ASSIGNMENT 3

ORDERS MANAGEMENT

Studenta: Hirean Roxana-Maria

Grupa 30224

Cuprins

[Obiective 3](#__RefHeading___Toc2281_1065694278)

[Analiza problemei: scenarii și cazuri de utilizare 3](#__RefHeading___Toc2283_1065694278)

[Cerinte functionale 3](#__RefHeading___Toc2285_1065694278)

[Cazuri de utilizare 3](#__RefHeading___Toc2389_1065694278)

[Simularea modului de desfasurare a evenimentelor 3](#__RefHeading___Toc2438_1065694278)

[Proiectare 5](#__RefHeading___Toc2453_1065694278)

[Diagrama UML 5](#__RefHeading___Toc2455_1065694278)

[Implementare 6](#__RefHeading___Toc2465_1065694278)

[Clase 6](#__RefHeading___Toc2467_1065694278)

[Clasa Client 6](#__RefHeading___Toc2469_1065694278)

[Clasa Product 6](#__RefHeading___Toc2471_1065694278)

[Clasa Order 6](#__RefHeading___Toc2473_1065694278)

[Clasa ClientDA 6](#__RefHeading___Toc2469_1065694278)

[Clasa ProductDA 6](#__RefHeading___Toc2471_1065694278)

[Clasa OrderDA 6](#__RefHeading___Toc2473_1065694278)

[Clasa ClientBL 6](#__RefHeading___Toc2469_1065694278)

[Clasa ProductBL 6](#__RefHeading___Toc2471_1065694278)

[Clasa OrderBL 6](#__RefHeading___Toc2473_1065694278)

Connection

[Clasa ConnectionFactory 7](#__RefHeading___Toc2475_1065694278)

[Interfata Grafica 7](#__RefHeading___Toc2477_1065694278)

[Clasa View 7](#__RefHeading___Toc2481_1065694278)

[Clasa Controller 7](#__RefHeading___Toc2481_1065694278)

[Clasa Application 7](#__RefHeading___Toc2481_1065694278)

[Rezultate 8](#__RefHeading___Toc2485_1065694278)

[Dezvoltari ulterioare 8](#__RefHeading___Toc2487_1065694278)

**Obiective:**

Aceasta tema are ca scop principal proiectarea și implementarea unei aplicații gestionare a comenzilor clienților, iar bazele de date relationale sunt utilizate pentru a stoca produsele, clienții și comenzile. Aplicația este stucturata astfel încât sa facă cât mai ușoară plasarea comenzilor, dar și modificarea datelor din baza de date.

Obiectivul principal are în vedere indeplinirea mai multor obiective secundare. Acestea sunt:

1. Realizarea unei interfete grafice – Interfata are rolul de a realiza interacțiunea dintre utilizator si program, prin prelucrarea datelor introduse de catre utilizator, dar și de a prezenta rezultatul operatiilor de modificare a datelor efectuate de către utilizator.
2. Crearea unei baze de date care sa contina informatii referitoare la numele clientilor, adresa, denumirile produselor aflate in depozit, cantitatea si pretul corespunzatoare fiecarui produs, dar si informatii cu privire la comenzile plasate, respectand anumite restrictii( id ul trebuie sa fie unic, id ul produsului din comanda si cel al clientului care o plaseaza sa fie valide, cantitatea comandata sa fie mai mica decat stocul disponibil).
3. Afisarea intr un fisier .txt a facturilor tuturor comenzilor din baza de date. Acestea cuprind id-ul comenzii, al clientului, al produsului, data la care a fost plasata comanda, cantitatea de produs comandata, dar si suma pe care trebuie sa o plateasca clientul, aceasta fiind produsul dintre numarul de produse si pretul unui produs.
4. Realizarea conexiunii dintre aplicatie si baza de date creata.
5. Afisarea rezultatelor operatiilor de modificare a datelor in interfata grafica.
6. Actualizarea dupa plasarea fiecarei comenzi a numarului de produse din stoc, prin scaderea numarului de produse comandate de catre client.

Realizarea operatiilor de stergere, adaugare sau modificare a clientilor, respectiv a produselor, dar si afisarea tuturor produselor sau clientilor din baza de date, in functie de comanda aleasa.

