DOCUMENTATIE TEMA 3

MAGAZIN

Nume si prenume: Fagadar Ionela-Catalina

Grupa: 30226

Profesor Laborator: Marin Oana

Contents

[**1.Obiectivul temei** 3](#_Toc6419295)

[**2.Analiza problemei, modelare, scenarii, cazuri de utilizare** 3](#_Toc6419296)

[**3.Proiectare** 4](#_Toc6419297)

[**3.1. Decizii de proiectare** 4](#_Toc6419298)

[**3.2. Diagrame UML** 4](#_Toc6419299)

[**3.3. Structuri de date** 5](#_Toc6419301)

[**3.4. Proiectare clase** 5](#_Toc6419302)

[**3.5. Proiectare interfete** 6](#_Toc6419303)

[**3.6. Relatii** 6](#_Toc6419304)

[**3.7. Pachete** 6](#_Toc6419305)

[**3.8. Algoritmi** 6](#_Toc6419306)

[**3.8.0. Algoritmul de conexiune la baza de date - ConnectionFactory** 6](#_Toc6419307)

[**3.8.1. Algoritmii de validare – clasele din pachetul bll.validators** 6](#_Toc6419308)

[**3.8.2. Algoritmul de interogare a bazei de date** 7](#_Toc6419309)

[**4. Implementare** 7](#_Toc6419310)

[**4.1. Clasa Client** 7](#_Toc6419311)

[**4.2. Clasa Produs** 7](#_Toc6419312)

[**4.3. Clasa Comanda** 7](#_Toc6419313)

[**4.4. Clasa DetaliiComanda** 7](#_Toc6419314)

[**4.5. Clasa ConnectionFactory – clasa de conexiune cu baza de date** 7](#_Toc6419315)

[**4.6. Clasa AbstractDAO** 7](#_Toc6419316)

[**4.8. Clasa ClientDAO** 8](#_Toc6419317)

[**4.9. Clasa ProdusDAO** 9](#_Toc6419318)

[**4.10. Clasa ComandaDAO** 9](#_Toc6419319)

[**4.11. Clasa DetaliiComandaDAO** 9](#_Toc6419320)

[**4.12. Clasele ClientBLL, ComandaBLL, ProdusBLL, DetaliiComandaBLL** 9](#_Toc6419321)

[**4.13. Clasa Controller, View, ViewClient, ViewComanda, ViewJTable, ViewProdus** 9](#_Toc6419322)

[**4.14. Clasa Start** 10](#_Toc6419323)

[**5.Rezultate** 10](#_Toc6419324)

[**6.Concluzii si dezvoltari ulterioare** 10](#_Toc6419325)

[**7.Bibliografie** 10](#_Toc6419326)

# **1****.Obiectivul temei**

Propuneti, proiectati si implementati un sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi.

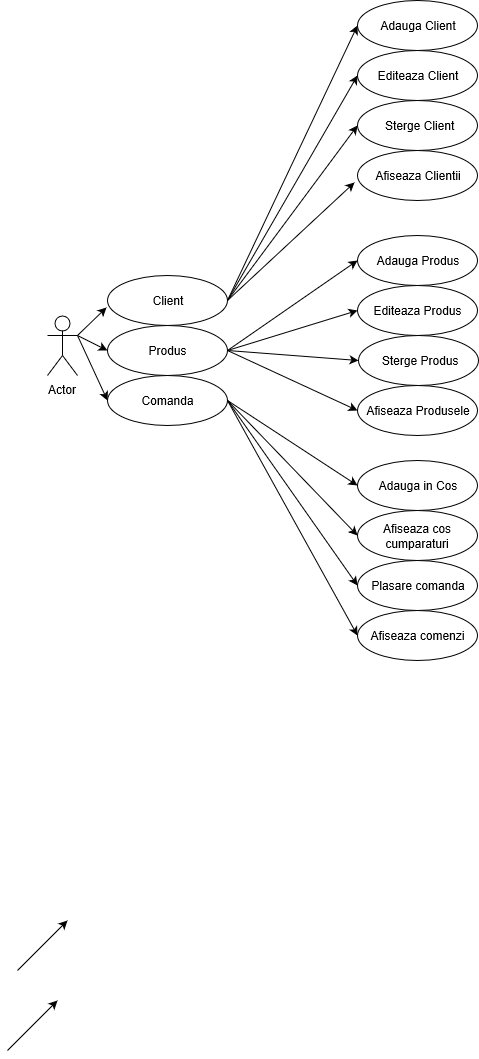
|  |  |  |
| --- | --- | --- |
| Obiectiv | Descriere | Capitol |
| dezvoltarea de use case-uri si scenarii |  | 2 |
| alegerea structurilor de date |  | 3 |
| impartirea pe clase |  | 3 |
| dezvoltarea algoritmilor |  | 3 |
| implementarea solutiei |  | 4 |
| testare |  | 5 |

