ORDER MANAGEMENT

[1]Obiectivul temei

Scopul acestei teme este de a proiecta o aplicatie pentru procesarea comenzilor clientilor pentru un magazine(depozit).Pentru a stoca clientii,produsele si comenzile este folosita baza de date.

Aplicatia are o interfata grafica care include:

- O fereastra pentru operatii cu clientii:adauga,editeaza,sterge clientul
- O fereastra pentru operatii cu produsele:adauga,editeaza,sterge produsul
- O fereastra pentru crearea comenzilor:utilizatorul poate selecta un client,un produs si sa insereze cantitatea pentru produs pentru a plasa comanda valida.In cazul in care nu sunt suficiente produse,se afiseaza mesajul 'UNDER STOCK'.Dupa ce comanda este finalizata,stocul produsului este decrementat.

Sub-obiectivele temei sunt:

- Analizarea problemei si identificarea cerintelor:se iau in considerare scenariile de utilizare si cerinte functionale[2]
- Proiectarea aplicatiei pentru procesarea comenzilor:divizarea structurala(pachete),tehnnica "reflection",utilizarea de sablon architectural de tipul Layered[3]
- Implementarea aplicatiei de comenzi:prezentarea in detaliu a claselor componente[4]

[2]Analiza problemei

Caz de utilizare:Utilizatorul selecteaza tab-ul 'Order'

Scenariu principal reusit:

1. Utilizatorul selecteaza un rand din tabelul 'Client'
2. Utilizatorul selecteaza un rand din tabelul 'Product'
3. Utilizatorul insereaza in campul de text cantitatea produsului selectat,apasand butonul 'Place the order'
4. Se afiseaza in zona de text istoricul comenzilor al clientului current

Scenariu alternativ:

a) Utilizatorul selecteaza clientul si produsul,dar introduce o cantitate mai mare de produs decat exista pe stoc

b) Se afiseaza in zona de text mesajul 'UNDER STOCK'

c) Procesul se reia de la pasul 1

Caz de utilizare:Utilizatorul selecteaza tab-ul 'Client'

Scenariu principal reusit:

1. Utilizatorul introduce in campul de text pentru id un id existent in tabelul client
2. Utilizatorul apasa butonul ‘delete by id’ pentru a sterge clientul cu id-ul introdus
3. Tabelul se actualizeaza, iar clientul este sters din baza de date

Scenariu alternativ:

a) Utilizatorul introduce in campul de text pentru id un id care nu se afla in tabelul client
b) Utilizatorul apasa butonul ‘delete by id’ pentru a sterge clientul cu id-ul introdus
c) Apare o caseta de dialog care antentioneaza utilizatorul ca id-ul nu a fost gasit in tabelul ‘client’
d) Procesul se reia de la pasul 1

Caz de utilizare:Utilizatorul selecteaza tab-ul ‘Product’

Scenariu principal reusit:

1. Utilizatorul introduce date in campul de text pentru nume,stoc si pret
2. Utilizatorul apasa butonul ‘insert’
3. Tabelul se actualizeaza, iar produsul nou apare in tabel

Scenariu alternativ:

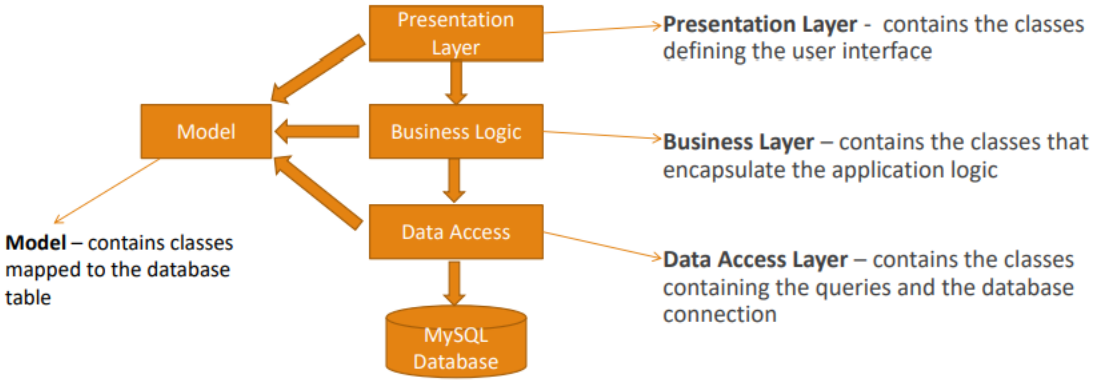
a) Utilizatorul introduce date in campul de text pentru nume,stoc si pret
b) Stocul introdus de utilizator nu este un numar natural nenul
c) Apare o caseta de dialog care antentioneaza utilizatorul ca stocul are format gresit
d) Procesul se reia de la pasul 1

