

DOCUMENTATIE TEMA 4

Bank

Baymuhammedov Akysh

Grupa: 30221

Profesor Laborator Assist Antal Marcel

Contents

1.	Cerinte Functionale	3
2.	Obiective	3
2.1.	Obiectiv Principal:	3
2.2.	Obiective Secundare:	3
3.	Analiza Problemei	3
4.	Proiectare	4
4.1.	Structuri de date	4
4.2.	Diagrama de clase	6
4.3.	Algoritmi	6
5.	Implementare	7
6.	Testare	12
7.	Concluzii si Dezvoltari Ulterioare	13
8.	Bibliografie	14

1. Cerinte Functionale

- 1.1. Implementati o aplicatie care face operatii de adaugare conturi bancare, stergere si modificare.
- 1.2. Implementati interfata BankProc care are metodele de addPerson, removePerson, addAccount si removeAccoun.
- 1.3. Creare si implementare clasele Person, Account, SavingAccount si SpendingAccount.
- 1.4. Definirea si implementarea Observer Design Pattern pentru notificare detinatorul contului despre orice modificari in contului.
- 1.5. Afisare conturi la JTable.
- 1.6. Implementare metoda de Design by Contract pentru pre, post conditii si invarianti.
- 1.7. Implementare un test drive pentru program folosind JUnit.
- 1.8. Salvare si citirea conturi din fisier.

2. Obiective

2.1. Obiectiv Principal:

(obiectivul principal al proiectului)

Proiectare program pentru banca si client folosind metoda de Design by Contract.

2.2. Obiective Secundare:

(pasii care trebuie urmati pentru a atinge obiectivul principal)

Requirement	Grading
Use JTables for display. Implement click listeners on the JTable.	1 point
Design a test driver for the system (JUnit)	1 point
Design by contract: preconditions and postconditions in the BankProc interface. Implement them in the Bank class using the assert instruction. Define the invariant for the class Bank.	1 point
Design Pattern Observer for notifying the account holder when one of its accounts is modified	1 point
Save the information from the Bank class in a file using serialization. Load the information when the application starts.	1 point

3. Analiza Problemei

Use cases:

1. Adaugare Persoane.

2. Stergere Persoane.
3. Adaugare Conturi.
4. Stergere Conturi.
5. Scoatere Bani.
6. Adaugare Bani.

Exemplu:

Pentru adaugare:

Person = Nume: "Chuck", Prenume: "Norris", CNP: "12345666141532"

Cont = tip: "Saving Account", bani: 100 euros

Reguli la crearea contul:

Person != NULL.

Person nu e inregistrat in Banca (adica nu are cont existent) .

Are nume si CNP correct.

