Tema numarul 3: ORDERS MANAGEMENT

Documentatie

Student : Radu Vlad

Grupa : 30226

Materie : Tehnici de programare

Profesor indrumator : Mitrea Dan

Cuprins

1. Obiectivul temei .
2. Analiza problemei , modelare , scenario , cazuri de utilizare .
3. Implementare .
   1. Diagrame UML .
   2. Clase si metode .
   3. GUI .
4. Rezultate si Testare .
5. Concluzii .
6. Bibliografie . . . .
7. Obiectivul temei

Obiectivul temei este proiectarea si implementarea unei aplicatii care sa simuleze functionarea unui order management; adica avand un client si un tip de obiect, clientul sa poata realiza o comanada. Comanda presupune scrierea intr-un tabel a datelor prin care un client cu un id specific sa selecteze un produs avand deasemenea un id specific si catitatea dorita. Pe langa operatia de efectuare comanda aplicatia trebuie sa efectueze oferatile fundamentale pe o baza de date precum: inserea, stergerea, editarea si vizualizarea tuturor elemenetelor tabelelor.

Pentru realizarea operatilor de baza adica inserare, stergere, editare, si vizualizare am realizat alte doua operatii de cautare: cautarea tutuor inregistrarilor din tabel dar si cautarea dupa o variabila anume.

Pentru o utilizare cat mai interactiva am construit o interfata prin care utilizatorul prin intermediul apasarii unui buton speficica in ce tabela vrea sa faca operatii, astefel este deschisa o noua fereastra care contine tabelul cu inregistrari si atat texfidurile campurilor necesare cat si butoanele operatilor ce se pot efectua.

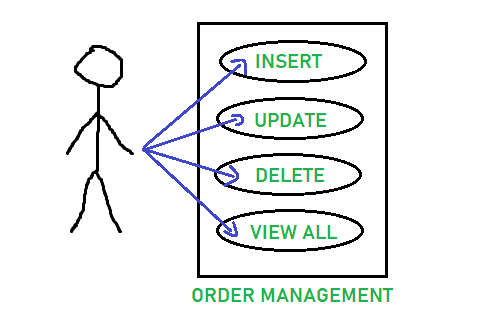
1. Analiza Problemei

Proiectul de fata scoate in evidenta utilizarea bazelor de date si imbinarea acestora cu un cod scris in java, precum si utilizarea printr-o interfata grafica cu utilizatorul (gui). Pentru realizarea bazei de date am utilizat aplicatia MySQL WorkBench cu care am creeat un cont (avand user si parola) si o noua baza de date cu 3 tabele.

MySQL este un sistem de gestiune a bazelor de date relaționale, produs de compania suedeza MySQL AB și distribuit sub Licența Publică Generală GNU. Este cel mai popular SGBD open-source la ora actuala.

Pentru conectarea la baza de date avem nevoie de un user, de o parola, de un dburl care sa relizeze conexiunea si de un driver. La instalarea aplicatiei MySQL am create un cont prin care se face conectarea la reteaua bazei de date. Aceste date vor fi trimise ulterior si in codul java. Inregistrarile din baza de date se disting prin prezenta primary keyurilor in fiecare tabel prima variabila numita id este setata ca primary-Key ea fiind unica in tabel si avand o valoare neneula.

Un mod usor de vizualizare a functilor pe care le poate realiza aplicatia este prin intermediul unei diagrame use

Astfel functile sunt: Insert: adauga un nou rand cu specificatile obiectului in tabelul din baza de date, Update: realizea modificari ale obiectului, Delete: sterge randul dorit din tabel, View All: afiseaza intreg continutul tabelui.

1. Implementare
2. Diagrama UML

Unified Modeling Language (UML) este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost dezvoltatla început pentru reprezentarea complexității programelor care erau contruite utilizând programarea orientată pe obiect, al căror fundament este structurarea programelor pe clase și instanțele acestora (obiecte). Diagrama clasei este elementul principal al modelarii orientate pe obiecte.

Am utilizat UML-ul pentru a vedea mult mai clar împărțirea pe clase a codului, precum și variabilele instanță și metodele pe care le-am utilizat în dezvoltarea order managementului.

Graphical user interface

Description automatically generated

1. Clase si Metode

La baza proiectul este construit dupa modelul MVC (model, view, controller) adică tot codul ar putea fi împărțit în 3 mari pachete: model, view și controller.

Clasa MainClass este clasa principala, de aici incepe executia programului acesta face legatura intre view si controller, are două instanțe; una a clasei View, pe care o setăm vizibilă, pentru a putea vedea interfața grafică (GUI) și cealaltă este o instanță a clasei Controller, care leagă Model-ul de View, fiind principalul punct de comandă al celor două, adica leaga partea de interfata grafica cu utilizatorul de functionalitatile aplicatei.