Cerinte functionale:

- Aplicatia permite utilizatorului sa faca operatii de inserare,cautare,stergere,update pe tabele clientilor si a produselor
- Utilizatorul poate selecta un client,un produs si poate introduce cantitatea acestuia pentru a plasa comanda
- Aplicatia afiseaza continutul istoricului comenzilor al clientului selectat,astfel ca utilizatorul poate verifica daca comanda a fost plasata

[3]Proiectarea aplicatiei pentru comenzi

Divizarea in pachete- utilizarea sablonului architectural de tipul Layered



Aplicatia este divizata in diferite ‘straturi’.Fiecare strat are un rol si apeleaza functii din straturile de dedesupt.

Pachetul Model:contine clase bazate pe tabelele din baza de date. Aceste clase au aceleasi nume de atribute ca si campurile coloanelor din tabele.

Pachetul Presentation:contine clase care definesc interfata grafica

Pachetul BusinessLogic:contine clase care incapsuleaza logica aplicatiei

Pachetul DataAccess:contine clasele care realizeaza interogarile asupra bazei de date

Pachetul Connection:realizeaza conexiunea cu baza de date

Tehnica ‘reflection’:

Pentru ca tehnica ‘reflection’ ofera informatii despre clasa careia ii apartine un obiect si despre metodele ei, am utilizat-o in cadrul crearii celor 3 tabele:Client,Product,Order(in clasa Tables), si in cadrul interogarilor bazate pe inserare,cautare,stergere si update(in clasa AbstractDAO).

Tehnica reflction a fost utilizata la crearea metodelor care primesc o lista de obiecte si genereaza capul de table extragand prin ‘oglindire’ proprietatile obiectului si apoi populeaza tabelul cu valorile elementelor din lista.

```
public void generateTableReflection(List<T> list)
 {
     int rows=list.size();
     ArrayList<String> columnNames=new ArrayList<>();

     //creez HEADER-ul tabelului
     for (Field field : type.getDeclaredFields())
     {
         String fieldName = field.getName(); //gaseste numele atributului
curent
         columnNames.add(fieldName);
```

```
        }
        tableModel=new DefaultTableModel(columnNames.toArray(),rows);

        //populez tabelul

        int i=0,j=0; //i linie,j coloana
        for(T object:list)
        {
            for (Field field : type.getDeclaredFields())
            {
                String fieldName = field.getName();
                try {
                    PropertyDescriptor propertyDescriptor = new Proper-
tyDescriptor(fieldName, type);
                    Method method = propertyDescriptor.getReadMethod();
                    Object value = method.invoke(object);
                    tableModel.setValueAt(value.toString(),i,j);
                }
                catch (IllegalAccessException e) {
                    e.printStackTrace();
                } catch (IntrospectionException e) {
                    e.printStackTrace();
                } catch (InvocationTargetException e) {
                    e.printStackTrace();
                }
                j++;
            }
            j=0;
            i++;
        }
}
```

Tehnica reflectiei a mai fost folosita si la crearea unei clase generice care contine metode pentru accesarea bazei de date :crearea obiectului,editarea obiectului,stergerea obiectului si gasirea obiectului.Interogarile pentru accesarea bazei de date pentru un obiect anume care corespunde unui tabel va fi generat prin reflexie.

