

**PROGRAMMING TECHNIQUES**

Homework 4

**Student:Nicolescu Alexandru**

**Group:** 30422

**Table of Contents**

**1. Objective 3**

**2. Problem analysis, modeling, scenarios, use cases 3**

**2.1. Problem Analysis 3**

**2.2. Modeling 4**

**2.3. Scenario and use cases 4**

**3. Design 5**

**3.1. UML Diagram 5**

**3.2. Data Structures 6**

**3.3. Class Projection 6**

**3.4. User Interface 8**

**4. Implementation and use of cases 9**

**5. Results 10**

**6. Conclusion and possible updates 10**

**7. References 10**

**Documentation Bank for processing reglar operations**

1. Objective

This homework’s aim is to use elements of object-oriented programming to implement a Bank with the usual operations used by a regular customer.

1. Problem analysis, scenarios, use cases

*1. 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.*

2.1 Problem analysis

If we take a closer look to the problem of bank management we shall find that it is a more complex problem that it may seem at first glance.

First of all we need to find an OOP style way to store the data that comes from the serializable file. I have used an object for each Person and Account. Therefore we have : Person, Account, Map<Person,Set<Account>>.

Another problem concerning the bank processing would be the way the user gives us the data and the way we tell him the result of the processing. After doing some research on the internet and looking at online applications that provide these services I have reached the conclusion that the best way in terms of displaying data which comes from tables is using naturally a JTable which is put in the Graphic User Interface along with the data input section.

Further on we shall analyze all the aspects which needed to be managed in order for the right functioning of this project.

