

**Tehnici de programare**

**Documentatie Tema2**

**ORDERS MANAGEMENT**

Muresan Ioana Danina

Grupa 30226

Cuprins:

1.Obiectivul temei

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

3.Proiectare(decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete , relatii, packages, algoritmi, interfata utilizator)

4.Implementare

5.Rezultate

6.Concluzii

7.Bibliografie

**1. Obiectivul temei**

Cerinta proiectului a fost: Implementati o aplicatie de gestionare a comenzilor si pentru procesarea comenzilor clientilor pentru un depozit. Bazele de date relational ear trebui sa fie utilizate pentru a stoca produse, client si comenzi. Aplicatia trebuie implementata in conformitate cu modelul arhitecturii stratificate si ar trebui sa utilizeze (minim) urmatoarele clase:

• Clase de model - reprezintă modelele de date ale aplicației

• Clasele Business Logic - conțin logica aplicației

• Clase de prezentare – clase legate de GUI

• Clase de acces la date - clase care conțin acces la baza de date

Aceasta tema foloseste bazele de date relationare , in acest caz MySQL . Baza de date este o colecţie organizată de informaţii Baza de date pentru a putea fi actualizata, modificata, utilizata in obtinerea de rapoarte si statistici, se afla sub controlul unui program. Întreg acest ansamblu poartă numele de SGBD (Sistem de Gestionare a Bazelor de Date), sau în engleză DBMS (Date Base Management Sistem). In practica prin baza de date se intelege de multe ori si softul aferent, iar prin SGBD un sistem integrat in care se pot construi si interoga baze de date . Baze de date RELAŢIONALE - înmagazinează datele în tabele care se pot lega logic după valorile anumitor coloane - relaţia dintre câmpuri realizează asocierea explicită - este un model simetric, uniformitatea reprezentării datelor determină uniformitate în mulţimea operatorilor - bazat pe teoria matematică a relaţiilor, proiectarea bazelor de date poate fi tratată algoritmic - utilizeaza un limbaj standardizat de interogare a bazei de date numit SQL .

• Obiectivul principal

Proiectarea si implementarea unei aplicatii de gestionare a comenzilor clientilor pentru un depozit

• Subobiective

Analizați problema și identificați cerințele .

Proiectați aplicația de gestionare a comenzilor .

Implementarea aplicatiei de gestionare a comenzilor .

Testați aplicația de gestionare a comenzilor .

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

**· Analiza**

**Use Case:** Principalul obiectiv ar putea fi adaugarea unui produs .

**Primary Actor**: Un angajat doreste sa introduca un produs in baza de date .

**Main Success Scenario**: Angajatul selectează opțiunea de a adăuga un produs nou. Aplicatia va afisa un formular in care se insereaza detaliile produsului . Angajatul introduce numele produsului, prețul acestuia și stoc current . Angajatul face clic pe butonul "Adăugați" . Aplicația stochează datele despre produse în baza de date și afișează un mesaj de confirmare .

**Alternative Sequence**: : Valori nevalide pentru datele produsului . Utilizatorul introduce o valoare negativa pentru stocul produsului. Aplicația afișează un mesaj de eroare și solicită utilizatorului să introduca un stoc valid . Scenariul revine la pasul 3 .

**· Modelarea**

Diagram

Description automatically generated

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 “ dao ” , in clasa “ AbstractDAO “ , am implementat operatiile bazei de date ( insert , delete , edit , list ) .

**· Scenarii**

Pentru ca programul sa poata rula corect , datele de intrare trebuie sa urmareasca un anumit tipar , astfel ca , ar trebui sa arate astfel comenzile :

· pentru inserarea clientului – “Insert client: Nume , adresa, email id ”;

· pentru inserarea produsului – “Insert Product: Nume, cantitate, pret id”;

· pentru stergerea clientului – “Delete client: Id client”;

· pentru stergerea produsului – “Delete Product: Id produs”;

· pentru comanda – “ Order: Nume produs, nume client, cantitatea dorita, pretul unui produs”;

In cazul generarii comenzii , daca cantitatea de produs dorita este mai mare decat cantitatea de produs exista in stoc , se va genera un mesaj de eroare .

**· Cazuri de utilizare**

Atunci cand se apasa butonu ‘Run’, se deschide interfata grafica , care este usor de utilizat. Apar 3 butoane: Clients , Products , Orders . Utilizatorul va apasa butonul aferent operatiei pe care doreste sa o realizeze si va deschide unul dintre cele 3 frame – uri.

In cadrul frame – ului ‘ Clients ‘ utilizatorul poate adauga un client nou , sterge un client , edita detaliile unui client sau poate vedea lista cu toti clientii .

In cadrul frame – ului ‘Products ‘ utilizatorul poate adauga un produs nou , sterge un produs existent , edita detaliile unui produs sau poate vedea lista cu toate produsele.