Exemplu stergere obiect:

```
public void insert(T t) {

    Connection connection = null;
    PreparedStatement statement = null;
    String query = createInsertQuery();
    try {
        connection = DatabaseConnection.getConnection();
        statement = connection.prepareStatement(query);

        int i = 1;
        for (Field field : type.getDeclaredFields()) {
            String fieldName = field.getName(); //gaseste numele atributului
curent

            if (fieldName.equals("id"))
                continue;
```

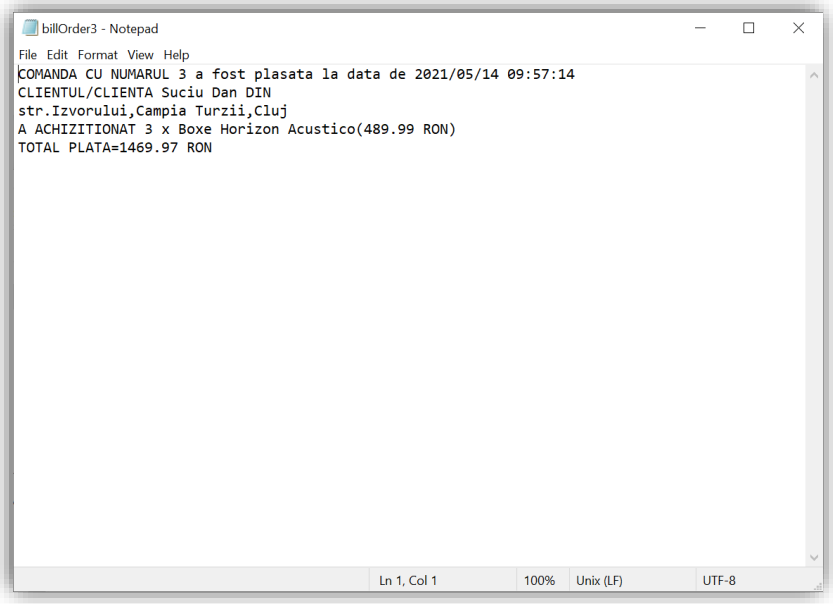
```
            PropertyDescriptor propertyDescriptor = new Proper-
tyDescriptor(fieldName, type);
            Method method = propertyDescriptor.getReadMethod();
            Object value = method.invoke(t); //gaseste getter si apeleaza-l
cu valoarea din bd

            statement.setObject(i, value);//inlocuiesc ? cu valoarea din obi-
ect de la campul curent

            i++;
        }

        statement.executeUpdate();

    } catch (SQLException | IntrospectionException e) {
        LOGGER.log(Level.WARNING, type.getName() + "DAO:insert " + e.getMes-
sage());
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    } catch (InvocationTargetException e) {
        e.printStackTrace();
    } finally {
        DatabaseConnection.closeStatement(statement);
        DatabaseConnection.closeConnection(connection);
    }

}
```

Interfata

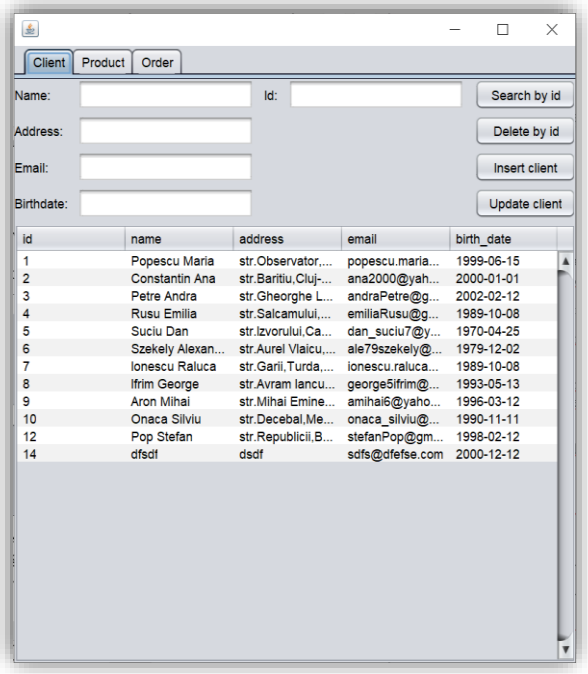
Este conceputa sa permita utilizatorului managementul clientilor,al produselor si al comenzilor.El poate adauga noi clienti,sterge cauta sau actualiza datele clientilor.La fel,poate insera noi produse,poate sterge,cauta sau actualiza produsele deja existente.