**Analiza problemei: scenarii si cazuri de utilizare**

Principalul scop este de a implementa un program care executa corect operatiile specifice bazei de date, utilizand paradigmele Programarii Orientate pe Obiect . Pentru a realiza acest lucru, trebuie sa ne conectam la baza noastra de date, sa implementam operatii de baza cu baza de date, pentru a putea introduce in aceasta clienti, produse, comenzi si facturi, si multe alte operatii ( stergere, listare, editare ) .

De asemenea , trebuie sa folosim anumite tehnici precum “Java Reflection” pentru a crea o metoda care primeste o lista de obiecte si genereaza antetul tabelului , extragand proprietatile obiectului si apoi populeaza tabelul cu valorile elementelor din lista .

Primul pas pe care trebuie sa il facem pentru a putea implementa cu success programul , este sa ne conectam la baza de date . Aceasta conectare am realizat-o in pachetul “connection” , in clasa “ ConnectionFactory “.

Apoi , trebuie sa adaugam in pachetul “model“ , cate o clasa pentru fiecare tabel pe care il avem in baza de date ( atributele fiecarei clase trebuie sa fie exact de acelasi timp ca si coloanele din tabelele bazei de date ).

Mai departe , in pachetul “Data Access” , in clasa“AbstractDA “ , am implementat operatiile bazei de date cu ajutorul JavaReflection ( insert , delete , update ) . In acelasi pachet am mai definit clasele “ClientDA”,”ProductDA”,”OrderDA” , pentru operatii specific fiecarui tabel.

Cerinte functionale:

* Aplicatia trebuie sa prelucreze în mod corect datele introduse de utilizator.
* Aplicatia ar trebui sa ii permita utilizatorului sa selecteze orice operație din interfata grafica.
* Aplicația ar trebui sa efectueze cu succes orice operație selectata de utilizator.
* Aplicatia ar trebui să facă posibila vizualizarea datelor din baza de date relationala după orice operație.
* Aplicatia ar trebui sa afiseze într-un fisier .txt câte o factura pentru fiecare comanda efectuata.
* Aplicația ar trebui să aibă o interfata grafica cu un aspect plăcut dar care să fie și organizata și ușor de utilizat.

Cazuri de utilizare:

ADAUGAREA UNUI CLIENT IN BAZA DE DATE:

**Actor principal**: utilizatorul

**Scenariul principal de succes:**  
1.Utilizatorul selecteaza prima fereastra (“CLIENTI”),

2.Utilizatorul introduce id-ul clientului, numele si adresa acestuia.  
3. Utilizatorul apasa pe butonul pe care scrie ”adaugare”.

4. Aplicatia adauga in baza de date datele introduse de catre utilizator si afiseaza in consola interogarea utilizata.

5. Utilizatorul apasa pe butonul “vizualizare”.

6. Utilizatorul observa daca datele pe care le a introdus in interfata grafica au fost inserate corect in baza de date.

**Secvență alternativă**:

1. Date de intrare introduse gresit

1. Utilizatorul introduce texte care nu pot sa fie convertite in formatul aferent (din string in integer – contin si alte caractere in afara de cifre),
2. Utilizatorul introduce un ID care se afla deja in baza de date.
3. Utilizatorul introduce numere care nu sunt intregi sau sunt negative.

2. Este afisata in consola o exceptie cu un mesaj sugestiv pentru problema aparuta.

3. Se asteapta introducerea altor date .

4. Secventa alternativa se repeta pana la introducerea unor date valide, dupa care se revine la secventa din scenariul principal de succes.

STERGEREA UNUI CLIENT DIN BAZA DE DATE

**Actor principal**: utilizatorul

**Scenariul principal de succes:**  
1.Utilizatorul selecteaza prima fereastra (“CLIENTI”),

2.Utilizatorul introduce id-ul clientului, pe care doreste sa il stearga.  
3. Utilizatorul apasa pe butonul “Stergere”.

4. Aplicatia realizeaza stergerea din baza de date a informatiilor despre clientul cu id-ul selectat si afiseaza in consola interogarea utilizata.

5. Utilizatorul apasa pe butonul “vizualizare”.

6. Utilizatorul observa daca datele persoanei pe care a dorit sa o stearga au fost eliminate din baza de date.

**Secvență alternativă**: ID invalid

1. Utilizatorul introduce un id care nu se gaseste in baza de date sau introduce date care nu corespund formatului( introduce un sir de caractere care nu este un numar intreg).

2. Este afisata in consola o exceptie cu un mesaj sugestiv pentru problema aparuta.

3. Se asteapta introducerea altor date .

4. Secventa alternativa se repeta pana la introducerea unor date valide, dupa care se revine la secventa din scenariul principal de succes.