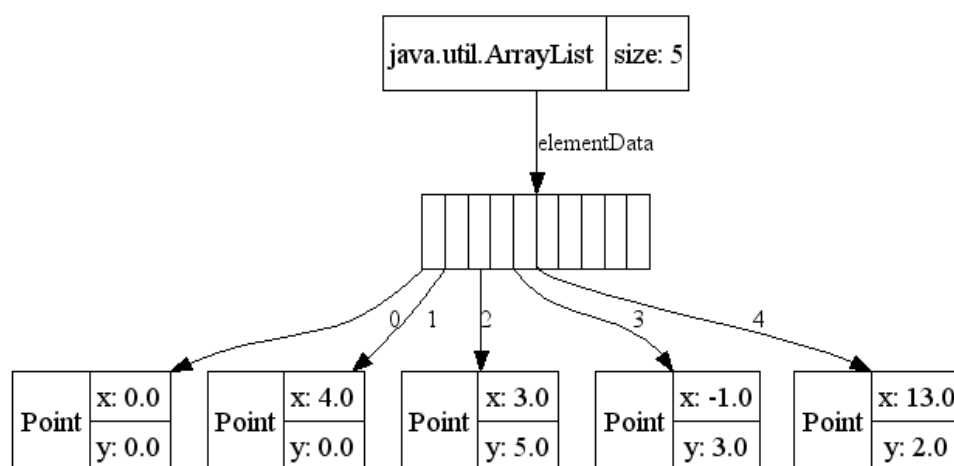
4. Proiectare

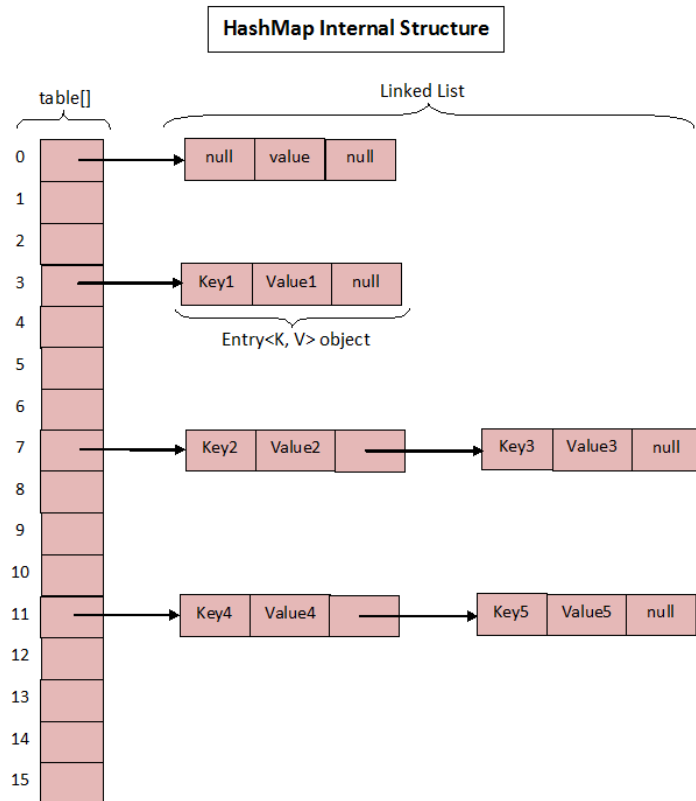
4.1. Structuri de date

Am folosit structuri de date de tip ArrayList pentru stocarea Conturi pe fiecare Person si pentru stocarea personae si conturi lui am folosit HashMap.

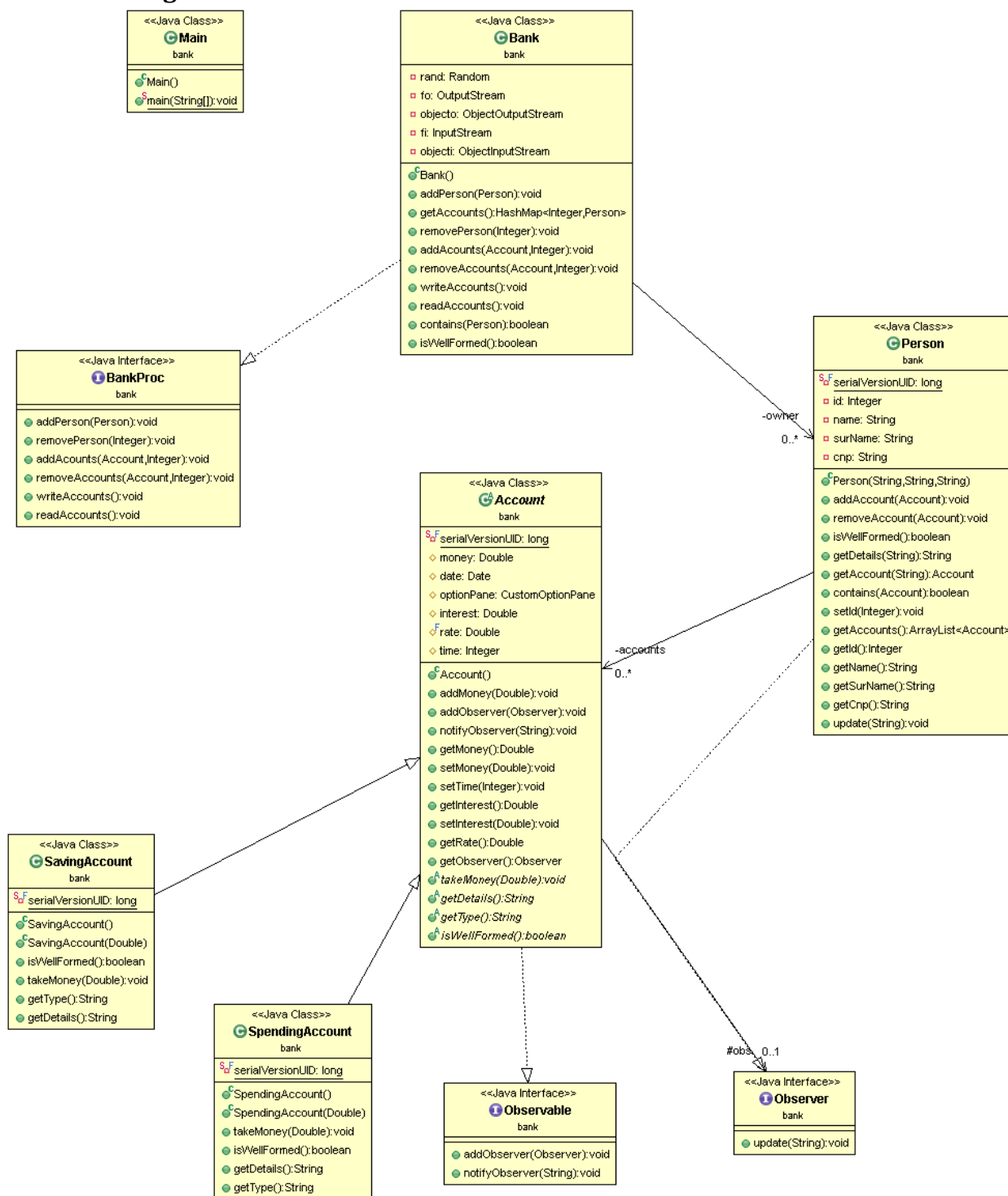
```
private ArrayList<Account> accounts;
```

```
private HashMap<Integer, Person> owner;
```