In cadrul tab-ului destinat comenzilor,utilizatorul poate plasa o comanda prin selectarea unui client din tabelul ‘Client’, selectarea unui produs din tabelul ‘Product’ si inserarea cantitatii produsului selectat.

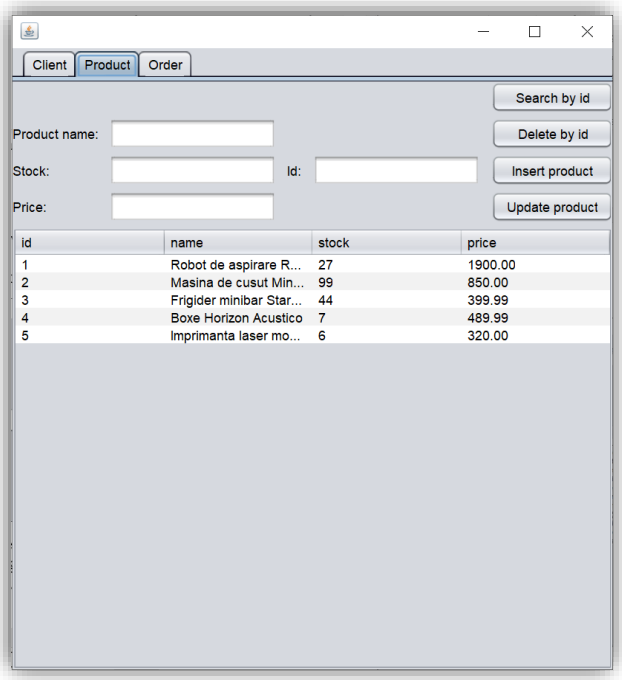
Ca rezultat, va aparea in caseta de text istoricul tuturor comenzilor clientului selectat, iar factura de plata pentru produsul actual comandat va fi tiparita intr-un fisier text cu numele “billOrderNo_order”.



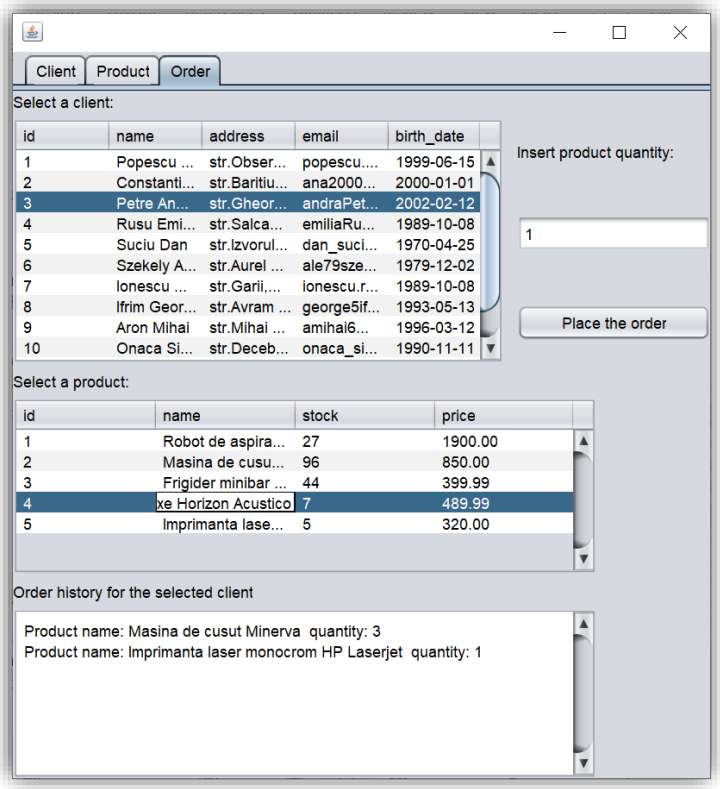
Tab-ul 'Client'



