Tema 4: banca

Student: Campean Bogdan Mihai

Grupa: 30221

**1. Obiectivul temei**

Obiectivul principal al acestei teme este de a simula functionarea corecta a unei banci din viata cotidiana, trecand prin toate aspectele si toate functionalitatile acesteia. Aceasta banca trebuie sa functioneze corect din 2 perspective: in primul rand din perspectiva bancii, unde trebuie lucrat cu conturi, client, etc; iar in al doilea rand din perspectiva unui client care poate deschide/inchide un cont sau sa adauge/retraga numerar din banca, functionalitatea unui ATM.

Obiectivul secundar al acestei teme este de a folosi serializarea combinata cu hashmap-urile din limbajul de programare orientat pe obiecte Java pentru scrierea in fisiere a datelor clientilor si a tuturor conturilor acestora, daca exista. De asemenea trebuie folosita tehnica design by contracts pentru a permite altor programatori care lucreaza la dezvoltarea ulterioara a aplicatiei de a cunoaste fiecare metoda ce preconditii si postconditii sunt necesare pentru a o folosi sau pt a o modifica.

2. Analiza problemei, asumptii, modelare, scenarii, cazuri de utilizare, erori

Incepem analiza acestei problem prin evidentierea cuvintelor cheie care se dau in enuntul problemei si impartirea problemei mari in subprobeleme mai mici.

In primul rand observam fraza cheie “design by contracts”, care ne oblige sa folosim aceasta metoda pentru intelegerea ulterioara mai usoara a codului scris de noi de alti programatori care dezvolta ulterior aplicatia. Aceasta tehnica este o tehnica informative care nu influienteaza deloc functionarea corecta a algoritmului propus; de aceea aceasta tehnica este scrisa in pseudocod in comentarii; se bazeaza pe unele pre conditii si post conditii care sunt impuse unei metode pentru functionarea corecta a acesteia; se folosesc conditiile ca fiind niste if-uri puse cu semnul “@”.

In al doilea rand gasim cuvantul cheie “serializare”, care se bazeaza pe folosirea fisierelor de serializare in limbajul de programare orientate pe obiect Java, pentru tinerea datelor tuturor clientilor bancii, impreuna cu conturile acestora, suma, etc. Aceasta tehnica trebuie folosita in locul crearii unei baze de date in alt limbaj de programare pentru baze de date, pentru a nu complica problema cu alte tabele, relatii intre tabele, etc.

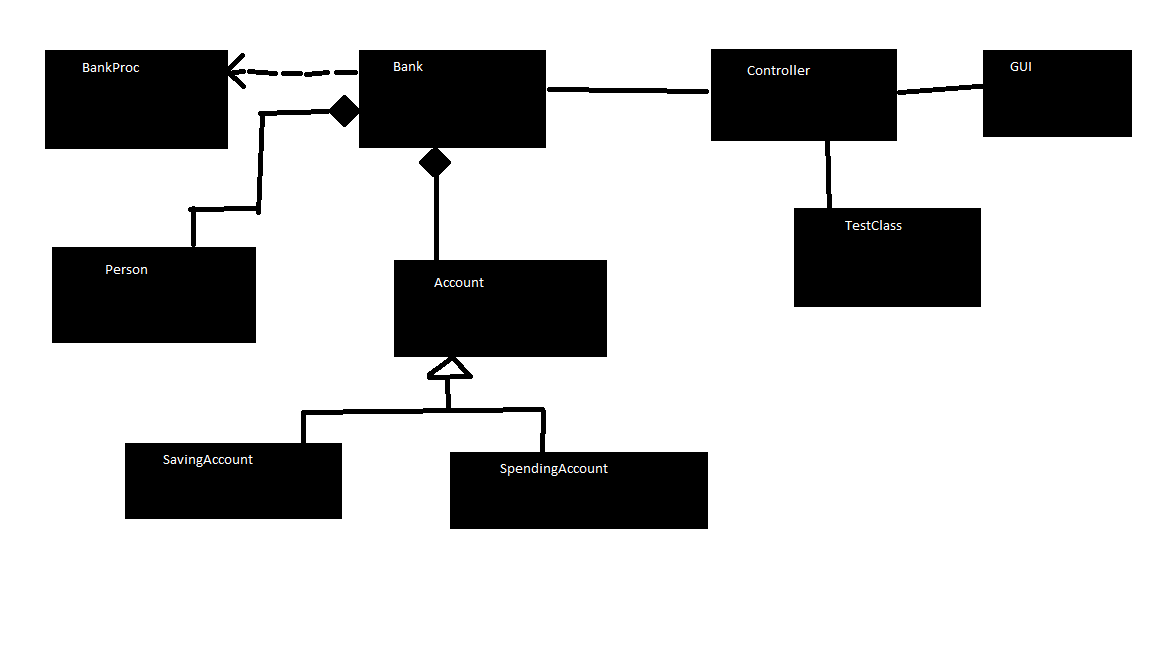
In al treilea rand observam cuvantul cheie, cel mai important din aceasta problema, “HashMap”. HashMap-ul se refera la construirea unei tabele de dispersie in limbajul de programare orientate pe obiect Java, unde sa mentinem obiecte de diverse tipuri. In cazul trebuie mentinute pe pozitiile tabelei obiectele de tipul Persoana, iar, folosind adresearea directa, se va prelungi fiecarei personae tabela pentru fiecare tip de cont pe care il are, deci o lista de obiecte de tipul contului respective. Vom reveni in capitolele ce urmeaza la detalii mai amanuntite.