ACTUALIZAREA DATELOR UNUI CLIENT

**Scenariul principal de succes:**

1. Utilizatorul introduce id-ul clientului, a carui date doreste sa le modificea.  
3. Utilizatorulintroduce in text fieldurile corespunzatoare datele pe care doreste sa le inlocuiasca, noile valori pentru acestea.

4. Aplicatia realizeaza actualizarea informatiilor din baza de date ale clientul cu id-ul selectat

5. Prin apasarea butonului “vizualizare” se pot observa daca datele persoanei pe care a dorit sa le modificam au fost actualizare cu valorile corecte.

**Secvență alternativă**: ID invalid

1. Utilizatorul introduce un id care nu se gaseste in baza de date sau introduce date care nu corespund formatului( introduce un sir de caractere care nu este un numar intreg).

2. Este afisata in consola o exceptie.

3. Se asteapta introducerea altor date .

4. Secventa alternativa se repeta pana la introducerea unor date valide.s

Operatiile corespunzatoare pentru produse se realizeaza asemanator cu cele pentru clienti, diferenta fiind doar tipul datelor introduse.

ADAUGAREA UNUI PRODUS IN BAZA DE DATE

**Actor principal**: utilizatorul

**Scenariul principal de succes:**  
1.Utilizatorul selecteaza prima fereastra (“Produs”),

2.Utilizatorul introduce id-ul cprodusului, tipul lui si numarul de produse din magazin si pentru pretul produsului.  
3. Utilizatorul apasa pe butonul pe care scrie ”adaugare”.

4. Aplicatia adauga in baza de date datele introduse de catre utilizator si afiseaza in consola interogarea utilizata.

5. Utilizatorul apasa pe butonul “vizualizare”.

6. Utilizatorul observa daca datele pe care le a introdus in interfata grafica au fost inserate corect in baza de date.

**Secvență alternativă**:

1. Date de intrare introduse gresit

1. Utilizatorul introduce texte care nu pot sa fie convertite in formatul aferent (din string in integer – contin si alte caractere in afara de cifre),
2. Utilizatorul introduce un ID care se afla deja in baza de date.
3. Utilizatorul nu introduce o valoare pentru o coloana care nu poate sa aiba valori nule (de exemplu valoarea pentru id);
4. Utilizatorul introduce numere care nu sunt intregi sau sunt negative.

2. Este afisata in consola o exceptie cu un mesaj sugestiv pentru problema aparuta.

3. Se asteapta introducerea altor date .

4. Secventa alternativa se repeta pana la introducerea unor date valide, dupa care se revine la secventa din scenariul principal de succes.

STERGEREA UNUI PRODUS DIN BAZA DE DATA

**Actor principal**: utilizatorul

**Scenariul principal de succes:**  
1.Utilizatorul selecteaza prima fereastra (“PRODUS”),

2.Utilizatorul introduce id-ul produsului pe care doreste sa il stearga.  
3. Utilizatorul apasa pe butonul “Stergere”.

4. Aplicatia realizeaza stergerea din baza de date a informatiilor despre produsul cu id-ul selectat si afiseaza in consola interogarea utilizata.

5. Utilizatorul apasa pe butonul “vizualizare”.

6. Utilizatorul observa daca datele persoanei pe care a dorit sa o stearga au fost eliminate din baza de date.

**Secvență alternativă**: ID invalid

1. Utilizatorul introduce un id care nu se gaseste in baza de date sau introduce date care nu corespund formatului( introduce un sir de caractere care nu este un numar intreg).

2. Este afisata in consola o exceptie cu un mesaj sugestiv pentru problema aparuta.

3. Se asteapta introducerea altor date .

4. Secventa alternativa se repeta pana la introducerea unor date valide, dupa care se revine la secventa din scenariul principal de succes.

PLASAREA UNEI COMENZI

**Actor principal**: utilizatorul

**Scenariul principal de succes:**  
1.Utilizatorul selecteaza prima fereastra (“COMANDA”),

2.Utilizatorul selecteaza din primul combo box id-ul clientului pentru care doreste sa plaseze comanda.

3.Utilizatorul selecteaza din al doilea combo box id-ul produsului pe care doreste sa il comande.

