Technical University of Cluj-Napoca

Faculty of Automation and Computer Science

Department of Computer Science

2016/2017

Programming Techniques

Homework 4

Design by Contract, Design Patterns

Lecturer: Prof. Ioan Salomie

Teaching Assistant: Claudia Pop

Student: Acu D. Raul-Mihai

Group: 30425

‘

Table of Contents

1. Problem specification

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. Design 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. 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. Design and implement a test driver for the system.

6. The account data for populating the Bank object will be loaded/saved from/to a file.

a. System`s requirements

• Implement the class diagram from the homework specification. Choose wisely the appropriate data structures for saving the Persons and the Accounts. Differentiate between the Map and the Set data structures.

• Graphical interface: o Window for Person operations: add new Person, edit Person, delete Person, view all Persons in a table (JTable) o Window for Account operations: add new Account, edit Account, delete Account, view all Accounts in a table (JTable)

• Differentiate between the saving and the spending accounts. The saving account allows a single large sum deposit and withdrawal and computes an interest during the deposit period. The spending account allows several deposits and withdrawals, but does not compute any interest.

• Add/Withdraw money from the accounts

• Documentation

2. Usages

a. Persons management

The user is able to add new persons, edit the information of the existing ones, or, remove some persons from the bank.

b. Accounts management

The accounts are created and associated to one person. Only one person can own an account, but a person can own more accounts.

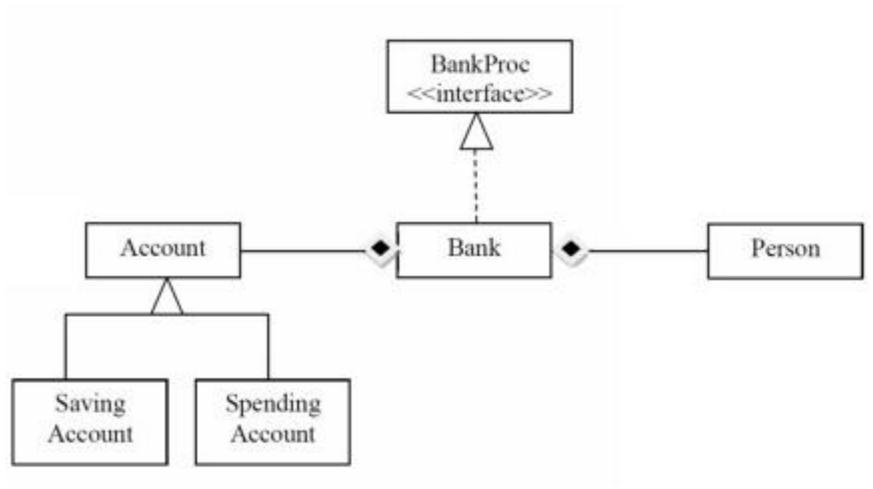
3. Design

a. General Discussions

b. UML Class Diagram

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system.

Structure diagrams emphasize the things that must be present in the system being modeled. Since structure diagrams represent the structure, they are used extensively in documenting the software architecture of software systems



c. Packages

Package **model.**

- Contains the classes used for creating the physical structure of the project.

Package **presentation.**

- Contains the main user interface class;

Package **tester.**

- Contains the JUnit test class.

d. Classes and methods

Person.java

- Stores data related to the Persons that are going to be inserted into the bank.

Account.java

- By calling the constructor **public Account()** of this class, a new object of type Account is created for storing the information about the accounts that are going to be linked to the persons of the bank.

SavingsAccount.java & SpendingsAccount.java

- Abstract class that extends the **Account** class and creates a specific type of **account**.

Bank.java

- The object created by this class is a pair of **Person** and a set of **Accounts**.

BankProc.java

- Interface for the Bank objects;

Gui.java

- Main class for designing the Graphical User Interface.

JUneet.java

- The JUnit test case file, for testing the correctness and functionality of the methods.

4. Using and testing the system

a. User interface

The graphical user interface (GUI), is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.

Designing the visual composition and temporal behavior of a GUI is an important part of software application programming in the area of human–computer interaction. Its goal is to enhance the efficiency and ease of use for the underlying logical design of a stored program.

Such an user interface, is implemented in the **presentation** package, by using the **Gui** class. The user interface is based on the properties of the above mentioned packages. All the objects we need are declared as attributes of the class and they are initialized in the constructor **Gui()**.

