

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

Tehnici de programare

Documentație

**ORDER**

**MANAGEMENT**

Bartalus Izabella

Grupa 30221

**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

Obiectivul temei cu numarul 3 este de a implementa o aplicatie Java care lucreaza cu o baza de date , adica datele vor fi stocate intr - o baza de date , la care vom avea o conexiune si vom efectua operatii cu aceste date , operatiile fiind metode Java .

Tema are la baza programarea pe obiectele si paradigmele acesteia . Programarea orientată pe obiecte ( POO , uneori și Programarea orientată obiect , uneori denumită ca și în limba engleză , Object Oriented Programming ( OOP ) ) este o [paradigmă de programare](https://ro.wikipedia.org/wiki/Paradigm%C4%83_de_programare" \o "Paradigmă de programare) , axată pe ideea încapsulării , adică grupării datelor și codului care operează asupra lor , într-o singură structură . Un alt concept important asociat programării orientate obiect este polimorfismul , care permite abstractizări ce permit o descriere conceptuală mai simplă a soluției . Principiile de baza ale acesteia , pe care le-am folosit in cod sunt : Abstractizarea  – Este posibilitatea ca un program să separe unele aspecte ale informației pe care o manipulează, adică posibilitatea de a se concentra asupra esențialului. Fiecare obiect în sistem are rolul unui “actor” abstract, care poate executa acțiuni, își poate modifica și comunica starea și poate comunica cu alte obiecte din sistem fără a dezvălui cum au fost implementate acele facilitați. Procesele, funcțiile sau metodele pot fi de asemenea abstracte, și în acest caz sunt necesare o varietate de tehnici pentru a extinde abstractizarea : Încapsularea – numită și ascunderea de informații: Asigură faptul că obiectele nu pot schimba starea internă a altor obiecte în mod direct (ci doar prin metode puse la dispoziție de obiectul respectiv); doar metodele proprii ale obiectului pot accesa starea acestuia. Fiecare tip de obiect expune o interfață pentru celelalte obiecte care specifică modul cum acele obiecte pot interacționa cu el . Polimorfismul – Este abilitatea de a procesa obiectele în mod diferit, în funcție de tipul sau de clasa lor. Mai exact, este abilitatea de a redefini metode pentru clasele derivate. De exemplu pentru o clasă Figura putem defini o metodă arie. Dacă Cerc, Dreptunghi, etc. vor extinde clasa Figura, acestea pot redefini metoda arie. Moștenirea – Organizează și facilitează polimorfismul și încapsularea, permițând definirea și crearea unor clase specializate plecând de la clase (generale) deja definite - acestea pot împărtăși (și extinde) comportamentul lor, fără a fi nevoie de a-l redefini. Aceasta se face de obicei prin gruparea obiectelor în clase și prin definirea de clase ca extinderi ale unor clase existente. Conceptul de moștenire permite construirea unor clase noi, care păstrează caracteristicile și comportarea, deci datele și funcțiile membru, de la una sau mai multe clase definite anterior, numite clase de bază, fiind posibilă redefinirea sau adăugarea unor date și funcții noi. Se utilizează ideea: ”Anumite obiecte sunt similare, dar în același timp diferite”. O clasă moștenitoare a uneia sau mai multor clase de bază se numește clasă derivată. Esența moștenirii constă în posibilitatea refolosirii lucrurilor care funcționează.

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 .

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

* Analiza problemei

Cerinta acestei aplicatii este de a implementa un depozit care sa contina trei tabele : Client , produs si Comanda intr - o baza de data relationala ( MySQL ) si sa se realizeze operatii de inserare , stergere si de update asupra clientilor , produselor si comenzilor . Pentru a implementa aceasta tema am implementat trei tabele in MySQL :

Diagram

Description automatically generated

* Modelare

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 Prenume, adresa, email”;
* pentru inserarea produsului – “Insert Product: Nume, cantitate, pret”;
* 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 afferent 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 .

1. Proiectarea ( decizii de proiectare , diagrama 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 sugestive care contin diferite clase :

* Pachetul ‘ bll ‘ care contine clasele : ClientBLL , ProductBLL , OrderBLL ;
* Pachetul ‘ connection ‘ care contine clasa : ConnectionFactoy ;
* Pachetul ‘ dao ‘ care contine clasele : AbstractDAO , ClientDAO , OrderDAO , ProductDAO ;
* Pachetul ‘ model ‘care contine clasele : Client , Order , Product ;
* Pachetul ‘ presentation ‘ care contine clasele : Bill , ClientView , OrderView , ProductView , View ;
* Pachetul ‘ start ‘ care contine clasa Start .

Diagrama UML a proiectului :

Diagram, engineering drawing

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 practice 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 .

1. Implementare

* Pachetul bll :
* Contine clasele ClientBLL, ProductBLL , OrderBLL.
* 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 ‘ dao ‘ :
* 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 ‘ ClientDA0 ‘ : 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 ‘ insertClient ‘ care insereaza un client in baza de date ;
* 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 ‘ findById ‘ care gaseste un produs dupa id – ul sau ;
* Metoda ‘ findByName ‘ care gaseste un produs dupa numele lui ;
* Metoda ‘ insertProduct ‘ care adauga un produs nou in baza de date ;
* 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 ‘ model ‘ :
* Clasa ‘ Client ‘ – contine atributele unui client ( id , nume , adresa , email ), constructori, getteri si setteri pentru atribute ;
* Clasa ‘ Order ‘ – contine atributele unei comenzi ( id , cantitate , numele clientului , numele produsului , pretul unui produs ) , constructori , getteri si setter pentru atribute ;
* Clasa ‘ Product ‘ – contine atributele unui produs ( id , nume , cantitate , pret ), constructori, getteri si setteri pentru atribute;
* Clasa ‘ Bill ‘ – contine o metoda pentru crearea unei facturi pentru o comanda ;
* Pachetul ‘presentation’:
* Clasa ‘View’ – genereaza frame – ul principal;
* Clasa ‘ClientView’ – genereaza frame – ul pentru client ;
* Clasa ‘OrderView’ – genereaza frame – ul pentru comenzi;
* Clasa ‘ProductView’ – genereaza frame – ul pentru produse;
* Pachetul ‘start’:
* Clasa ‘Start’ – main -ul.

1. Rezultate

Toate rezultate vor fi vizibile in interfata grafica unde exista metode pentru a lega toate butoanele de metodele care dorim sa se realizeze la apasarea unui buton . Informatiile cu care dorim sa facem operatii vor fi introduse de catre utilizator , cu ajutorul interfetei grafice , iar prin apasarea unui buton aceste operatii vor fi efectuate .

1. Concluzii

Sunt de parere ca aceasta tema este foarte utila pentru a mai invata/exersa lucrul cu bazele de date prin intermediul unui program Java.

1. Bibliografie
2. <https://mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/>
3. <https://www.baeldung.com/javadoc>
4. <http://tutorials.jenkov.com/java-reflection/index.html>
5. StackOverflow