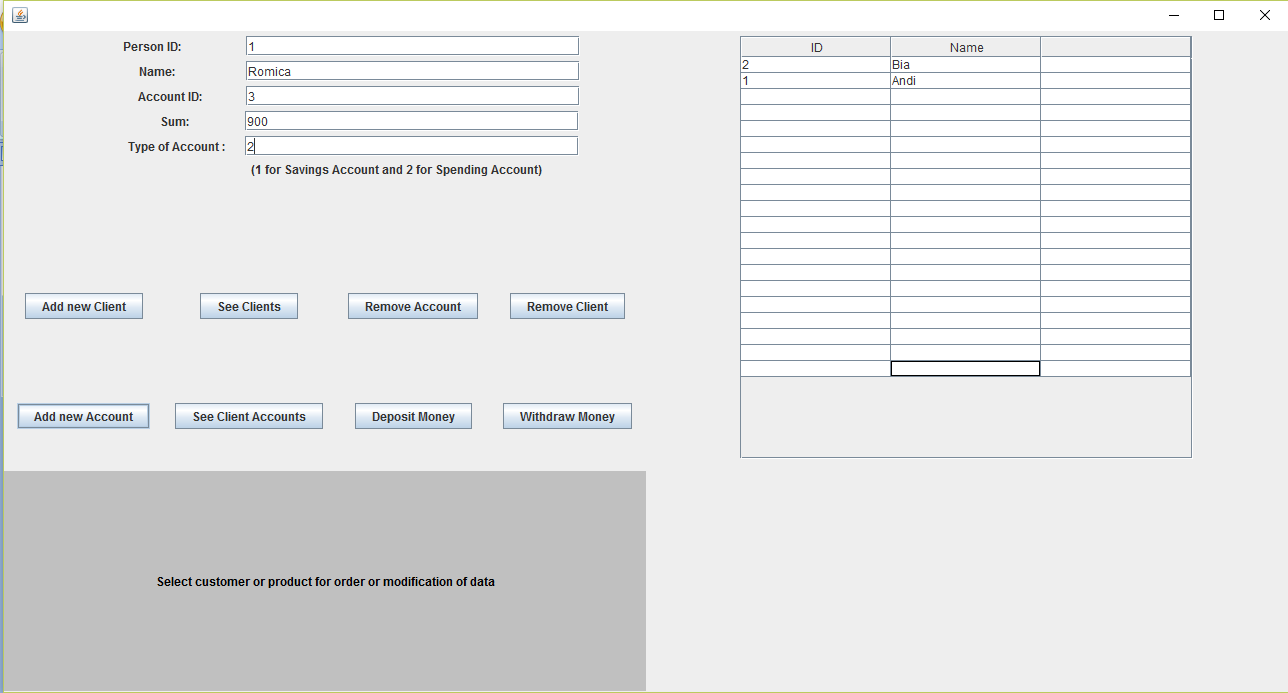
* 1. Modeling

The implementation of this project is the same one as the one presented in the section 2.1. The “assig4” package which stands for Assignemnt 4 has all the logic of storing and displaying data while the “GUI” package naturally only makes the application-user connection using a very smooth and efficient data inputting. There operations are implemented for each of the 2 important Models: Person, Account.

2.3 Scenarios and use cases

The scenarios were already mentioned, but I will present the details here. Firstly I thought about how the data will be managed by the application and my idea was: GUI for reading the data -> BLL for verifying the integrity of the data -> Bank to do the processing of Accounts and Persons if necessary -> Account for further processing the data -> GUI for displaying. I followed this plan and I did not have any unexpected surprise when it comes to the implementation and the structure of the program.

The use cases are strictly dependent on the user, and finally I order to make the application as user friendly as possible I decided to implement the following user interface:



From now on I will present the functionality of the application using the Persons functions as the one of Account processing are very similar.

-Create: The user needs to input the personID, name, new account and sum on the account and type of the account of the new person and then press Add new Client button which creates and the displays new list of clients.

-Read: See Persons displays all the current persons.

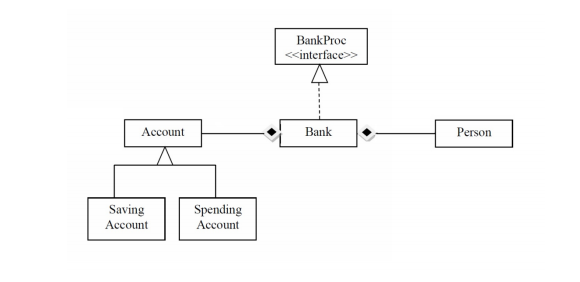
-Deposit Money: Write the data of the account and person and the sum to be added.

Delete: Write the data of the person to be deleted and press delete Client.

Similarly, the rest of the operations are done by this pattern.

1. Design

3.1 UML Diagram



The UML diagram is a class diagram in which we can find the relationship between classes and also the elements that the specified class contains.

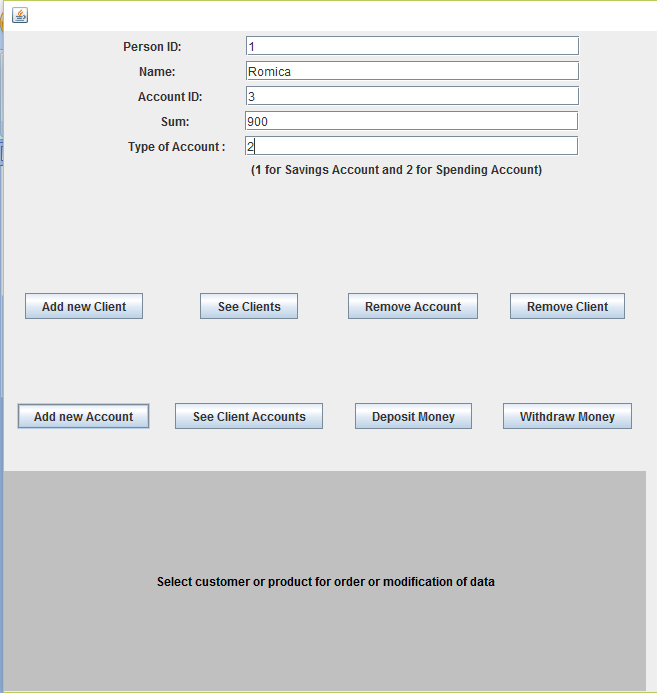
One could observe that for joining the classes I used several types of relationships. Between Account and Bank and Person and Bank there is composision relationship and SavingAccount and SpendingAccount extends Account. There is also a interface BankProc which is implemented by Bank. The GUI is not displayed here, but it uses Bank to diplay and input the data.

