Bank Management

Descriere

Proiectul de fata implementeaza operatile simplificate ce pot fi efectuate de catre o banca care retine date despre clienti si conturi. Ca si operetii de management sunt implementete: adaugere clienti, stergere, editare client, adaugare conturi,stergere, adaugere bani intr-un cont si preluare de bani dintr-un cont ales. Aceste operatii sunt atribuite direct unor butoane din partea grafica astfel ca interactiunea este usoara si sugestiva. Aceasta parte grafica este constituita din trei ferestre grafice: una care contine date despre conturi (interna), una date despre persoane (interna) si o alta de pe care se execute instructiunile de adaugare si retragere numerar. Modelul operatilor efectuate (pentru fiecare in parte) este descris in clasa Bank, clasa care contine un HashMap in care cheia este o persoana (Person) si valoarea un TreeSet() de conturi (Account). Controller-ul este divizat in mod asemanator in doua parti. Prima parte contine metoda de main si apeleaza la initializare operatile implemetete in cealalta parte, care opereaza in mod exclusiv cu Listenere si operatori lamda. Operatile descrise sunt testate intr-o clasa separata numita BankJUnit unde actiuni precum depuneri si retrageri sunt efectuate pe diferite tipuri de conturi.

Proiectul insumeaza un total de 12 entitati java, incapsulate in pachetul BankSimulation care comunica intre ele, in mod principal prin gettere si settere. Odata pornita aplicatia se asteapta modificarile efectuate de administratorul banci (sau clientul) si se salveaza starea obiectelor intrun fisier special de serializare numit bank.ser care este preluat la deschiderea aplicatiei si atribuie variabilelor de rulare actuale valorile ultimei interactiuni (deserializare). De asemenea, folosind paternul Observer Observable fiecare client este instintat atunci cand sa facut o operatie asupra unuia dintre cele doua conturi pe care le poate detine (Spending si Saving). Biblioteci precum “java.awt”, “java.swing”, “java.lang”, “java.util”,”java.beans” si “java.io” au fost folosite pentru a descrie in mod atomic complex functile scrise. Pentru compilare afost folosit modulul Java 8 si mediul Intelij.

Descrierea Componentelor

Person

Clasa implementeaza interfate Serializable si contine urmatoarele field-uri:

identifier, name, age, address, phone si email. Acestor obiecte private le sunt create in mod particular gettere si settere ce vor fi folosite in interiorul metodelor de prelucrare a structurii bancii.

Account

Aceasta clasa extinde Observable si impementeaza interfetele Comparable si Serializable. Contine atributele comune celor doua tipuri de conturi: sold, holderIdentifier si type alaturi de gettere si settere pentru aceste atribute private.

SpendingAccount

Clasa extinde Account si in constructor dupa apelarea constructorului din superclasa cheama metoda setType cu parametrul actual 0. Contine un unic UID (folosit la serializare), gettere si settere ce apeleaza metodele din parinte si metoda toString care returneaza tipul account –ului (Spending).

SavingAccount

Asemanator celeilalte clase de tipul Account contine un serialVersionUID si un constructor care suprascrie tipul accountului la 1, getere si settere pentru atributele clasei si superclasei. Diferentierea fata de spending account este indicata de fieldul interestRate initilializat cu 10 care reprezinta dobanda acumulata in urma unei perioada de timp (ignorata).

BancProc | Bank

Interfata BankProc descrie headerele a 14 metode care vor fi descrise in clasa implementatoare, Bank. Structura de date care retine modificarile facute in banca este un HashMap ce contine ca cheie un client si ca valoarea un TreeSet de conturi. Urmatoarele metode sunt descrise:

- isInHash/inNoInHash – metode folosit drept invariant (asertiuni)in metodele directe de prelucrare a hash-ului. Cele doua metode primesc un identificator si returneaza o valoare booleana daca exista persoana sau nu persoana cu acel identificator in hash.