4.2. Diagrama de clase



4.3. Algoritmi

Calculare Interesul pentru conturi Saving Account. Scadere suma bani dupa scoatere bani si adunare bani dupa adaugare la cont si etc..

5. Implementare

Sunt 5 clase si 3 interfete.

1. Clasa Bank are campuri:

```
private HashMap<Integer, Person> owner = new HashMap<Integer, Person>();  
private OutputStream fo;  
private ObjectOutputStream objecto;  
private InputStream fi;  
private ObjectInputStream objecti;
```

HashMap pentru stocare Person cu conturi lui.

fo si fi sunt Streamuri pentru deschidere fisier si objecto si objecti sunt pentru citirea si scrierea in fisier.

Metode:

```
public void addPerson(Person person);
```

Este metoda pentru adaugare perosane noua la HashMap in conditiil de:

Person!=NULL, Person !exista in HashMap, Person are nume si cnp corecte. (Pre conditions)

Daca conditiile sunt indeplinite atunci adauga la HashMap.

Dupa adaugare verifica size-ul Hashmap-ului daca preSize < currentSize. (Post condition)

Dupa verifica da HashMap-ul != NULL.

```
public void removePerson(Integer id);
```

Gaseste personul din HashMap prin id apoi il sterge cu toate conturile lui.

Conditii care trb sa fie indeplinite pentru stergere:

Id trb sa fie mai mare decat 0 si nu trb sa fie NULL. (Pre condition)

Dupa stergere verifica size-ul HashMap, trb sa fie preSize > currentSize. (Post condition)

Si apoi verifica daca HashMap-ul nu e NULL.

```
public void addAccounts(Account ac, Integer id);
```

Adaugare un al doilea cont pentru un anumit person prin gasirea person din HashMap dupa id apoi prin aduagare contul 'ac' la lista lui de cont.

Conditiiile de pre si post sunt cam aceleasi ca si la metoda removePerson in afara de verificare size=ul (listAccount.currentSize > listAccount.preSize).

```
public void removeAccounts(Account ac, Integer id);
```

Stergere unui cont din person.

Conditiiile sunt aceleasi ca si in addAccounts in afara de listAccount.currentSize < listAccount.preSize.

Si in caz daca dupa stergere person nu are nici un cont atunci sterge si prsonul respective.

```
public void writeAccounts() throws IOException;
```

Scriere la fisier Accounts.txt. In caza de eroare arunca Exceptie.

```
public void readAccounts() throws IOException;
```

Citirea din fisier Accounts.text . In caza de eroare arunca Exceptie.

2. Clasa Account are campuri:

```
protected Double money = 0.0;
protected Observer obs;
protected CustomOptionPane optionPane = new CustomOptionPane();
protected Double interest = 0.0;
protected final Double rate = 0.7;
protected Integer time;
```

obs este Observerul contului (main holder).

Metode:

```
public void addMoney(Double money);
```

Adauga bani la cont prin coditiile:

Bani trb sa fie mai mare decat 0. (Pre condition)

Dupa adaugare verifica daca currentMoney > preMoney. (Post condition)

Dupa verifica daca money != null si nu e < 0.