- **JPanel** for the main window application:
  - **private JPanel** panelAccounts;
  - **private JPanel** panelPersons;

- **JLabels** for the written text
  - **private JLabel** lblPersonId;
  - **private JLabel** lblAccountId;

- **JTextFields** used for communicating with the database

  private JTextField txtPersonID;
  private JTextField txtPersonAge;
  private JTextField txtPersonName;
  private JTextField txtAccountID;
  private JTextField txtAccountOwner;
  private JTextField txtAccountBalance;
  private JTextField txtMoney;

- **JTables** for displaying the tables` content .

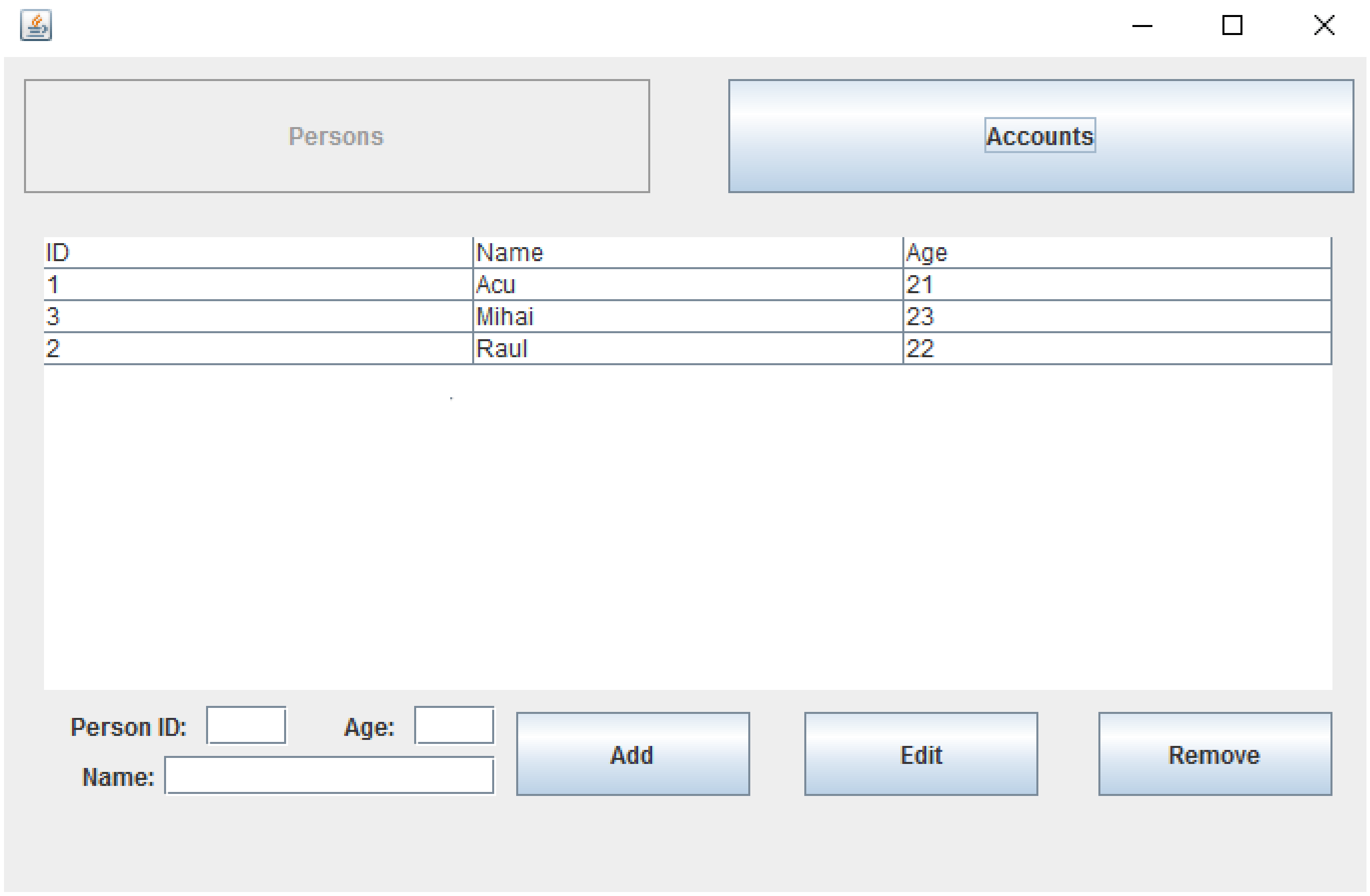
  JTable table;
  JTable table_1;