3.2 Data structures

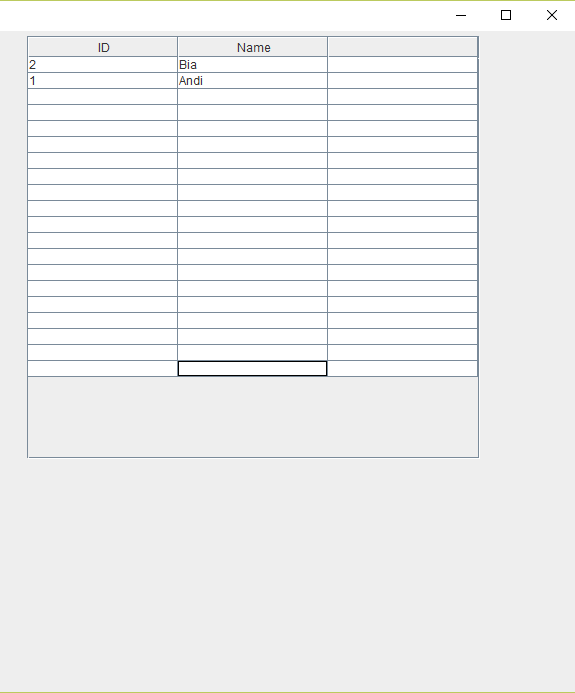
In this application I have used various data types including JButtons, JTextFields, JPanels, JFrame for the GUI package, Account, Person for transferring data between packages and the classic int, String, Boolean for regular operations.

3.3 The interface

The interface is made out of 2 JPanels which are included in the JFrame in a GridLayout(1,2). We shall describe each JPanel in order to understand how the user interface is created.



This photo represents the input panel which is used in order for the user to enter the command data and/or select the command. Some of the data can be selected from the table. There is also a news button which is located on the button of the panel and it displays all the information for using this application and if the case, the errors resulted out of entering the wrong kind of data, therefore the use of this application is easy and does not need any other extra information.



This second photo shows how the resulted queries or edits of the data are displayed to the user using a very straight forward table that displays only the essential information. Numerous bugs are treated in this project such as: selecting no item and trying to deleting it, trying to add a number of “char” sum and so on.

1. Implementation

In what the implementation is concerned this project was developed in Eclipse and it was only tested in this environment. However the program should maintain its portability. Concerning the code implementation I did not make use of laborious algorithms, but I have rather stayed faithful to the classical algorithms of computing polynomials learned in high school. However I have tried to implement my problem in a way that appears to me as being the most efficient one, this is why I have changed my model at first. Testing implies checking for any errors in the program or limitations of this program. Due to the fact that the program is rather simplistic, they are few errors that might generate this program to work wrong or to stop. These errors are mostly related to the interface or the database connection. I have assumed that the user reads the instructions from the interface and respects them, otherwise if he enters data with invalid format the program will probably generate some bugs and will stop. Hence this part with checking all the possible scenarios will be seen as future development.

1. Results

The application is an user friendly and useful application to perform basic create update delete and read operations on a sample created database. As the application is developed on a Java platform, it is highly portable and allows it to run on several operating systems (as long as they have the Java SDK installed). The application is straightforward an easy to understand and to use by any user who respects the instructions given in the interface and who has some basic knowledge of database storing, of course. Even though being limited, this application can be considered as being a helpful tool that can be used when dealing with such data storing situations.

1. Conclusions

All in all, the application works perfectly on the required operations and it is a user friendly interface which gives the users a simple and efficient answer to any problems related to a selected bank operation. As in possible updates I would add more fields to the account and have specific constraints implemented in the bank class, but this of course depends on the application that this program will have.

1. References
2. <http://stackoverflow.com/questions/20327005/jtable-actionlistener-for-select-a-row>