- isInHash/inNoInHash – metode ce primesc un account si verifica daca acesta este atribuit sau nu vreunei persoane din hash, o valoare booleana care indica acest lucru este returnata.
- isPositive/isAccountPositive – verifica daca numarul total de bani aflati in banca este unul pozitiv respectiv daca intrun anumit cont soldul este pozitiv.
- addPerson/removePerson / editPerson - descriu operatia de adaugare a unei persoane in hash, eliminarea ei si modificarea atributelor propri la alte valori. Metodele folosesc pre si post conditii pentru validarea datelor introduse
- addAccount / removeAccount – Parcurg structura de date si adauga, respectiv elimina un cont in functie de tipul si identificatorul detinatorului;
- addMoney- Pentru anumite date account verifica in hash tipul accountu-lui si adauga o suma. Daca tipul contului este saving atunci o conditie este plasata, astfel ca o singura depunere se poate face, in celalat caz (spending) se pot face un numar nelimitat de depuneri;
- takeMoney - operatia opusa celei de depunere, se cauta in hash clientul care efectueaza depunerea si daca soldul este mai mare decat suma care este extrasa metoda returneaza suma actuala. In mod similar daca tipul contului este Saving o singura extragere se poate efectua, contul ramanand cu soldul nul.
- Update – metoda este una ceruta in mod obligatoriu cu implementarea interfetei Observer si la fiecare modificare a soldu-lui unui client acesta este instintat in legatura cu noua suma pastrata in cont:

```
@Override
public void update(Observable o, Object arg) {
    Account account=(Account)arg;
    double currentSold;
    if(account.getType()==0){
        currentSold=account.getSold()*1.05f;
    }
    else {
        currentSold=(double) account.getSold();
    }
    System.out.println("The person with identifier
"+account.getHolderIdentifier()+" get his/her sold modified at
"+currentSold+" for the account type: "+account.toString());
}
```

Graphic

GraphicWindow | PersonWindow | AccountWindow

PerosonWindow

Clasa este una de tipul JFrame si contine containare specifice ce vor fi afisate in fereastra, metode, gettere si settere. Containare: doua panouri, o tabela, un model de tabela, trei butoane si daua tablouri, unul de etichete si unul de zone de text. In constructor sunt apelate doua metode private: init si setFrame. Metoda de init initializeaza obictele seteaza unui panou tabela, tabelei modelul si proprietatea de nemodificare directa a celulelor din aceasta si creaza obiecte de tipul tablourilor, initiaalizate cu un identificator (nume) specific. Metoda setFrame oferea caracteristicile spefcifice ferestrei: dimensiune, vizibilitate, locatie, titlu etc. Cele doua panouri sunt adaugate ferestre grafice si sunt apelate metode care plaseaza in fereastra fiecare container, seteaza culori si layout. Alte metode continute de clasa sunt gettere si settere.

AccountWindow

Clasa este una de tipul JFrame si contine containare specifice ce vor fi afisate in fereastra, metode, gettere si settere. Containare: doua panouri, o tabela, un model de tabela, un combo box, doua butoane si daua tablouri, unul de etichete si unul de zone de text. In constructor sunt apelate doua metode private: init si setFrame. Metoda de init initializeaza obictele seteaza unui panou tabela, tabelei modelul, proprietatea de nemodificare directa a celulelor din aceasta, hederul de tabel si creaza obiecte de tipul tablourilor, initializate cu un identificator (nume) specific. Metoda setFrame oferea caracteristicile spefcifice ferestrei: dimensiune, vizibilitate, locatie, titlu etc. Cele doua panouri sunt adaugate ferestre grafice si sunt apelate metode care plaseaza in fereastra fiecare container, seteaza culori si layout. Alte metode continute de clasa sunt gettere si settere.