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

Programul este conceput astfel incat sa se poata introduce, edita, sterge sau vizualiza, atat clienti si produse, cat si comenzile de produse pe care le fac acesti clienti. Pentru introducerea, editarea, stergerea sau vizualizarea de clienti, trebuie sa se aleaga optiunea clienti din fereastra ce apare la executia programului. La fel se procedeaza si in cazul in care dorim introducerea, editarea, stergerea sau vizualizarea unui produs sau a unei comenzi. Dupa aceasta actiune, se va deschide o alta fereastra corespunzatoare optiunii selectate. Aici, se introduc date, care sunt preluate in momentul in care se apasa pe butoanele corespunzatoare adaugarii, editarii, stergerii sau vizualizarii. Atentie! Pentru adaugare trebuie introduse toate datele, iar id-ul sa fie unic! Pentru editare, se introduc toate datele, si se editeaza doar clientul/produsul cu id-ul introdus. Pentru stergere, e suficient sa se introduca doar id-ul, iar pentru afisare, nu trebuie introdusa nicio data. La afisare se afiseaza un tabel cu toate datele introduse pana in momentul actual.In cazul comenzii, fereastra ce apare in momentul alegerii optiunii comanda, este una diferita de celelalte: Fereastra contine un id, un comboBox cu clienti, unul cu produse, si o cantitate. Se pot alege urmatoarele optiuni: Adauga in cos, Afisare cos de cumparaturi, plasare comanda, Afiseaza comenzi.

Diagrama use-case a acestui proiect este:

Daca datele introduse, sunt invalide, se afiseaza mesaje dupa cum urmeaza: „Id-ul trebuie sa fie unic!” daca id-ul se regaseste in tabela, iar daca acesta e negativ, se afiseaza „ID-ul trebuie sa fie pozitiv”. Daca nu sunt destule date introduse, se afiseaza un mesaj „Introdu datele”.Daca varsta nu este intre 18 si 100, se afiseaza „Limita de varsta nu e acceptata”. Daca adresa de email nu corespunde cu formatul unei adrese, se afiseaza „Email-ul nu e valid!”. Cand se adauga clientul, Se afiseaza un mesaj „S-a introdus clientul”. Daca lasam id-ul pe zero, id-ul clientului indrodus va fi egal cu id-ul ultimului client+1. La actualizare se afiseaza „S-a actualizat clientul”. La fel ca si in cazul produselor, se procedeaza la fel, doar ca, daca se adauga o cantitate negativa, se afiseaza „Conditia de cantitate nu e respectata”, iar daca se adauga un pret negativ, se afiseaza „Condifita de pret nu e respectata”.