Tab-ul 'Product'



Tab-ul ‘Order’



[4]Implementarea aplicatiei pentru comenzi

Clase:

- Pachet Model:
    a) Client: Clasa retine datele despre un client:id,nume,adresa,email,data nasterii
    b) Produs: retine informatii despre un produs:id,nume,pret,stoc
    c) Comanda: detine datele unei comenzi:id,id client,id produs,cantitate
- Pachet DataAccess:
    a) AbstractDAO: Clasa generica care contine metode pentru accesarea bazei de date:creare obiect,update obiect,stergere obiect,gasire obiect.Are ca si parametru una dintre clasele Client,Product sau Order.
    b) ClientDAO: Aceasta clasa mosteneste AbstractDAO si precizeaza parametrul pentru clasa parinte ca fiind Client.
    c) ProductDAO: Aceasta clasa mosteneste AbstractDAO si precizeaza parametrul pentru clasa parinte ca fiind Produs.

d) OrderDAO: Aceasta clasa mosteneste AbstractDAO si precizeaza parametrul pentru clasa parinte ca fiind Order.

- Pachetul BusinessLogic:

a) ClientBusiness: Aceasta clasa face legatura intre Controller si ClientDAO. Mai exact,ceea ce se insereaza in interfata,in tabul Client,este preluat de Controller si transformat in obiect/lista de obiecte de tip Client de catre clientDAO.Toate metodele din ClientBusiness doar apeleaza metodele din clientDAO.

b) ProductBusiness: Aceasta clasa face legatura intre Controller si ProductDAO. Mai exact,ceea ce se intampla in interfata,in tabul Product,este preluat de Controller si transformat in obiect/lista de obiecte de tip Product de catre productDAO. Toate metodele din ProductBusiness doar apeleaza metodele din productDAO.

c) OrderBusiness: Aceasta clasa face legatura intre Controller si OrderDAO.

Mai exact,ceea ce se intampla in interfata,in tabul Order,este preluat de Controller si transformat in obiect/lista de obiecte de tip Order de catre orderDAO.Toate metodele din OrderBusiness doar apeleaza metodele din orderDAO.

- Pachetul Connection:

a) DatabaseConnection:realizeaza conexiunea dintre baza de date si IntelliJ

- Pachetul Presentation:

a) ClientTable: Aceasta clasa mosteneste Tables si precizeaza parametrul pentru clasa parinte ca fiind Client.

b) ProductTable: Aceasta clasa mosteneste Tables si precizeaza parametrul pentru clasa parinte ca fiind Product.

c) Controller:preia datele introduse de utilizator si le trimite claselor din BusinessLogic.

d) Tables: Aceasta clasa creeaza un tabel cu tipul dat ca parametru.

e) GenerateBill: Clasa se ocupa de tiparirea facturii,adica scrierea detaliilor de plata in fisier.

f) Interface: Aceasta clasa se ocupa de realizarea interfetei grafice.

Concluzii

Aceasta tema a adus cu sine noutati in ceea ce priveste tehnica ‘Reflection’ si divizarea aplicatiei in mai multe ‘straturi’.Arhitectura in ‘straturi’ faciliteaza modificarile ulterioare, adaugarea de noi componente,pentru ca noile schimbari nu vor afecta toata aplicatia,ci doar clase izolate.Tehnica ‘reflection’ a venit in ajutor cu lucrul cu clase,metode fara a fi nevoie sa se cunoasca dinainte numele acestora.

Ca dezvoltari ulterioare, aplicatia s-ar putea imbunatati prin introducerea posibilitatii ca utilizatorul sa poata adauga in comanda unui client mai multe produse o data.De asemenea, poate fi un efort cautarea

clientului/produsului dupa id, un plus ar fi cautarea clientului/produsului dupa nume urmand ca rezultatul afisat sa fie un mini-tabel cu toti clientii/toate produsele cu acelasi nume.

Bibliografie

StackOverFlow

Geeks for Geeks

Support Presentation