Apoi face notify la observerul lui prin o fereastră de mesaj.

```
public void addObserver(Observer obs);
```

Adauga observer(main holder) pentru notificare in caz de modificari in cont.

3. Clasa SavingAccount:

Este o clasa care extinda de catre clasa abstracta Account.

Metoda:

```
public void takeMoney(Double money);
```

Scoatere bani din cont. La Savings account e diferit regulile de scoatere bani decat in Spending Account. Suma care vrei sa iei trb sa fie mare (mai mare decat 100 euro) si poti imprumut bani din banca in caz daca nu ai bani sufficient in cont dar cu conditiile de platirea Interest (suma care ai imprumutat + procent din suma) intre anumit timp (cel putin un an). Calculare de Interest este $\text{Interest} = \text{money} * (\text{time} * \text{rate})$. Rate este procentul pe care cere bank in plus in caz imprumutare.

Conditiiile trb sa fie:

Money > 100, in caz de money > moneyInAccount atunci + Interest. (Pre condition)

Dupa scoatere bani verifica daca money < 0. (Post condition)

Dupa terminarea cu success face notify la observerul lui.

4. Clasa SpendingAccount:

Este o clasa care tot extinda de clasa Account.

Metoda:

```
public void takeMoney(Double money);
```

Scoatere bani din cont. Reguli sunt simple:

Bani sa fie > 0 si sa fie divizibil la 10 si sa nu fie $> \text{moneyInAccount}$. (Pre conditions)

Dupa scoatere bani verifica daca $\text{money} \neq \text{null}$, $\text{preMoney} > \text{currentMoney}$. (Post conditions)

Face notify la Observrul lui.

5. Clasa Person are campuri:

```
private Integer id;  
private String name, surName, cnp;  
private ArrayList<Account> accounts;  
private DateFormat df = new SimpleDateFormat("dd/MM/yy HH:mm:ss");  
private Date dateobj = new Date();
```

accounts este lista care stocheaza toate conturi.

Metode:

```
public void addAccount(Account ac);
```

Este apelat de catre clasa Bank pentru adaugare cont nou. Condițiile sunt aceleasi ca si in Bank doar ca verifica si daca exista un cont de aceleasi tip.

```
public void removeAccount(Account ac);
```

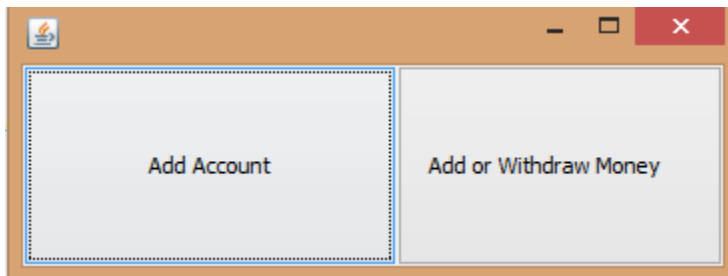
Este apelat de catre Bank si conditiile sunt aceleasi. Sterge cont daca exista prin iteratie spre lista pana sa gaseste contul si il sterge.

```
public void update(String changes);
```

Primește notificari si il arata prin o fereastră cu afisare data de modificari si ce a modificat din cont.

Interfata:

La rulare proiect o sa apare doua butoane:



1 Buton este pentru adaugare cont/person, stergere cont/person iar al 2-lea buton este pentru vederea detalii despre cont.

Pentru adaugare un cont nou:

Id	Name	Surname	CNP	type
260	wdwdwaa	dawdawddw	21321332121321...	Saving Account
302	dadawdawdd	fafwfwafwfw	2321213123	Saving Account

Completati campurile selectati tipul de cont si varsta (varsta trebuie sa fie mai mare decat 18 ani) si apasati la butonul add.