- **Various methods** that contribute to the element`s properties,

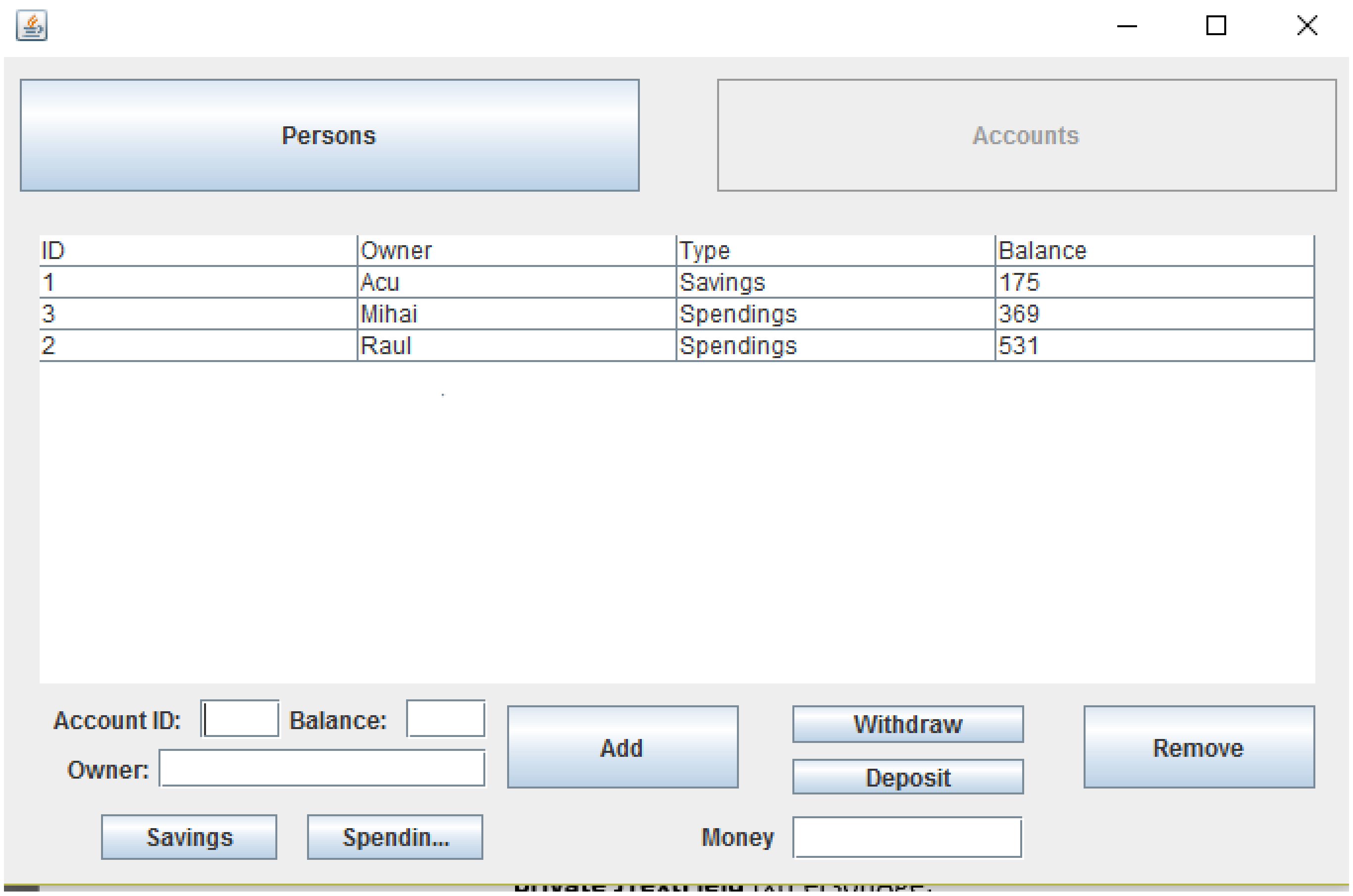
are responsible of adding them to the interface and describing their behavior:

public void tabS()
public void tabA()
public void writeBank()
public void readBank()

The actual **Graphical User Interface** is presented below.



ID	Name	Age
1	Acu	21
3	Mihai	23
2	Raul	22



ID	Owner	Type	Balance
1	Acu	Savings	175
3	Mihai	Spendings	369
2	Raul	Spendings	531

5. Conclusion

a. Advantages and Disadvantages

- **Advantages**
  - Easy to use;
  - Plain and simple, nothing to be confused by.
- **Disadvantages**
  - One client cannot order more than a product at a time.
  - It might fail if given wrong inputs;

b. Future development: ways of improving the system

- Desigining a better relational database;
- Trying to use as less memory as possible;
- Trying to improve the response time;
- Designing a more user-interactive interface;