# **3.Proiectare**

## **3.1. Decizii de proiectare**

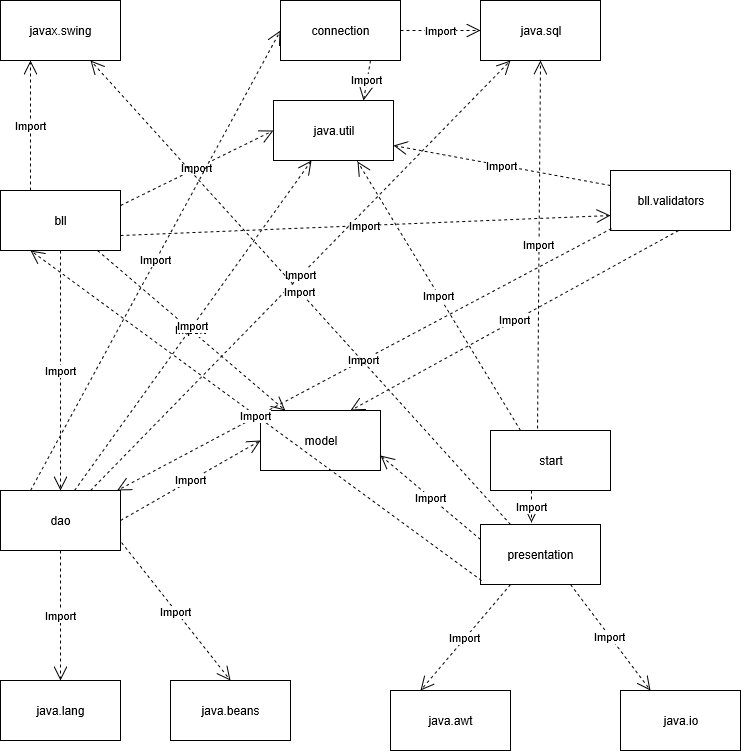
Am ales sa proiectez acest proiect in modul 3-tier architecture, cu layerele: Presentation Layer, Business Logic, Data Access, impartindu-l in 6 pachete: bll cu bll.validators, connection, dao, model, presentation, start. Am decis ca, clasele din pachetul model sa fie accesate de orice pachet, astfel rezultand o buna functionare a proiectului. Am ales ca pachetul dao sa contina clase ce acceseaza direct baza de date a proiectului, cu ajutorii conexiunii din conection, apoi logica sa se afle in pachetul bll, pachetul presentation afisand doar rezultatele. Pachetul start este utilizat pentru a porni aplicatia.

## **3.2. Diagrame UML**

Diagrama de clase a acestui proiect:

## 

Diagrama de pachete a acestui proiect:



## **3.3. Structuri de date**

Singura structura de date folosita in acest proiect este Lista, folosita in momentul cand, se stocheaza mai mult de un rand din tabela bazei de date. Cel mai des folosita este in momentul in care se afiseaza toate produsele/clientii/comezile.

## **3.4. Proiectare clase**

Se folosesc 4 clase pentru modelarea obiectelor, cum sunt: Clasa Client, Clasa Produs, Clasa Comanda in care se stocheaza cate un produs cumparat, si clasa DetaliiComanda care cuprinde toate produsele cumparate de cumparator. Se foloseste o clasa ConnectionFactory pentru a stabili o conexiune cu baza de date. Se folosesc clase pentru interogarea bazei de date, cum sunt: AbstractDAO ce contine metode de interogare a bazei de date, comuna tuturor tabelelor, clasa ClientDAO, ComandaDAO, ProdusDAO, DetaliiComandaDAO ce contin metode specifice claselor pentru a interoga tabelele bazei de date. Se folosesc clasele ClientBLL, ComandaBLL, DetaliiComandaBLL, ProdusBLL pentru a stabili logica programului, cat si validatori, ce au clasele specifice, ce valideaza datele introduse: CantitateValidator, EmailValidator, IdClientSiProdusValidator, IdValidator, PretValidator, Validator, VarstaValidator. Clasele specifice interfetei cu utilizatorul: Controller-ul ce da logica interfetei grafice, View ce se afiseaza prima oara – aici se alege pe ce tabel se fac operatii, ViewClient – vederea pentru client, ViewComanda- vederea pentru comanda, ViewProdus – vederea pentru produs si ViewJTable, vederea in care se afiseaza rezultatele, in tabele de tip JTable.

## **3.5. Proiectare interfete**

Singura interfata este Validator<T>, ce e o interfata generica, care are doar o singura metoda de validare: **public** **void** validate(T t);

## **3.6. Relatii**

## **3.7. Pachete**

Proiectul e structurat in 6 pachete: bll cu bll.validators, connection, dao, model, presentation, start. Pachetul start contine clase pentru pornirea proiectului. Pachetul model contine clasele a caror obiecte vor putea fi instantiate, corespunzatoare a cate unei linii din tabel. Aceste clase vor fi accesate din orice alte clase din alte pachete. Pachetul connection contine o clasa ce face conexiunea cu baza de date, astfel stocandu-se datele introduse de utilizator. Pachetul DAO contine clase ce ajuta la interogarea bazei de date, continand metodele accesatoare a acesteia. Pachetul BLL contine logica acestei aplicatii, facand legatura intre Presentation Layer si Data Access. Acest pachet contine un subpachet denumit BLL.validators, ce contine validatorii acestei aplicatii. Pachetul Presentation contine interfata grafica a acestei aplicatii, continand un controller si 5 Jframe-uri.