4. Utilizatorul introduce o cantitate de ptoduse.  
5. Utilizatorul apasa pe butonul “Plasare”.

4. Aplicatia realizeaza adaugarea comenzii in baza de date.

5. Aplicatia actualizeaza datele din baza de date( a cantitatii produselor din depozit).

6. Aplicatia afiseaza in fisierul “save.txt” facturile comenzilor plasate.

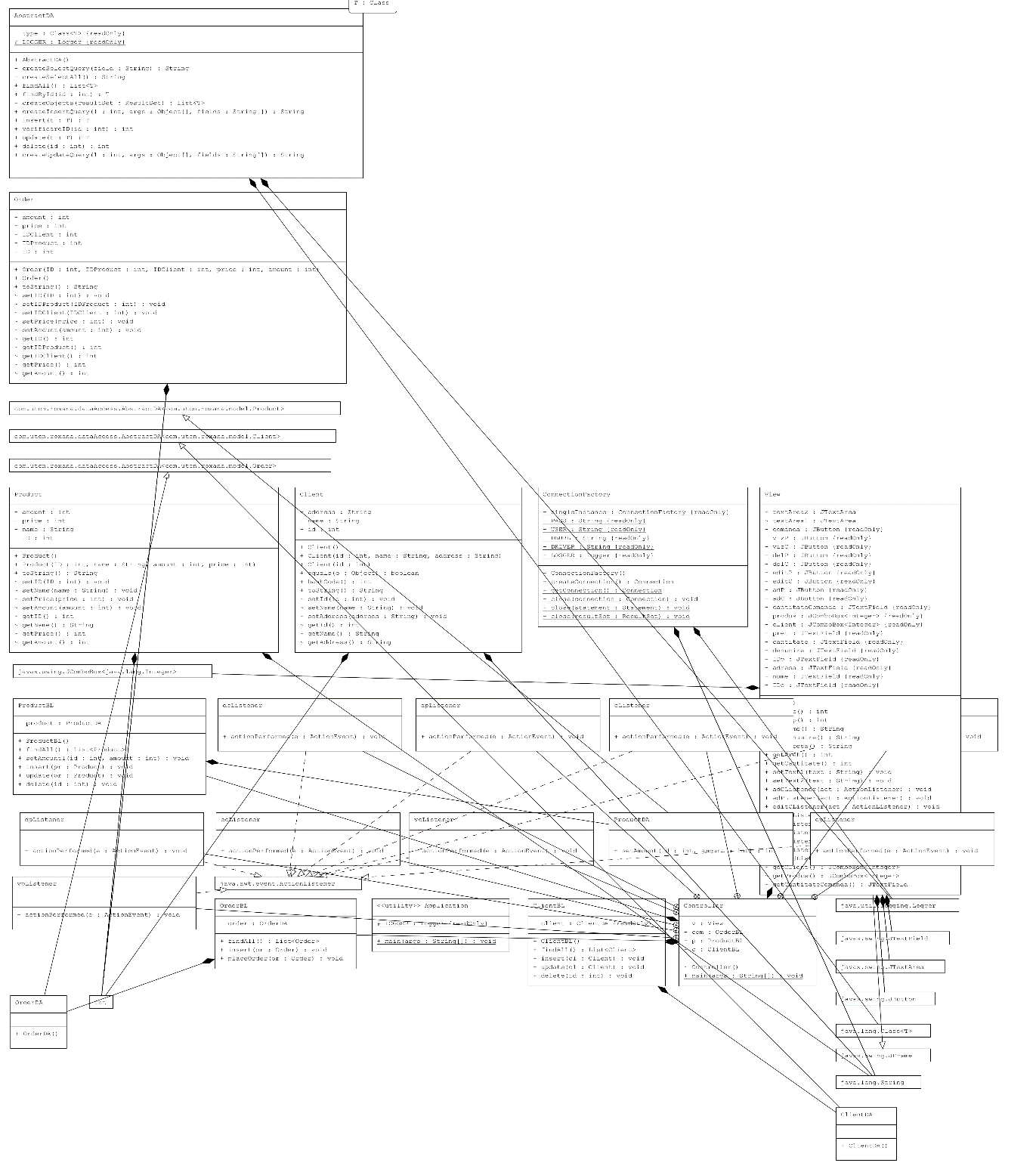
**Secvență alternativă**: Cantitate prea mare

