|  |
| --- |
| **Assignment 4 - Group Assignment - Banking** |

# Deadline

December 1st 2019 00:00 (midnight)

# Group Assignment

* Teams of 3
* All teams must declare themselves and be approved by the teacher before the assignment can be started
* Each team will have 1 member responsible for code review
  + Each team should go through at 1 least code review with the teacher prior to final submission of the assignment

# Banking application specifications

This application allow the user to perform 1 of 7 tasks:

* [1] Add a new Client
* [2] Create a new Account
* [3] Make a Deposit
* [4] Make a Withdrawal
* [5] List Account Transactions
* [6] List Clients
* [7] List Client Accounts

# [1] Add a new Client

Adding a new Client requires that the process follows these guidelines:

1. Retrieve the client’s information from the console
2. Create the client
3. Add the client to the bank’s list of clients
4. **No output necessary**

# [2] Create a new Account

Creating a new Account requires that the process follows these guidelines:

1. Retrieve the client’s id from the console
   1. Any account must be created only for an existing user
2. Retrieve the type of account (checking or savings) desired from the console
3. Create the account
4. Add the account to the list of the client’s account
5. **No output necessary**

# [3] Make a Deposit

Making a new deposit requires that the process follows these guidelines:

1. Retrieve the client’s id from the console
2. Retrieve the desired account number from the console
   1. Any selected account must belong to the selected client
3. Retrieve the amount of the transaction from the console
4. Create the transaction
5. Add the transaction to the list of transactions for the selected account
6. Update the account’s balance
7. Output: **Savings(10): 500.00$**
   1. *The Account type, account number and balance*

# [4] Make a Withdrawal

Making a new withdrawal requires that the process follows these guidelines:

1. Retrieve the client’s id from the console
2. Retrieve the desired account number from the console
   1. Any selected account must belong to the selected client
3. Retrieve the amount of the transaction from the console
4. Create the transaction
5. Add the transaction to the list of transactions for the selected account
6. Update the account’s balance
7. Output: **Savings(10): 500.00$**
   1. *The Account type, account number and balance*

# [5] List Account Transactions

Listing all the account’s transactions requires that the process follows these guidelines:

1. Retrieve the client’s id from the console
2. Retrieve the desired account number from the console
3. Retrieve the desired account from the client’s account list
4. Display all the transactions listed in the account
5. Output: **Deposit of 500.00$**

**Withdrawal of 215.00$**

1. *Transaction type and amount*

**Savings (10): 285.00$**

* 1. *The Account type, account number and balance*

# [6] List Clients

Listing all the bank’s clients requires that the process follows these guidelines:

1. Retrieve all the clients from the bank’s client list
2. Display all the clients
3. Output: **List of current clients:**
   1. *Regular text*

**(1) Raphael, Ronald**

**(2) Raphael, Daniel**

* 1. *Client id, Last Name, First Name*

# [7] List Client Accounts

Listing all the client’s accounts requires that the process follows these guidelines:

1. Retrieve the client’s id from the console
2. Retrieve the selected client from the bank’s client list
3. Display all the accounts from the selected client
4. Output: **Accounts for Raphael, Daniel (2):**
   1. *Regular text followed by Last Name, First Name and client id*

**Savings(2): 1210.60$**

**Savings(3): 85.15$**

**Checking(7): -73.70$**

* 1. *Account type, account number, balance*

# Classes and properties:

## Banking

Also the name of the project

The class containing the main() method

Instantiates the **Bank** object and calls the methods to process the user requests

## Bank

Properties:

* String bankNumber
* address;
* ArrayList<> clientList;

## Client

* int id;
* firstName;
* lastName;
* ArrayList<> accountList;
* counter;

## Account

* int accountNumber;
* double balance;
* ArrayList<> transactions;
* type (checking or savings)
* owner;
* counter;

## CheckingAccount

## SavingsAccount

* NONE

## Transaction

* type (withdrawal or deposit)
* double amount;

## UserInputManager

* NONE

# Tips

## Account

This class should never be instantiated directly, always instantiate a **CheckingAccount** or a **SavingsAccount**

## CheckingAccount

## SavingsAccount

* When instantiating the object, simply set the account type correctly

# Files

A group of interfaces are provided in the *Distributed Documents* section of *Omnivox*. These interfaces contain the required functions and descriptions of these functions. The provided interfaces must be implemented by their matching classes.

The files that need to be handed in through Omnivox are:

* Classes.zip

Classes.zip must contain the following source code files:

* Account.java
* Bank.java
* Banking.java
* CheckingAccount.java
* Client.java
* SavingsAccount.java
* Transaction.java
* UserInputManager.java

# Evaluation Grid

|  |  |
| --- | --- |
| Code Review | 15 points |
| Code Testing | 60 points |
| Programming Code | 25 points |