GraphicWindow

Clasa este una de tipul JFrame si contine cele doua ferestre grafice descrise

mai sus, containare specifice ce vor fi afisate in fereastra, metode, gettere si settere. Containare: doua panouri, un model de tabela, un combo box, patru butoane si un tablour de etichete si trei zone de text. In constructor sunt apelate doua metode private: init si setFrame, oferea caracteristicile spefcifice ferestrei: dimensiune, vizibilitate, locatie, titlu etc. Metoda de init initializeaza obictele create, cele doua ferestre grafice, etichetele, zone de text, butoanele combo box-ul si panoul ferestrei unde urmeaza a fi adaugate toate containarele. Metoda setPanel adauga prin metoda add toate containarele create panoului (dupa ce ii seteaza acestuia layout - ul) si apeleaza metoda showContainers. Metoda seteaza in coordonate x, y, width si height coordonatele spatiale in fereastra grafica toate obiectele create(mai putin fram- urile) si adauga in felul urmator un ActionListener butoanelor ce trebuie sa lanseze ferestrele grafice.

```
toAccount.addActionListener(e -> accountWindow.changeVisible(true));
toPerson.addActionListener(e -> personWindow.changeVisible(true));
```

Alte metode continute de clasa sunt gettere si settere precum si metode ce apeleaza metodele din cele doua ferestre pentru a putea folosi imbricat resursele puse la dispozitie de acestea.

BankActions

Clasa contine actiunile bancare care sunt implementate de Controller si care modifica la interactiune cu fereastra obiecte din diferite clase ale proiectului. Metodele descrise sunt statice primesc doi parametri formali, unul de tipul bancii si unul de tipul ferestrei grafice si ataseaza un listener lamda unui obiect si se comporta in felul urmator:

- getAccount – Preia datele introduse in text field-uri si creaza un nou account (operatii specifice pentru fiecare tip de cont in parte) al carui sold este setat la 0. Metoda addAccount este apelata pe obiectul de tip Bank si in functie de valoarea returnata datele despre cont vor fi sau nu introduse in hash si in JTabel-ul ferestrei AccountWindow. Valoarea 0 returnata

specifica faptul ca un assert a fost declansat sau ca account-ul de introdus nu respecta anumite cerinte;

- deleleteAccount- ataseaza butonului de delete din AccountWindown actiunea urmatoare: se cauta linia din tabela care corespunde identificatorului account-ului, prin metoda removeRow linia este stearsa si pe obiectul de tip bank metoda deleteRow este apelate (pentru stergerea efectiva din hash);
- addNewPerson – preia datele din containarele grafice, creaza o noua persoana, o adauga in JTable si in hashMap, toate sub actiunea butonului de adaugare din PersonWindow;
- deletePerson – asteapta ca field-ul de id sa fie introdus, parcurge multimea de linii din JTable si daca gaseste o linie cu identificatorul specificat sterge linia si apeleaza metoda de stergere din Hash;
- editPerson - preia datele din containarele PersonWindow, creaza o noua persoana, sterge linia din tabela cu identificatorul persoanei si adauga o linie cu noile date. Metoda de editPerson este de asemenea apelata pe obiectul bank;
- addSold – din fereastra grafica principala preia identificatorul, tipul contului si suma ce se doreste a fi depusa, se updateaza linia corespunzatoare din JTable – ul deaccount-uri si se apeleaza metoda addMoney pentru introducerea in hash;
- withDrawMoney – apeleaza in mod asemanator, dar concurent cu metoda de adaugare numerar un operator lamda, updateaza linia din JTable si modifica elementul hash-ului.
- initializeTabele – este folosita la deserializare, preia toate cheile si valorile din hash si le introduce in tabela corespunzatoare din cele doua ferestre grafice;
- autoFill – metoda atribuie tabelei de persoane un clickListener care dupa ce s-a selectat o linia, datele ei sunt introduse in field-urile de text ale panoului.

BankControl

Clasa are un constructor care in mod static apeleaza metodele din Bank

Action. Metoda main incepe cu deserializarea obiectelor de tipul bancii, initializeaza fereastra grafica, creaza un obiect de tipul clasei, asteapta modificari asupra ferestrei grafice si dupa serializeaza noul hassMap (Tipul bancii) in fisierul bank.ser.

BankJUnit

Clasa extinde TestCase, creaza obicte Bank, Account si person si contine patru metode ce modifica aceste obiecte. Metodele creaza o pesoana si un account, apelaza metode de adaugare numerar si retragere si prin asertiuni de tipul assertEqual verifica daca operatile au fost efectuate in mod corect, daca dobanda a fost adaugata si daca se respecta diferentierea functionala dintre cele doua tipuri de conturi.