## **3.8. Algoritmi**

Algoritmii utilizati pentru acest proiect sunt algoritmii utilizati pentru interogarile bazei de date, continuti de clasa AbstractDAO, ClientDAO, ComandaDAO, DetaliiComandaDAO, ProdusDAO. Alti algoritmi utlizati sunt cei pentru conexiunea cu baza de date, cat si pentru validarea datelor introduse in baza de date.

## **3.8.0. Algoritmul de conexiune la baza de date - ConnectionFactory**

Pentru a crea o conexiune, se apeleaza functia DriverManager.getConnection(DBURL, USER, PASS); unde DBURL= "jdbc:mysql://localhost:3306/magazin" , magazin fiind numele bazei de date, USER fiind numele de utilizator cu care se logheaza, iar PASS, parola contului. Aceasta functie poate arunca o exceptie SQLException in momentul in care nu se poate crea conexiunea, aceasta exceptie trebuind sa fie prinsa, fiind afisat un mesaj de eroare: „Nu se poate face conexiunea”. In clasa de conexiune, ConnectionFactory, se adauga si metode de getConnection(), cat si metode ce opresc diferite componente, ca: conexiunea close( Connection), interogarea close( statement), cat si setul de rezultate: close( ResultSet)

## **3.8.1. Algoritmii de validare – clasele din pachetul bll.validators**

Pentru validare, se creeaza o clasa ce implementeaza interfata generica Validator<T>. In functie de apartenenta validatorului, se completeaza T cu clasa specifica. Daca insa validatorul este potrivit si folosit pentru mai multe clase model, nu se completeaza T. Fiecare clasa de validare implementeaza metoda **public** **void** validate(T t). Aici se verifica a carei instanta apartine t, clasa de validare este specifica mai multor clase model, apoi, pentru fiecare instanta a clasei, se valideaza. Daca in schimb, clasa de validare este specifica doar uneia, se valideaza in mod normal. Pentru calitate si pret, este suficienta conditia ca numarul corespunzator calitatii sau pretului sa fie pozitiva. Pentru email, se creeaza un pattern, folosind regex pattern, apoi acest email se compara cu patternul, trebuind sa se poriveasca. Validatorul pentru id client si pentru id produs, este specific clasei Comanda. Aici se cauta in tabelul Client si Produs al bazei de date, iar daca nu se gaseste id-ul introdus, se afiseaza un mesaj de eroare „Client ID necunoscut!”, „Produs ID necunoscut!” – validator pentru chei straine. Validatorul pentru id verifica ca id-ul sa fie pozitiv, iar in cazul in care aceste e negativ, se afiseaza „Id trebuie sa fie pozitiv!”cauta in baza de date id-ul introdus, iar daca se gaseste, afiseaza un mesaj „Id Client/Produs/Comanda trebuie sa fie unic” . Validatorul pentru varsta, verifica ca varsta sa se afle intre un MIN\_AGE si un MAX\_AGE. Am ales ca MIN\_AGE sa fie 18, iar MAX\_AGE sa fie 100. In cazul in care aceasta varsta nu se incadreaza in intervalul mentionat, se afiseaza un mesaj de eroare „Limita de varsta nu e acceptata!”.

## **3.8.2. Algoritmul de interogare a bazei de date**

Toti algoritmii care lucreaza pe baza de date se definesc in felul urmator: Se creeaza un String query care va contine intructiunea SQL ce trebuie executata. Se incearca stabilirea unei conexiuni cu baza de date, se pregateste sa se execute instructiunea, se executa aceasta intructiune, si se creeaza cate un obiect pentru setul de rezultate. La sfarsit, se opreste setul de rezultate, instructiunea, cat si conexiunea cu baza de date.

# **4. Implementare**

## **4.1. Clasa Client**

Atribute: id, nume, varsta, adresa, email – corespund cu campurile tabelei client din baza de date!