Id	Name	Surname	CNP	type
260	wdwdwaa	dawdawddw	21321332121321...	Saving Account
302	dadawdawdd	fafwfwafwfw	2321213123	Saving Account

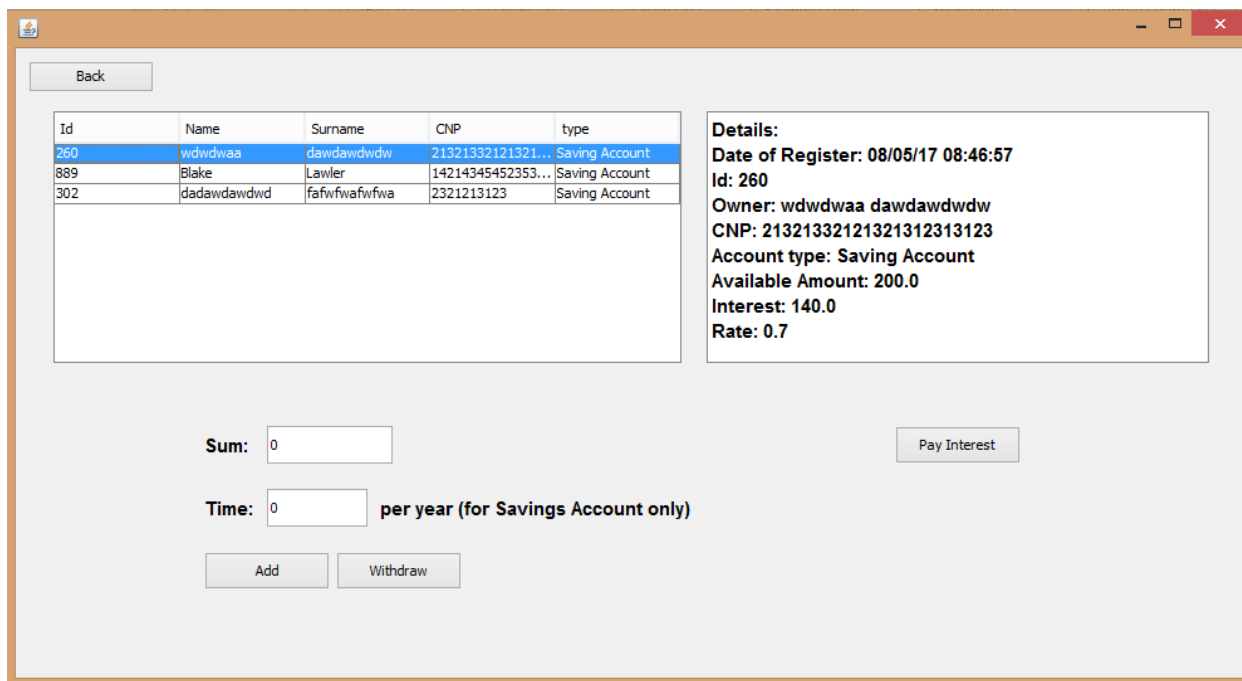
Si o sa apare la table de stanga.

Pentru adaugare un al doilea cont la un person selectati un id din table si selectati un tip de cont care nu exista si pasati la butonul 'add 2nd account'.

Pentru stergere un person selectati un id si pasati la 'Remove Person'.

Pentru stergere un cont din person selectati tipul de cont din table si pasati la 'Remove Account'

Daca pasam la al 2-lea buton o sa apare:



Back

Id	Name	Surname	CNP	type
260	wdwdwaa	dawdawddw	21321332121321...	Saving Account
889	Blake	Lawler	14214345452353...	Saving Account
302	dadawdawdw	fafwfwafwfiwa	2321213123	Saving Account

Details:
Date of Register: 08/05/17 08:46:57
Id: 260
Owner: wdwdwaa dawdawddw
CNP: 21321332121321312313123
Account type: Saving Account
Available Amount: 200.0
Interest: 140.0
Rate: 0.7