Cazurile de utilizare sunt simple si se observa usor la folosirea aplicatiilor, iar in caz de erori se vor arunca exceptii si vor fi afisate, intr-un mesaj, instructiunile ce trebuiesc urmate pentru folosirea acelui caz care a transmis exceptia.

In cazul asumptiilor putem afirma ca pentru a folosi ATM-ul din aplicatie este necesara deschiderea unui cont pe numele unei personae, iar apoi folosirea bancomatului.

3. Proiectarea

In acest capitol vom prezenta detaliile de impartire a problemei in clase si subclase si legatura dintre ele. Vom reprezenta acest aspect prin diagram de clase UML



Dupa cum se observa in diagram de clase UML se poate observa existent unei interfete din Java, BankProc, care contine toate metodele ce trebuiesc neaparat construite de catre clasa principala Bank. In aceasta interfata se vor scrie deasupra metodelor, in comentarii, tehnica design by contracts. Pe langa aceste metode principale ce trebuie sa existe intr-o banca se mai pot define si alte metode ajutatoare, dar vom reveni la acestea in capitolul urmator.

Se observa, de asemenea, clasa Person care defineste o persoana reala, in relatie de agregare cu banca deoarece o banca nu poate exista fara client. De asemenea se observa si clasa Account care definieste elementele unui cont real care se afla tot in relatie de agregare, deoarece nu exista un cont fara o banca.

4. Implementare

In acest capitol vom prezenta in detaliu toate clasele si metodele utilizate in proiect, cu detalii tehnice si cu fiecare functionalitate a metodelor si claselor existente.

In primul rand observam clasa Person care contine setter si getteri, dar si un id unic care poate fi interpretat ca si un cnp unic. Acest cnp unic va fi dat direct de catre banca pentru utilizarea mai usoara a aplicatiei; acest lucru va fi facut automat prin folosirea unui id static care se incrementeaza automat de fiecare data cand se adauga un client nou bancii. Se foloseste design patternul Observable prin care trimitem notificari de la banca in momentul cand s-a schimbat ceva unui cont al unei personae.Acest lucru este folosit prin implementarea interfetei Observer si implementarea metodei update, care va afisa mesajul pe ecran in momentul schimbarii.

Urmatoarea calasa este clasa Account care defineste toate elementele unui cont: numarul unic al contului care este dat de asemenea automat de catre banca. Aceasta clasa extinde clasa Observable, iar in momentul schimbarii unui element la un cont se vor apela metodele setChanged() si notifyObservers() pentru notificarea observerului care exist ape acel cont. Metodele care sunt folosite sunt urmatoarele: constructorul contului cu un parametru prin care se seteaza suma initiala de bani, setter, getteri si o metoda incrase Money() prin care se incrementezaza sau decrementeaza suma de bani existent; aceasta utilizare se va folosi in momentul in care se vor adauga sau extrage bani de catre un client. De asemenea, aceasta este o metoda abstracta deoarece nu o instantiem niciodata cu ea insasi, ci o vom extinde prin clasele Saving Account si Spending Account care va contine toate elementele clasei Account si metodele ei, dar care vom avea particularitati diferite la finctionalitate. De asmeneea exista metodele abstracte deposit Money(), withdraw Money() si new Account() cu parametric, care vor fi implementate obligatoriu de subclasele care mostenesc clasa Account, si anume clasa Saving Account si clasa Spending Account. Aceasta impartire a clasei Account in cele 2 clase Saving Account si Spending Account se face datorita utilizarilor diferite ale celor 2 tiopuri de conturi si anume: contul de tip Saving Money este un cont pentru economii, sau un deposit mai bine zis din domeniul economic, iar acestuia ii va creste suma la momente diferite de timp, alese de catre banca cu o dobanda care se poate modifica ulterior; in cazul contului de tim Spending Account nu se va incrementa suma de bani exist ape cont, ci va ramane fixa tot timpul.