Constructori , Getter, Setter, metoda toString().

## **4.2. Clasa Produs**

Atribute: id, nume, cantitate, pret – corespund cu campurile tabelei produs din baza de date!

Constructori , Getter, Setter, metoda toString().

## **4.3. Clasa Comanda**

Atribute: id, idclient, idprodus, cantitate– corespund cu campurile tabelei client din baza de date!

Constructori, Getter, Setter, metoda toString().

## **4.4. Clasa DetaliiComanda**

Atribute: id, idprodus, pret– corespund cu campurile tabelei client din baza de date!

Constructori, Getter, Setter, metoda toString().

## **4.5. Clasa ConnectionFactory – clasa de conexiune cu baza de date**

Atribute: LOGGER, DRIVER, DBURL, USER, PASS.

Metode: Constructori, createConnection(), getConnection(), close( Connection), close( Statement), close( ResultSet).

## **4.6. Clasa AbstractDAO**

Atribute: LOGGER, type

Constructor ce gaseste tipul clasei si initializeaza atributul type.

Metode:

createSelectQuery(String field) pentru a crea o interogare de tipul „SELECT \* FROM tabel WHERE field = ?” unde tabel se inlocuieste de tipul clasei, adica de continutul lui type, iar ? de o variabila.

createSelectQueryWithoutWhere() – creeaza o interogare de tipul „SELECT \* FROM tabel”

createInsertQueryClient(), createInsertQueryProdus, createInsertQueryComanda, createInsertQueryDetaliiComanda() – creeaza o interogare de forma „INSERT INTO tabel VALUES(...)”, unde „...” se inlocuiesc cu cate valori contine un rand din tabel.

createDeleteQuery() – formeaza o interogare de forma „DELETE FROM tabel WHERE id = ?”, unde stergerea se face dupa cheia primara, care este id-ul in cazul nostru, pentru fiecare tabel.

findAll() – returneaza o lista de obiecte de tipul generic T. Aceste obiecte fiind returnate executand interogarea returnata de createSelectQueryWithoutWhere()

findById(int id) – gaseste obiectul ce are id-ul egal cu cel primit in functie (id). Se face executand interogarea returnata de metoda createSelectQuery(„id”)

findByNume(String nume) – gaseste obiectul ce are numele egal cu cel primit in functie. Se face executand interogarea returnata de metoda createSelectQuery(„nume”)

createObjects(ResultSet resultSet) – se creeaza o lista de obiecte ce corespund setului de rezultate resultSet

statementClient(Connection, PreparedStatement, T t), statementProdus(Connection, PreparedStatement, T t), statementComanda(Connection, PreparedStatement, T t), statementDetaliiComanda(Connection, PreparedStatement, T t) – se inlocuiesc toate „?” cu valorile lui t

insert(T t) – se introduc date in tabel, executand interogarea corespunzatoare celor 4 statement-uri de mai sus, si se returneaza ce s-a inserat.

Delete(int id) – se sterg datele din tabelul din baza de date, folosindu-se metoda care returneaza interogarea corespunzatoare: createDeleteQuery().

afiseazaColoane() – in aceasta metoda se face un select all, si se extrag numele campurilor corespunzatoare tabelelor din baza de date.

createTable(List<T> lista) – contine un vector de stringuri de coloane, care contine numele coloanelor returnate de functia de mai sus. Matricea de date contine datele ce se afiseaza in JTable. Astfel, primul rand din patrice este ocupat de numele coloanelor, iar restul matricii este compusa din valorile listei primite ca parametru.

## **4.8. Clasa ClientDAO**

Atribute: findVarstaStatementString, findAdresaStatementString, findEmailStatementString.

Metode:

createUpdateQuery(String field) – se creeaza interogarea pentru update

findByVarsta(int varstaMin, int varstaMax) – se gasesc clientii cu varsta ce apartine de intervalul [varstaMin, varstaMax]

findByAdresa(String adresa) – se gasesc clientii dupa adresa

findByEmail (String email) – se gasesc clientii dupa email

update(Client c) – se face update la clientul din baza de date.

## **4.9. Clasa ProdusDAO**

Atribute: findCantitateStatementString, findPretStatementString.