Sum: **Pay Interest**

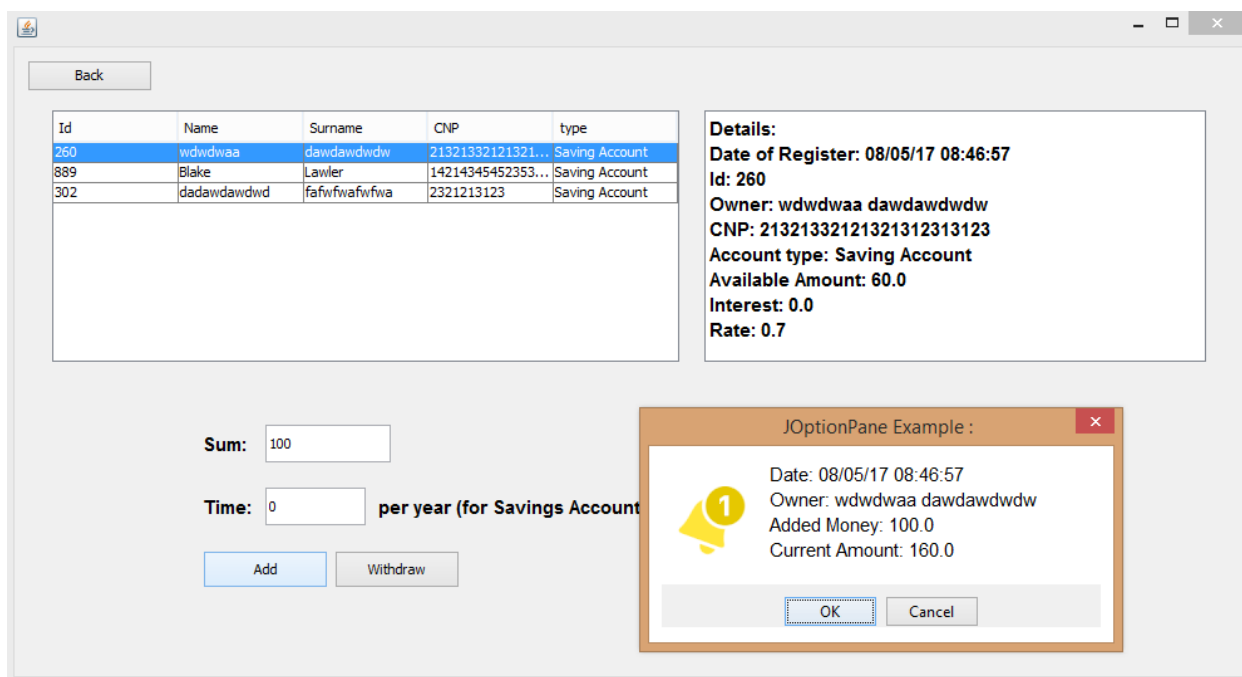
Time: **per year (for Savings Account only)**

Add **Withdraw**

Unde se pot vedea detalii despre cont in fereastra de stanga dupa selectare un cont.

Pentru aduagarea bani inserati suma d bani la campul sum si apasati la butonul 'Add'

Si o sa va apare o fereastra mica care este o notificare pentru main holder-ul contului ca sa a facut modificare la cont:



Back

Id	Name	Surname	CNP	type
260	wdwdwaa	dawdawddw	21321332121321...	Saving Account
889	Blake	Lawler	14214345452353...	Saving Account
302	dadawdawdw	fafwfwafwfiwa	2321213123	Saving Account


Details:
Date of Register: 08/05/17 08:46:57
Id: 260
Owner: wdwdwaa dawdawddw
CNP: 21321332121321312313123
Account type: Saving Account
Available Amount: 60.0
Interest: 0.0
Rate: 0.7

Sum: **Pay Interest**

Time: **per year (for Savings Account only)**

Add **Withdraw**

JOptionPane Example :

 **1**

Date: 08/05/17 08:46:57
Owner: wdwdwaa dawdawddw
Added Money: 100.0
Current Amount: 160.0

OK **Cancel**

Daca vreti sa scoateti bani selectati cont si inserati suma si apasati la buton 'Withdraw'. (Doar pentru conturi Saving Account) in caz daca vreti sa scoateti suma mai mare decat ce aveti in cont, puteti imprumuta din banc cu conditie de platirea procent. Pentru asta inserati timpul (pe ani) de asteptare pentru banc pana platiti si apasati la withdraw. O sa vedeti la detalii ca va a calculate suma care trb sa platiti pentru banc in campul 'Interest'. Pentru platirea Interesul pasati la 'pay interest'.

6. Testare

Am facut Testare pentru clasele Bank, Account si Person. (Exemplele de implementare am scris deja sus la 'Interfata').

BankTest:

Am testat metodele AddPerson(), RemovePerson(), addAccounts(), removeAccounts() si getAccounts():