1. Utilizatorul introduce o cantitate care este mai mare decat numarul produselor aflate in depozit.

2. Este afisata in consola o exceptie .

3. Se asteapta introducerea unordate valide ,dupa care se revine la secventa din scenariul principal de succes.

**Diagrama UML:**



**Tabele din baza de date Din MySQL:**

**Diagram

Description automatically generated**

**Proiectare:**

Respectand paradigmele programarii orientate pe obiect si folosind Layered Architecture, am impartit programul in mai multe pachete, fiecare continand mai multe clase. Voi prezenta mai jos structura fiecarui pachet:

• Pachetul ‘ connection ‘ care contine clasa : ConnectionFactory ;

• Pachetul ‘ dataAccess ‘ care contine clasele : AbstractDA , ClientDA, OrderDA, ProductDA ;

• Pachetul ‘ model ‘care contine clasele : Client , Order , Product;

• Pachetul ‘ businesslogic ‘ care contine clasele : ClientBL , OrderBL , ProductBL

• Pachetul ‘ viewandcontroller ‘ care contine clasa View, Controller .

Clasele Model:

Pachetul model cuprinde trei clase corespunzatoare fiecarui tabel din baza de date :Client, Comanda, si Produs. Fiecare clasa are argumente corespunzatoare datelor din fiecare tabel, dar si tipurilor acestora. Clasa Client contine id-ul, numele si adresa clientilor; clasa Comanda contine informatii cu privire la id-ul comenzii, data plasarii, clientul care a plasat-o, produsul comandat de client, numarul de produse comandate si suma totala pe care trebuie sa o plateasca clientul. Clasa produs retine in variabilele sale informatii legate de id-ul produsului, denumirea acestuia, numarul de produse din acel tip aflate in depozit, precum si pretul produselor.

Pe langa acestea, clasele mai contin ci constructori dar si metode de acces la date: metode pentru a atribui o valoare unui atribut privat sau a returna valoarea unui astfel de atribut.

Clasa Connection:

Pachetul Connection contine o singura clasa,si anume “ConnectionFactory” care are un rol semnificativ, fiind reponsabila de realizarea legaturii dintre aplicatie si baza de date . Are in componenta sa definite un driver, o adresa url, un string care reprezinta numele de utilizator si altul pentru memorarea parolei. Este utilizata arhitectura singleton pentru a declara o singura instanta de tipul acestui obiect, astfel ne asiguram ca avem o singura conexiune la baza de date care este inchisa dupa fiecare operaite. Mai sunt definite si metode pentru a inchide legatura cu baza de date: metoda supraincarcata close cu diferite tipuri de parametrii.

Clasele Data Access:

Cea mai importanta clasa din acest pachet este clasa generica abstractDA, in cadrul careia s-a folosit reflectia pentru a implementa metodele corespunzatoare tuturor claselor din model. Aceasta clasa contine petode pentru a creea o interogare pentru selectarea instantelor cu un anumit id, clasa “Find by id” care realizeaza aceasta interogare si returneaza rezultatul obtinut sub forma unui obiect de tipul corespunzator clasei care o extinde, metoda create object care transforma o rezultatul unei interogari care returneaza mai multe randuri intr-o lista de obiecte corespunzatoare si metodele implementate corespunzatoare ascultatorilor butoanelor din interfata grafica: inser, update, delete, si find all. Pe langa aceasta clas, pachetul mai contine si alte clase care extind clasa generica(clientDa, ComandaDA, si produsDA).

Clase ViewAndController:

Contine clasele care implementeaza interfata grafica, acestea fiind clasa View si clasa Controller, implementand o arhitectura de tip mvc, modelul fiind prezent in alt pachet separat. Clasa View realizeaza interfata cu tuilizatorul, continand trei ferestre, in fiecare gasindu se text fielduri sau combo boxuri pentru introducerea sau selectarea datelor si butoane corespunztoare operatiilor efectuate de catre aplicatie. Clasa controller are rolul de a face legatura dintre model si view avand ca argumente de tipul view, client, produs si comanda. Aceasta clasa implementeaza ascultatorii definiti in view , apland metodele implementate in clasele de tip model.

**Rezultate:**

Au fost efectuate mai multe testari cu rezultat pozitiv, al caror rezultat a fost vizibil in interfata grafica, dat si in fisierul save.txt, continand descrierea evenimentelor din fiecare moment, dar si rezultatul simularii sub forma unei statistici.

**Dezvoltari ulterioare:**

Acest proiect poate sa fie dezvoltat prin marirea intervalului de valori pe care il pot lua atributele, sau chiar prin punerea in practica intr-un magazin adevarat.