**TECHNICAL UNIVERSITY OF CLUJ-NAPOCA**

**Homework 4:**

**Bank Simulation**

Student name: Nistor Rares

Group: 30425

**Content:**

1. **Introduction**

**1.1 Homework objective . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .page 3**

1. **Problem solving. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .page 3**
2. **Projection**

**3.1 UML diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .page 4**

**3.2 Data structures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . page 5**

**3.3 Class projection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .page 6**

**3.4 User interface. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .page 7**

**3.5 Packages . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . page 8**

**3.6 Algorithms . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . page 8**

1. **Implementation and testing. . . . . . . . . . . . . . . . . . . . . . . . .page 9**
2. **Results . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .page 10**
3. **Conclusions. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . page 10**
4. **Bibliography. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . page 11**
5. **Introduction**
   1. **Homework objective**

Define the interface BankProc (add/remove persons, add/remove holder associated accounts, read/write accounts data, report generators, etc). Specify the pre and post conditions for the interface methods.

2. Define and implement the classes Person, Account, SavingAccount and SpendingAccount. Other classes may be added as needed (give reasons for the new added classes).

3. An Observer DP will be defined and implemented. It will notify the account main holder about any account related operation.

4. Implement the class Bank using a predefined collection which uses a hashtable. The hashtable key will be generated based on the account main holder (ro. titularul contului). A person may act as main holder for many accounts. Use JTable to display Bank related information.

4.1 Define a method of type “well formed” for the class Bank.

4.2 Implement the class using Design by Contract method (involving pre, post conditions, invariants, and assertions). 5. Implement a test driver for the system. 6. The account data for populating the Bank object will be loaded/saved from/to a file.

2). **Problem solving**

I have solved this problem using HashTables and HashMaps . The ideea is that I have a private " Hashtable<String, LinkedList<Account>> accounts " .The key is a String variable, and the HashTable is a Linked List of objects of type account .The String variable is the key ,in my case this represents the CNP of a person . That means that when I try to add an account or to delete one in my Hash Table ,I mandatory need to use the CNP . Is the unique key that knows which account to delete .Why I can not delete using name or other item ?Because there can be 2 persons with the same name ,but with different CNP .

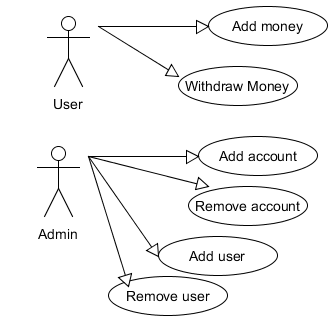
**3). Projection**

**UML diagram**

**Class diagram** is **UML** **structure diagram** which shows structure of the designed system at the level of **classes** and **interfaces**, shows their **features**, **constraints** and relationships - **associations**, **generalizations**, **dependencies**, etc.

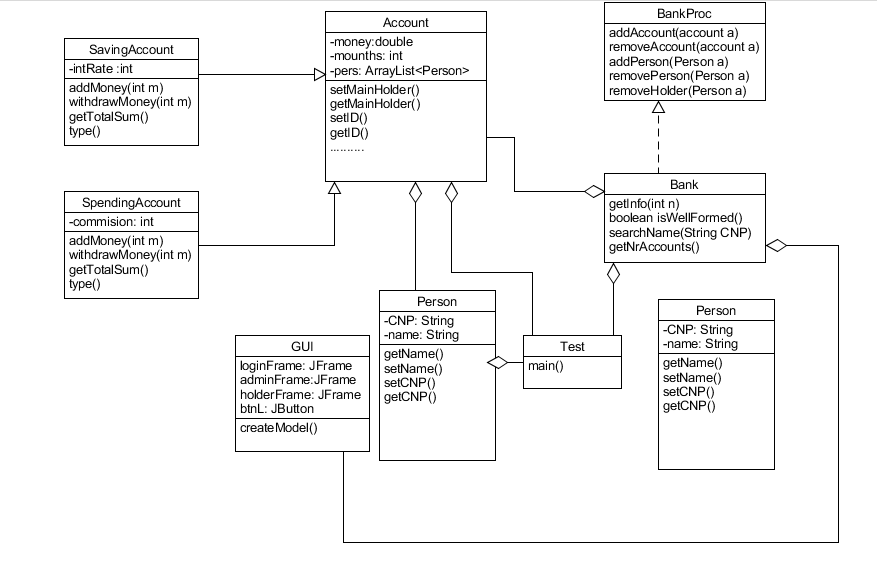
**Dependency** is often confused as Association. Dependency is normally created when you receive a reference to a class as part of a particular operation / method . **Association** is reference based relationship between two classes. Here a class A holds a class level reference to class B. Association can be represented by a line between these classes with an arrow indicating the navigation direction. In case arrow is on the both sides, association has bidirectional navigation.

**Use case diagram:**

****

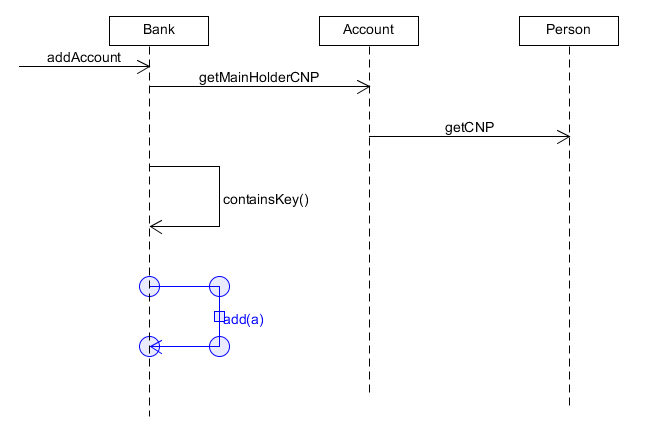
This shows what can an user do while he is trying to attempt to use this application .You can add an user ,remove an user and for a certain account to withdraw or add money to a specific account ( spending or saving ) .