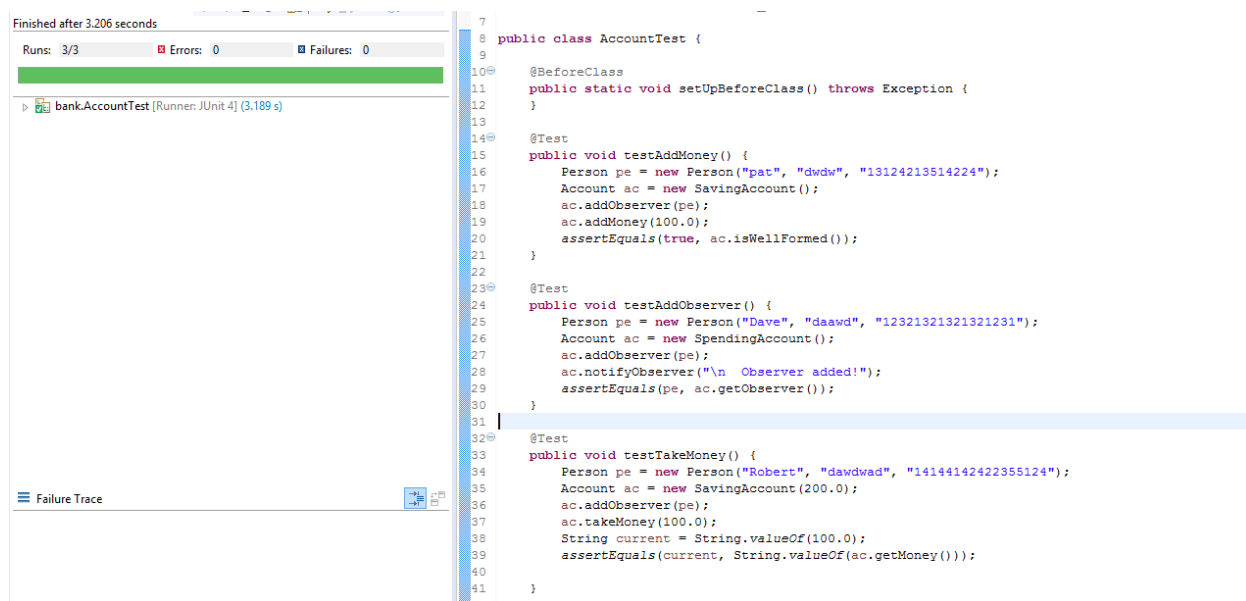
```

10 @BeforeClass
11 public static void setUpBeforeClass() throws Exception {
12 }
13
14 @Test
15 public void testAddPerson() {
16     Bank bank = new Bank();
17     Person p = new Person("Nate", "blabla", "11233214221456686");
18     SavingAccount ac = new SavingAccount(100.0);
19     p.addAccount(ac);
20     bank.addPerson(p);
21     assertEquals(true, bank.isWellFormed());
22 }
23
24 @Test
25 public void testGetAccounts() {
26     Bank bank = new Bank();
27     Person p = new Person("Felix", "goowill", "3213213231234");
28     SpendingAccount ac = new SpendingAccount(50.0);
29     Person p1 = new Person("Ken", "lawler", "3213343433434343");
30     SavingAccount ac1 = new SavingAccount(50.0);
31     p1.addAccount(ac1);
32     p.addAccount(ac);
33     bank.addPerson(p);
34     bank.addPerson(p1);
35     assertEquals(2, bank.getAccounts().size());
36 }
37
38 @Test
39 public void testRemovePerson() {
40     Bank bank = new Bank();
41     Person p = new Person("Mark", "Pettis", "13232342456564542");
42     SavingAccount ac = new SavingAccount(10.0);
43     p.addAccount(ac);
44     bank.addPerson(p);
45     assertEquals(1, bank.getAccounts().size());
46     bank.removePerson(p.getId());

```

AccountTest:

Am testat metodele addMoney(), takeMoney() si addObserver():



