DOCUMENTATIE

TEMA 3

**GESTIONAREA COMENZILOR**

NUME STUDENT: Sand Elena - Andreea

GRUPA: 30223

# CUPRINS

[1. Obiectivul temei 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297886)

[3. Proiectare 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297887)

[4. Implementare 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297888)

[5. Rezultate 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297889)

[6. Concluzii 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297890)

[7. Bibliografie 3](file:///C:\Users\Utilizator\Desktop\PT2021-2022_Documentation_Template.doc#_Toc95297891)

# 1.Obiectivul temei

Obiectivul acestei teme este implementarea unei aplicatii care faciliteaza gestionarea comenzilor unui depozit de produse. Aplicația a fost concepută pentru a fi utilizată de unul dintre membrii personalului depozitului si pune la dispozitie întregul set de operațiuni, atât pe clienți , cat si pe produse, într-o singură vizualizare printr-o interfață grafică. Interfața grafică facilitează munca operatorului aplicației prin faptul că este intuitiva și pune la dispoziție întregul ansamblu de operațiuni care ar trebui efectuate asupra depozitului. Aceste includ: adăugarea, ștergerea, actualizarea sau vizualizarea datelor unui client; actualizarea cantității unui produs,actualizarea prețului unui produs; operațiuni de gestionare a comenzilor.

# 2.Analiza problemei, modelare, scenarii, cazuri de utilizare

**2.1 Analiza problemei**

Principalul motiv pentru care s-a folosit limbajul Java pentru implementarea acestei aplicații este necesitatea respectarii celor patru elemente de bază ale programării orientate pe obiecte care sunt: ​​încapsularea, abstracția, moștenirea și polimorfismul. Prima, paradigma încapsulării se referă la conceptul care leagă împreună datele și funcțiile care manipulează datele și care le păstrează pe ambele în siguranță de interferențele din exterior și utilizarea necorespunzătoare. Încapsularea este importanta si se refera la conceptul de ascundere a datelor. Paradigma de abstractizare este una dintre conceptele cheie ale POO, si se refera la selectarea datelor dintr - un grup mai mare pentru a afișa doar detaliile relevante pentru obiect. Obiectele din OOP pot fi găsite în viața reală, dar cu mai puține proprietăți. Paradigma moștenirii este folosită pentru a crea ierarhii de clase care au în comun un set de atribute și metode, astfel încât o clasă poate fi derivată din altă clasă. Acea clasa se numește superclasă, în timp ce clasa derivată se numește subclasă sau clasă copil. Acest va evita duplicarea codului. Ultima paradigmă OOP, polimorfismul este folosita de obicei pentru suprascrierea unei metode, deci va avea același nume, dar va avea o alta funcționalitate într-o subclasă.

**Cerintele functionale ale acestui proiect sunt:**

• Aplicatia trebuie sa prelucreze în mod corect datele introduse de utilizator.

• Aplicatia ar trebui sa i 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.

**Cerintele non-functionale ale acsetui proiect sunt:**

• 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 operatiilr de modificare a datelor efectuate de către utilizator.

• 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).

• 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.

• Realizarea conexiunii dintre aplicatie si baza de date creata.

• Afisarea rezultatelor operatiilor de modificare a datelor in interfata grafica.

• 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.

**2.2 Modelare:**

A fost necesar să se găsească o soluție de stocare pentru datele de introdus program. În proiect am folosit o conexiune cu o bază de date MySQL unde păstrez 3 tabele : client , produs si comenzi. Aceste 3 tabele vor stoca toate datele care sunt utilizate sau generate ca urmare a comenzilor clienților. În proiectarea teoretică a acestei aplicații, am dedus unele necesitati pentru dezvoltarea și implementarea practică a acestui proiect. În primul rând, pentru a testata și exemplifica utilitatea acestei aplicații, este necesar un set inițial de produse, dar nu este necesar un set inițial de comenzi. În al doilea rând, având în vedere numărul de operațiuni care trebuiesc implementate și fluxul de date care trebuie gestionat, este recomandat sa existe o interfață detaliată pentru a facilita utilizarea corectă.