Class diagram :



We have the class diagram that shows us all the classes and the relations between all of them. We have association between some classes ,and also the class Bank that implements the interface BankProc. In this homework , we have use the Design By Contracts . This means that we have to implement some methods in this interface, and also the class bank , that are called Pre and Post conditions. The main idea is that whoever you wish to do any operations on account or persons , first we check some conditions. For example, talking about adding an account in the list. First we check if we have that account already , then ,after we have added everything we need for an account , then you check ,as a Post condition , of the nr of accounts has increased by 1 . We have the method isWellFormed that checks this. All we had to do was to "assert" that method before trying to implement a new one.

Sequence diagram:

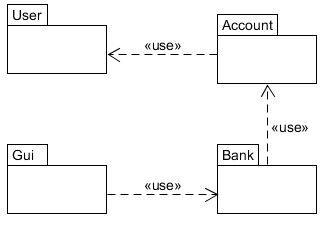


The sequence diagram shows how an account is added in the bank. When the addAccount method is called we call the getMainHolderCNP method to go to the class Account and then the getCNP method to get the CNP of the person. We werify if the key exists and add the account, as we mentioned before ,the key is the CNP .

Package diagram :

**Classes and packages:**

In my homework , I got 4 packages : account ,bank ,Gui, User.



The package acount I got 3 classes : one abstract class named "Account" and 2 other classes name "SavingAccount " and "SpendingAccount" .The abstract class refers to the ideea that we have 2 different tipes of accounts ,but they are not totally different.So ,in the Abstract class we defined the constructor and the methodes that both classes should implement . Even thow the "Deposit" methode and "Withdraw" are different ,we create them in the abstract class and then after the classes extends the abstract one , we must implement them different in both classes .

Bank package contains 2 classes : The interface BancProcessing and the class Bank . Here is where we can see the Design By Contracts and the method isWellFormed() . Also ,in the class bank we got the main operation on the bank accounts .

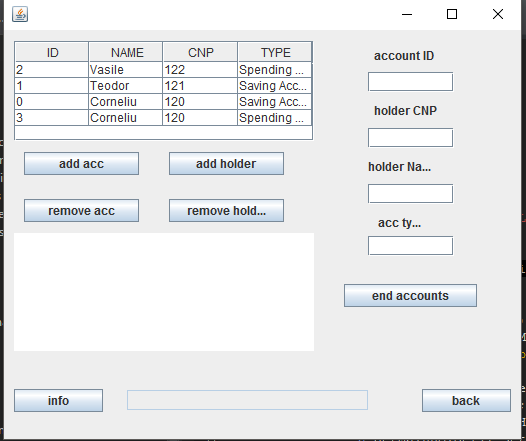
gui package contains only one class that is the Graphic User Interface and here we draw the interface and implements some methods for GUI .

The user package got also only one class named "Person" . Not a very complex one , there we simply declare some details about every person like Name ,CNP .

Implementation and Testing .

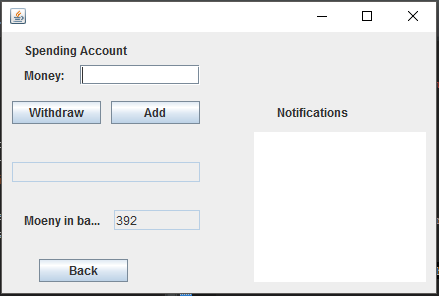
My GUI is not very complicated and easy to understand. I got 2 frames one for the user and one for the admin

In the admin frame:



I have information about the accounts and buttons to add, delete holders or accounts

In the user frame:



I have displayed the amount of money in the account and buttons to add or withdraw money.

**4). Results**

The result of this homework is a user friendly application that simulates a simple bank application that allows us to add acount and make operations like deposit / withdraw just like any other real bank application in real life .I don’t think there will be any problems understanding the output of this application.

**5). Conclusions**

In conclusion the homework helped me to better understand the notion of Lists , ArrayLists and also the main use of a HashTable and HashMap . I have learned by doing this homework why it is so important to understand and use the HashMaps and why they are net superior to Lists or any other implementation . Also ,I could understand the Design By Contracts technique that uses the Pre and Post conditions in order to make sure the program is working propertlly . I hope that this homework is a decent one because I know it’s far from perfect, or even good.

For example here are some change and improvements that I would make:

-trying to make more holders of an account , I mean , like in real life ,more persons to have access on an account but only one is the mainHolder .

-trying to make an account also unique ,by adding a certain ID to a specific account and ,when we try to work with them ,not only we check the CNP ,but from that CNP we see more accounts .

-also , adding more details to the JTable could be a little more helpfull because we dont know which type of account we have(when we try to add money or withdraw from an account ) .

**6). Bibliography**

For this homework I used different sites to help me understand concepts that I did not yet understood or to help me create different diagrams.

* <http://stackoverflow.com/>
* <https://www.youtube.com/>
* <http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html>