In cadrul frame – ului ‘Orders ‘ utilizatorul poate vedea lista cu toate comenzile , poate adauga o comanda si poate crea o factura .

**3.Proiectare(decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete , relatii, packages, algoritmi, interfata utilizator)**

Pentru a respecta paradigmele POO, programul a fost impartit in mai multe pachete cu nume suggestive care contin diferite clase:

-Pachetul Start : Contine doar clasa Main

-Pachetul Presentation : Contine clasele : View , ViewClient, ViewOrder ,ViewProduct

-Pachetul Model: Contine clasele : Bill ,Client,Order,Product

-Pachetul dataAcces : Contine clasele: AbstractDAO , ClientDAO , OrderDAO ,ProductDAO

-Pachetul Connection : Contine clasa ConnectionFactory

-Pachetul BusinessLogic: Contine la randul sau un alt pachet, Validators ,care contine clasele EmailValidator si QuantityValidator, dar si clasele ClientBLL, OrderBLL, ProductBLL .

**Diagrama UML a proiectului:**

Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora ( numite și obiecte ). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Așa se face că există aplicații ale UML-ului pentru management de proiecte, pentru business Process Design etc.

Graphical user interface

Description automatically generated

Interfata grafica cuprinde urmatoarele elemente:

· Frame = “rama” in care se adauga toate elementele de care avem nevoie pentru buna functionare a programului . E practic o fereastra care , dupa utilizare , apasand butonul ‘X’ din dreapta sus se poate inchide ( Exit\_on\_close ) ;

· Butoane – avem 3 butoane in frame – ul principal fiecare cu alta functionalitate :

CLIENTS – deschide frame - ul pentru client care contine la randul lui alte butoane cu operatiile care se pot realiza asupra tabelei ‘ clients ‘ ;

PRODUCTS – deschide frame – ul pentru produse care contine la randul lui alte butoane cu operatiile care se pot realiza asupra tabelei ‘ products ‘ ;

ORDERS – deschide frame – ul pentru comenzi care contine la randul lui alte butoane cu operatii care se pot realiza asupra tabelei ‘ orders\_table ‘ ;

· TextField = spatii dreptunghiulare in care se pot introduce date de la tastatura , dar pe langa asta pot fi folosite si pentru a afisa rezultatul fara a se putea introduce date de la tastatura .

· Label = o etichetă , care poate fi titlul , informații , indicații etc . , care pot ajuta utilizatorul să folosească progamul. In cazul nostru e folosit doar pentru afisarea informatiilor aflate deasupra casetelor de text .

**4.Implementare**

**Pachetul Presentation:**

**Clasa View**: · Butoane – avem 3 butoane in frame – ul principal fiecare cu alta functionalitate :

CLIENTS – deschide frame - ul pentru client care contine la randul lui alte butoane cu operatiile care se pot realiza asupra tabelei ‘ clients ‘ ;

PRODUCTS – deschide frame – ul pentru produse care contine la randul lui alte butoane cu operatiile care se pot realiza asupra tabelei ‘ products ‘ ;

ORDERS – deschide frame – ul pentru comenzi care contine la randul lui alte butoane cu operatii care se pot realiza asupra tabelei ‘ orders\_table ‘

**Clasa ClientView**

Contine 4 butoane: pentru adaugare client,pentru editare client, pentru stergere client si pentru a vedea toti clientii intr-un tabel . De asemenea , mai contine 4 JTextField-uri pentru a putea introduce un nume, o adresa ,un email si un id. De asemenea,mai exista un buton de revenire la frame-ul principal.

**Clasa ClientProduct**

Contine 4 butoane: pentru adaugare produs,pentru editare produs, pentru stergere produs si pentru a vedea toate produsele intr-un tabel . De asemenea , mai contine 4 JTextField-uri pentru a putea introduce un nume, o cantitate ,un pret si un id. De asemenea,mai exista un buton de revenire la frame-ul principal.

**Clasa ClientOrder**

Contine 4 butoane: pentru adaugare comanda,pentru a genera o factura si pentru a vedea toate comenziile intr-un tabel . De asemenea , mai contine 5 JTextField-uri pentru a putea introduce un nume client,un nume produs, o cantitate ,un pret si un id. De asemenea,mai exista un buton de revenire la frame-ul principal.

**Pachetul Model**

**Clasa Product** : contine atributele unui produs, constructor si metode de getter si setter.

**Clasa Client**: contine atributele unui client, constructor si metode de getter si setter.

**Clasa Order** : contine atributele unei comenzi, constructor si metode de getter si setter.

**Clasa Bill** : contine o metoda generateBill(Order order) care scrie intr-un fisier .txt informatii relevante despre o anumita comanda, trimisa ca si parametru.