**2.3 Scenarii, cazuri de utilizare**

Simularea modului de desfasurare a evenimentelor

**Cazuri de utilizare**

▪**Adaugarea unui client in baza de date**

**Actor principal**: utilizatorul

**Scenariul principal de succes**:

1.Utilizatorul selecteaza prima fereastra (“Client Operations”),

2.Utilizatorul introduce id-ul clientului, numele si adresa si numarul de telefon al acestuia.

3. Utilizatorul apasa pe butonul pe care scrie ”add”.

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

a) Utilizatorul introduce texte care nu pot sa fie convertite in formatul aferent (din string in integer – contin si alte caractere in afara de cifre),

b) Utilizatorul introduce un ID care se afla deja in baza de date.

c) Utilizatorul nu introduce o valoare pentru o coloana care nu poate sa aiba valori nule (de exemplu valoarea pentru id);

d) 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 (“Client Operations”),

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

a) Utilizatorul introduce texte care nu pot sa fie convertite in formatul aferent (din string in integer – contin si alte caractere in afara de cifre),

b) Utilizatorul introduce un ID care se afla deja in baza de date.

c) Utilizatorul nu introduce o valoare pentru o coloana care nu poate sa aiba valori nule (de exemplu valoarea pentru id);

d) 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 (“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 “a.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.

# 3.Proiectare

Aplicatia contine 16 clase impartite in 5 pachete:

* Connection: - ConnectionWarehouse
* Dao: - AbstractDao

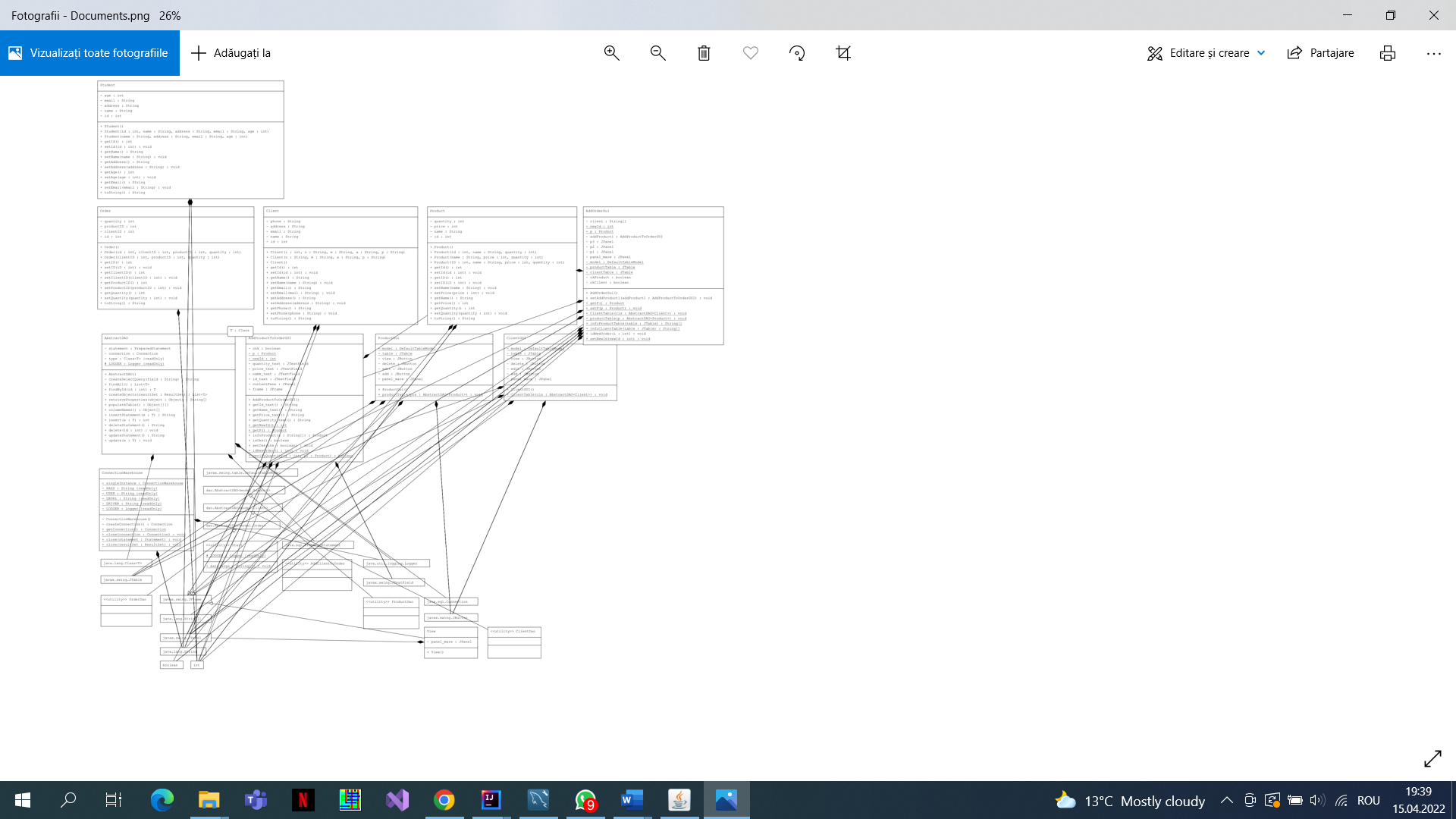
- ClientDao

- OrderDao

- ProductDao

* Model: - Client
* Product
* Order
* Presentation: -AddOrderGui
* AddProductToOrderGui
* ClientGUI
* ProductGui
* View
* Controller
* Start: - Start

**Diagrama de clase:**



# 4.Implementare

### **Clasele model**

Pachetul model cuprinde 3 clase corespunzatoare fiecarui tabel din baza de date :**Client, Order si Product.** 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 sar si suma totala pe care trebuie sa o plateasca clientul. Clasa produs retinr 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.

### **Clasele pentru conectare**

Pachetul Connection contine o singura clasa,si anume “**ConnectionWarehouse**” 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 operatie. In metodele Create connection si get connection se realizeaza legatura cu baza de date utilizant adresa url, numele de utilizatoe si parola definite, iar in get connection este returnata aceasta legatura. Mai sunt definite si metode pentru a inchide legatura cu baza de date: metoda supraincarcata close cu diferite tipuri de parametrii.

### **Clasele de acces la date**

Cea mai importanta clasa din acest pachet este clasa generica **AbstractDAO**, in cadrul careia s a folosit reflectia pentru a implementa metodele corespunzatoare tuturor claselor din model. Aceasta clasa contine metode pentru a creea o interogare pentru selectarea instantelor cu un anumit id, metodele insertStatement, deleteStatement, updateStatement care realizeaza interogarile 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, metoda verificare id care verifica daca id-ul introdus se gaseste sau nu in baza de date, si metodele implementate corespunzatoare ascultatorilor butoanelor din interfata grafica: inser, update, delete, si view. Pe langa aceasta clase, pachetul mai contine si alte clase care extind clasa generica(ClientDao, OrderDao, si ProductDao).

### **Clasa pentru prezentare**

Contine clasele care implementeaza interfata grafica, acestea fiind clasa View, ClientGui, ProductGui, AddOrderGui si AddProductToOrderGui, implementand o arhitectura de tip mvc. Clasa View realizeaza interfata cu utuilizatorul, continand trei butoane : Client Operations, Product Operations si ProductOrder, la apasarea carora se deschide cate o feresatra noua cu operatiile specific.

# 5.Rezultate

Deoarece problema a fost rezolvată constructiv, testarea funcțională a fost efectuate imediat după implementarea funcţiilor. Intregul set de operații a fost verificat dupa implementare. Dacă se încearcă un apel neadecvat, aplicația va returna o notificare către utilizatorul specificând eroarea exactă de introducere. Pentru forma corecta a intrarilor, aplicația funcționează corect.

# 6.Concluzii

În analiza, proiectarea și implementarea acestui proiect, am revizuit concepte de baza ale programarii orientate pe obiecte și m-am familiarizat cu mai multe elemente existente ale limbajului cum ar fi Reflectia si conectarea la o baza de date MySQL.

# 7.Bibliografie

* <https://www.baeldung.com/java-jdbc>
* <http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/>
* <https://dzone.com/articles/layers-standard-enterprise>
* <http://tutorials.jenkov.com/java-reflection/index.html>
* <https://www.baeldung.com/javadoc>
* https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html