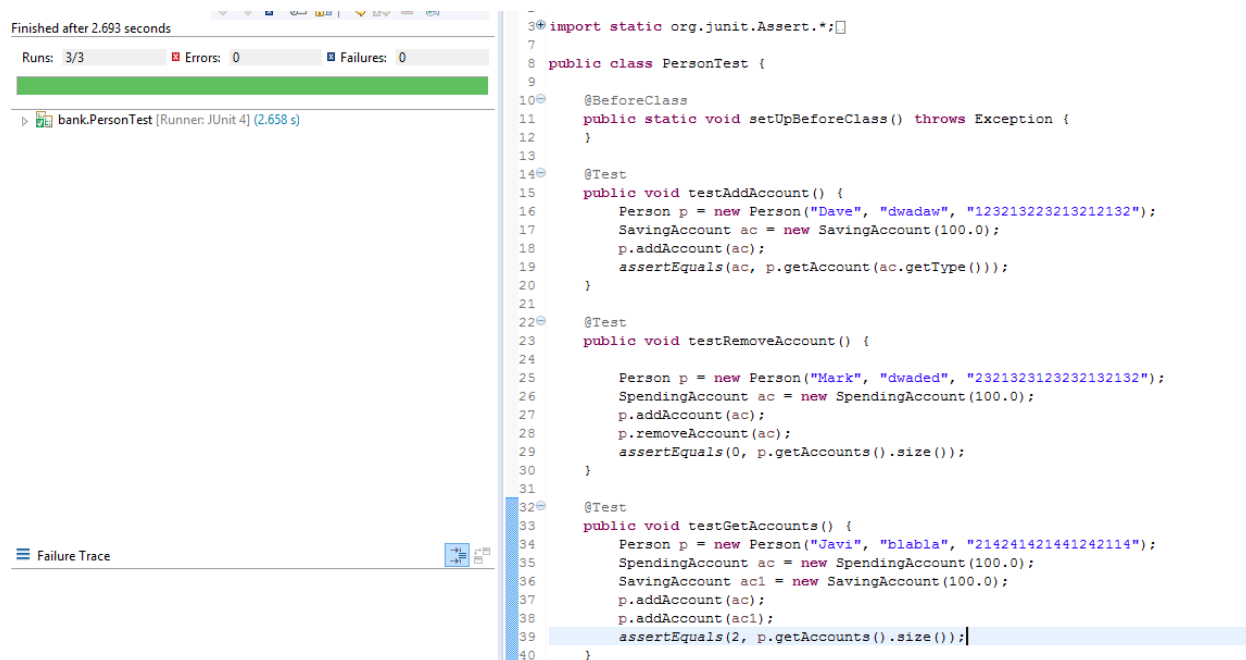
```

7
8 public class AccountTest {
9
10     @BeforeClass
11     public static void setUpBeforeClass() throws Exception {
12     }
13
14     @Test
15     public void testAddMoney() {
16         Person pe = new Person("pet", "dwdw", "13124213514224");
17         Account ac = new SavingAccount();
18         ac.addObserver(pe);
19         ac.addMoney(100.0);
20         assertEquals(true, ac.isWellFormed());
21     }
22
23     @Test
24     public void testAddObserver() {
25         Person pe = new Person("Dave", "daawd", "12321321321321231");
26         Account ac = new SpendingAccount();
27         ac.addObserver(pe);
28         ac.notifyObserver("\n Observer added!");
29         assertEquals(pe, ac.getObserver());
30     }
31
32     @Test
33     public void testTakeMoney() {
34         Person pe = new Person("Robert", "dawdwad", "14144142422355124");
35         Account ac = new SavingAccount(200.0);
36         ac.addObserver(pe);
37         ac.takeMoney(100.0);
38         String current = String.valueOf(100.0);
39         assertEquals(current, String.valueOf(ac.getMoney()));
40     }
41
42 }

```

PersonTest:

Pentru clasa Person am testat metodele addAccount(), removeAccount() si getAccounts()



```

3 import static org.junit.Assert.*;
7
8 public class PersonTest {
9
10     @BeforeClass
11     public static void setUpBeforeClass() throws Exception {
12     }
13
14     @Test
15     public void testAddAccount() {
16         Person p = new Person("Dave", "dwadaw", "123213223213212132");
17         SavingAccount ac = new SavingAccount(100.0);
18         p.addAccount(ac);
19         assertEquals(ac, p.getAccount(ac.getType()));
20     }
21
22     @Test
23     public void testRemoveAccount() {
24
25         Person p = new Person("Mark", "dwaded", "2321323123232132132");
26         SpendingAccount ac = new SpendingAccount(100.0);
27         p.addAccount(ac);
28         p.removeAccount(ac);
29         assertEquals(0, p.getAccounts().size());
30     }
31
32     @Test
33     public void testGetAccounts() {
34         Person p = new Person("Javi", "blabla", "214241421441242114");
35         SpendingAccount ac = new SpendingAccount(100.0);
36         SavingAccount ac1 = new SavingAccount(100.0);
37         p.addAccount(ac);
38         p.addAccount(ac1);
39         assertEquals(2, p.getAccounts().size());
40     }
41
42 }

```

7. Concluzii si Dezvoltari Ulterioare



Trimite notificari pentru Observer prin OptionPane care am modificat, highlight campuri cand inserezi date, calculare interesul pentru conturi Saving Account si etc...

8. Bibliografie

www.stackoverflow.com

www.google.com

www.youtube.com