**Pachetul dataAcces**

**Clasa AbstractDAO** : aceasta clasa este una generica , care contine metode pentru a aceesa baza de date : crearea unui obiect , editarea unui obiect , stergerea unui obiect si gasirea unui obiect ;

**Clasa ClientDAO** : aceasta clasa extinde clasa ‘ AbstractDAO ‘ , implementeaza metodele ei si in plus mai adauga metode :

* Metoda ‘ findById ‘ care gaseste un client dupa id – ul sau ;
* Metoda ‘ findByName ‘ care gaseste un client in baza de date dupa numele sau;
* Metoda ‘ deleteClien t’ care sterge un client din baza de date ;
* Metoda ‘ updateClient ‘ care actualizeaza informatiile referitoate la un client;
* Metoda ‘ listAllClients ‘ care afiseaza toti clientii din baza de date ;

**Clasa OrderDAO** : aceasta clasa extinde clasa ‘ AbstractDAO ‘ , implementeaza metodele ei si in plus mai adauga metode :

* Metoda ‘ findById ‘ care gaseste o comanda dupa id – ul ei ;
* Metoda ‘ insertOrder ‘ care creeaza o noua comanda in baza de date ;
* Metoda ‘ listAllOrders ‘ care afiseaza toate comenzile din baza de date ;

**Clasa ProductDAO**  : aceasta clasa extinde clasa ‘ AbstractDAO ‘ , implementeaza metodele ei si in plus mai adauga metode :

* Metoda ‘ findByName ‘ care gaseste un produs dupa numele lui ;

Metoda ‘ deleteProduct ‘ care sterge un produs existent din baza de date ;

* Metoda ‘ updateProduct ‘ care actualizeaza informatiile referitoare la un produs existent in baza de date ;
* Metoda ‘ listAllProducts ‘ care afiseaza toate produsele existente in baza de date ;

**Pachetul connection** :

**Clasa ConnectionFactory** : aceasta clasa contine un obiect cu o singura instant ( ceea ce inseamna ca putem instania un singur obiect de acest tip – ne putem conecta la o singura baza de date la un moment dat ) . Aceasta clasa contine metode pentru creearea unei conexiuni , oferind o conexiune active si , in final , inchizand conexiunea , Statement – ul si ResultSet – ul .

**Pachetul businessLogic**:

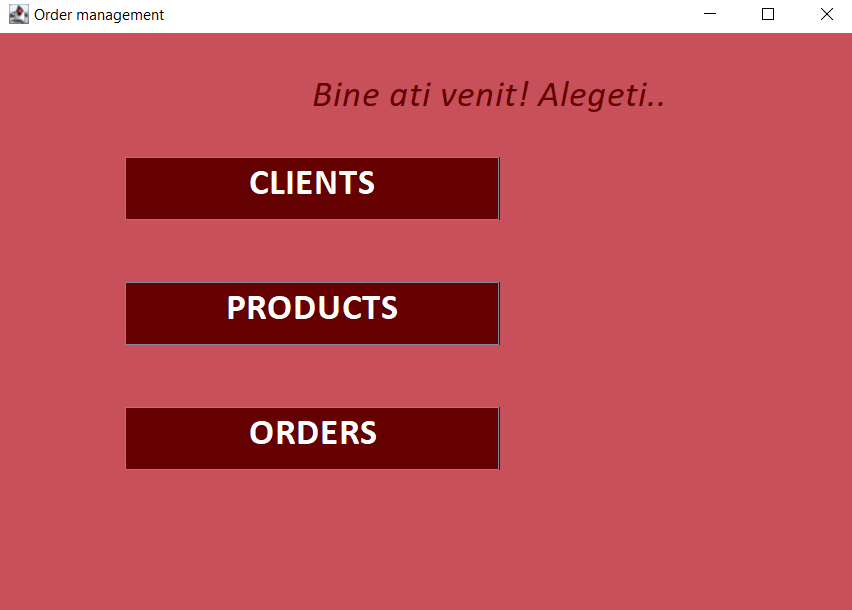
**Clasa EmailValidator**: verifica daca un email e valid( Contine un @ si text inainte si dupa el)

**Clasa QuantityValidator**: verifica daca o cantitate e valida( nu poate fi negative)

**Clasa ClientBLL,OrderBLL si ProductBLL** folosesc metodele din pachetul dataAcces pentru a face operatii pe produse,client si comenzi.

**5.Rezultate**

Vom putea vedea toate rezultatele prin intermediul interfetei grafice.









**6.Concluzii**

In concluzie , sunt de parere ca acest proiect mi-a aprofundat cunostiintele in tot ce inseamna limbajul Java , implementarea paradigmelor POO , creearea unui program cu o interfata grafica . De asemenea , m-a ajutat sa imi reamintesc tehnicile de programare invatate in semestrul trecut . În același timp, a avut un rol important și în aprofundarea cunoștințelor despre liste. Nu în ultimul rând, a avut un impact pozitiv si m-a ajutat sa invat lucrul cu bazele de date.

**7.Bibliografie**

1.StackOverflow

2. <http://tutorials.jenkov.com/java-reflection/index.html>

3. Wikipedia