date` DATE NOT NULL,
  `amount` DECIMAL(9,2) NOT NULL,
  PRIMARY KEY (`transaction_id`),
  INDEX `transactions_accounts_idx` (`account_id` ASC)
VISIBLE,
  CONSTRAINT `transactions_accounts`
    FOREIGN KEY (`account_id`)
    REFERENCES `bank`.`accounts` (`account_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

Thank you for your attention!