Metode:

createUpdateQuery(String field) – se creeaza interogarea pentru update

findByCantitate(int cantitateMin, int cantitateMax) – se gasesc produsele dupa cantitate

findByPret(int pretMin, int pretMax) – se gasesc produsele dupa pret

update(Produs c) – se face update la produs

updateStoc( int id, int cantitate) – se face update la stoc, iar daca nu mai exista produse, se arunca o exceptie cu mesajul „ Stoc epuizat! Sunt 0/.. produse!”

## **4.10. Clasa ComandaDAO**

Atribute: findCantitateStatementString, findPretStatementString.

Metode:

createUpdateQuery(String field) – se creeaza interogarea pentru update

findByCantitate(int cantitateMin, int cantitateMax) – se gasesc comenzile dupa cantitate

findByPret(int pretMin, int pretMax) – se gasesc produsele dupa comanda

update(Produs c) – se face update la comanda

## **4.11. Clasa DetaliiComandaDAO**

Atribute: findPretStatementString.

Metode:

findByPret(int pretMin, int pretMax) – se gasesc produsele dupa comanda

findByIdList(int id) – se afiseaza o lista de detaliicomanda ce au id-ul corespunzator.

## **4.12. Clasele ClientBLL, ComandaBLL, ProdusBLL, DetaliiComandaBLL**

Clase ce apeleaza metodele corespunzatoare din DAO , si folosesc validatorii pentru a valida datele.

## **4.13. Clasa Controller, View, ViewClient, ViewComanda, ViewJTable, ViewProdus**

Clase ce compun interfata grafica a aplicatiei, astfel: Controller-ul detine controlul interfetei grafice, adaugand ascultatori butoanelor din interfata grafica. Clasa View contine o fereastra cu 3 butoane in care se alege pe ce tabel dorim sa executam operatii de adaugare, editare, stergere sau afisare. Aceasta este prima fereastra ce se deschide la executia programului. Apoi, in functie de ce buton am apasat, se pot deschide ViewClient, ViewProdus, ViewComanda, in care se introduc date ce pot modifica tabela, adaugand date noi, editand datele, stergandu-le sau afisandu-le pe toate. La afisare, se deschide o alta fereastra, ViewJTabel ce contine tabelul Jtabel generat prin metoda findAll(). Diferenta dintre interfetele grafice se afla la ViewComanda, unde, se pot adauga produse in cos, apasand pe „adauga in cos”, unde se poate afisa ce s-a cumparat pana acum, adica continutul cosului: „Afisare cos de cumparaturi” – tot in fereastra ViewJTabel, se poate plasa comanda, ceea ce genereaza o factura pentru acest client, sau se pot afisa toate comenzile pana in momentul actual.

## **4.14. Clasa Start**

Metoda main() ce porneste aplicatia

# **5.Rezultate**

Aceasta aplicatie descrie o solutie optima de a simula cumparaturile clientilor, intr-un mediu online. Rezultatele se pot vizualiza atat in baza de date, cat si prin apasarea butonului de afisare. Un alt mod de a se vizualiza rezultatul, este prin factura printata pe care aplicatia o genereaza la apasarea butonului „Plasare Comanda”.

# **6.Concluzii si dezvoltari ulterioare**

In concluzie, consider ca acest program simuleaza un magazin online, intr-un mod usor simplist. Ca dezvoltari ulterioare, s-ar putea adauga o optiune de stergere a produselor din cos, cu incrementarea cantitatii. Aplicatia ar fi putut sa fie compusa din mai multe tabele, ca de exemplu tabelul furnizor.

# **7.Bibliografie**

* http://coned.utcluj.ro/~salomie/PT\_Lic/4\_Lab/HW3\_Tema3/Tema3\_HW3\_Indications.pdf
* <https://docs.oracle.com/javase/tutorial/uiswing/layout/box.html>
* [https://stackoverflow.com/questions/696782/retrieve-column-names-from-java-sql-resultset](https://stackoverflow.com/questions/696782/retrieve-column-names-from-java-sql-resultset?fbclid=IwAR3w-mcM5aay-iooDNaVVdua9MjZLYAsKkPaT3zFqps4Ee_RBT-VU_fhpR4)