Urmatoarea clasa pe care o vom descrie este clasa Bank, clasa principala a acestui proiect, deoarece aceasta contine toate informatiile legate de personae si conturi pentru a spori gradul de Securitate in cazul spargerii de catre oameni cu intentii rele. Metodele definite sunt urmatoarele: constructorul clasei Bank care citeste din fisier datele tuturor persoanelor si conturilor la fiecare utilizare a aplciatiei si setarea id0ului static al conturilor si persoanelor la cel mai mare dintre ele. Metoda get Hash Map returneaza un obiect de tipul HashMap cu argumentele Person si o lista inlantuita de Account pentru utilizarea in G U I. Se implementeaza, de asemenea, metoda update pentru notificarea crearii sau stergerii unor conturi sau pentru ntificarea crearii sau stergerii unor personae din cadrul bancii. Metoda add Person este folosita pentru a adauga o persoana noua in cadrul bancii; aceasta se va adauga pe pozitia functiei de hash implicita data de limbajul de programare orientat pe obiect Java. In momentul in care se adauga o persoana noua se creeaza automat o lista inlantuita de Account care va fi goala la inceput. Metoda set Money() cu un parametru de tip double este folosita pentru calcularea si setarea dobandei tuturor conturilor de tip Saving Account. Aceasta va itera toate persoanele existente in hash si pentru fiecare persoana va itera toare conturile existente in lista de Account; se foloseste functia instanceof pentru identificarea cpnturilor de tip Saving Account, iar pentru acele conturi se va seta valoarea banilor din cont cu valoarea nou calculate cu tot cu dobanda. Metoda remove Person() cu parametru de tip Person va sterge o persoana si toate conturile associate ei din cadrul hashmapului. Aceasta metoda va itera taote persoanele existente in hash map si va compara id-urile lor cu id-ul persoanei date ca si arametru, iar daca sunt egale se va sterge persoana respective. Se vor putea sterge astfel personae unice din cauza utilizarii unui id unic pentru fiecare persoana. Urmatoarea metoda ete metoda add New Account() cu 2 parametri: Person si Account, care va adauga un cont droit de client persoanei respective care a facut cererea. Se vor itera toate persoanele din hash map si se vor compara id-urile proprii cu id-ul asociat persoanei date ca si parametru; la gasirea persoanei se va adauga la capatul listei inlantuite de conturi un cont de tipul cerut de client. De asemenea se va implementa intre aceasta persoana si contul respective un observer pentru notificarea persoanei respective daca s-a schimbat ceva la contul respective. Urmatoarea metoda este metda deposit Money() cu 3 parametri: person, money si id. Aceasta metoda va introduce in contul asociat persoanei date ca si parametru sumade bani data ca si parametru. Aceasta metoda va itera toate persoanele existente in hash map pan ava gasi persoana cautata. In acest moment se vor itera toate conturile associate persoanei pana la gasirea contului cu id-ul introdus de catre client; cand se va gasi contul respective, suma acestuia se va incrementa cu suma de bani transmisa ca parametru de client. Metoda withdraw Money() cu parametric person, money si id este folosita pentru retragere numerar al unui client; acesta va apasa butonul de pe interfata dup ace va complete campurile cerute, iar apoi metoda se va putea executa in conditii de siguranta fara a modifica functionarea corecta a programului; metoda itereaza toate persoanele din hash map iar cand va gasi persoana cu id-ul egal cu id-ul persoanei transmise ca si parametru se va opri, iterand toata lista inlantuita de conturi pana la gasirea contului cu id-ul egal cu id-ul transmis ca si parametru metodei; in acest moment while-urile se vor opri si se va executa metoda withdrawMoney() din clasa Account, care va extrage suma de bani data ca si parametru din suma totala a contului respectiv; in caz de eroare aceste metode vor arunca exceptii care se vor trata corespunzator de catre client, deoarece se va afisa un mesaj cu ce pasi trebuie urmati pt functionarea corecta a metodelor sip t extragerea in conditii de maxima siguranta a banilor din cont. Metoda get Person Accounts() este folosita pt a returna toate conturile detinute de o persoana transmisa ca si parametru; aceasta va itera toate persoanele din hash map, iar cand va gasi o persoana cu id-ul egal cu id-ul persoanei transmise ca si parametru se va opri; in acest moment se va returna lista inlantuita de tipul Account. Metoda get All Accounts() este folosita pentru a returna toate conturile existente in hash map; aceasta va itera toate persoanele din hash map si va adauga intr-o lista inlantuita auxiliara de tipul Account toate conturile gasite si iterate de la fiecare persoana; la sfarsit se va returna lista auxiliara de tipul Account. Metoda remove Account() e oflosita pt a sterge un cont al unei personae.Aceasta metoda va itera toate persoanele din hash map pan ace va gasi un id egal cu id-ul persoanei transmise ca si parametru; apoi se vor itera toate conturile persoanei respective pana cand va fi gasit contul cu id-ul egal cu id-ul cautat si transmis ca si parametru metodei; in acest moment, daca s-a gasit contul acesta se va sterge in conditii de maxima siguranta; in caz de eroare se vor arunca exceptii care va indruma clientul cum sa foloseasca corect si in conditii de maxima siguranta parteea aplicatiei ce foloseste aceasta metoda.

5. Testare

6. Rezultate

7. Concluzii

In concluzie, despre aceasta aplicatie putem afirma ca este o aplicatie banala si des intalnita si utilizata in domeniul economic pentru conturile bancare si pentru banci in general. De asemenea, pentru folosirea acestei aplicatii nu sunt necesare cunostinte avansate de progrmare sau de utilizare a calculatorului, deoarece este foarte simpla pentru utilizatori, iar in cazuri de eroare se afisaeaza un indrumator despre cum ar trebui folosita aplicatia in conditii corecte si de maxima siguranta.