Text

Description automatically generated

Tot in pachetul main este clasa Refection care implementeaza metoda retriveProperties(Object object), metoda parcurge fiecare field, ce reprezinta numele coloanelor din tabel si citeste cu ajutorul metodei get(), in variabila value valoare corespuzatoare. Metoda returneaza lista de headere si valoarea acestora pentru obiectul object.

Pachetul bll contine la randul lui un pachet numit validators si 3 clase corespunzatoare celor 3 tabele numele lor fiind terminate cu sufixul BLL. Aceste 3 clase definesc apelurile de metode din clasa AbstractDAO pe baza reflexiei.

Text

Description automatically generated

Pachetul validators continue 2 metode prin care este testat daca datele primite de la tastatura corespund formatului impus, inainte de a fi introduse in baza de date.

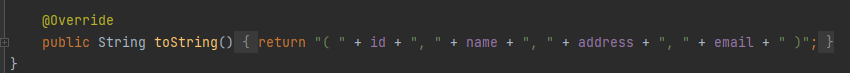
Text

Description automatically generated

Clasa ClientBLL precum si ProductBLL sau OrderBLL contin o lista prin care se valideaza datele obiectelor si un obiect al clasei ClientDAO In constructor este intializat acest obiect prin intermediul caruia, pe baza refexiei sunt apelate metodele insert(), update(), delete(), find(). Acest lucru se petrece în niște metode sugestive, precum insertClient(), updateClient(), deleteClient(), findAllClients() și findCleintByID().

Pachetul model conține clasele Client, Order și Product, care sunt proiecția în Java ca structură a celor 3 tabele din baza de date. Variabilele instanță ale clasei Client sunt: id, name, adresa si email. Cele ale clasei Product sunt: id, name, stoc si price, în timp ce pentru clasa Order avem: id, idClient, idProduct și quantity.

În fiecare clasă avem 3 constructori, unul declarat implicit, unul care conține toți parametrii și altul care nu conține parametrul corespunzător primary key-ul din tabelul bazei de date. Ultimii 2 constructori conțin și un apel explicit al constructorului unei clase părinte, prin metoda super().

Fiecare clasă conține gettere și settere pentru fiecare variabilă declarată private, pentru a obține/modifica datele. Pe lângă acestea, mai avem și o metodă toString(), folosită pentru a afișa un obiect într-un format mult mai lizibil. 

Pachetul Connection marcheaza legatura dintre codul scris in java si baza noastra de date. Pentru conectarea la baza de date se foloseste un obiect de tipul singleton. Singletonii pot fi folosisiti in timp ce lucrati cu baze de date. Ei pot fi utilizati pentru a creea un pool de conexiuni pentru a accesa baza de date in timp ce refolosesc aceeasi conexiune pentru toti clientii. Clientii pot accesa instanta clasei Singleton doar prin intermediul operatiei Instance. Astfel date vor fi modificate correct in urma operatiolor efectuate.

Pachetul dao conține clasele AbstractDAO, ClientDAO, OrderDAO, ProductDAO. Ultimele 3 sunt doar clase prin care apelăm metodele generale din AbstractDAO, singurul merit al lor fiind acela că mostenesc clasa AbstractDAO.

Clasa AbstractDAO contine toate metodele prin care sunt implementate functile cu care se lucreaza in baza de date. Formatul fiecarei interogari se creeaza in metode separate care mai apoi sunt introduse in metodele ce executa operatile de inserare, editare, stergere si afisare. Text

Description automatically generated

Text

Description automatically generatedMetoda createSelectQuery(String field) returneaza queryul format pe baza fieldului primit ca parametru.

Type.getSimpleName() extrage numele tabelului pe care lucram iar field reprezinta coloana in functie de care avem conditionarea interogarii.

Text

Description automatically generatedMetoda createSelectAllQuery() functioneaza pe acelasi principiu cu metoda createSelectQuery(String field) insa aceata nu are o variabila de conditionare, astfel ca vor fi afisate toate datele din tabelul respectiv.

Text

Description automatically generatedMetoada createInsertQuery(Object object) genereaza queryul de insert specific limbajului sql. Sunt parcurse toate campurile si sunt introduse semne de intrebare acolo unde ar trebui sa avem valorile pe care dorim sa le inseram.

Aceste valori corespund fiecarei coloane in parte.

Aceeasi logica se aplica si in cazul metodei createUpdateQuery(Object object) precum la metoada createInsertQuery(Object object) doar ca in acest caz structura interogarii este cea pentru update-query.

Text

Description automatically generatedIn cazul metodei createDeleteQuery() se aplica aceeasi pasi in creearea interogarii.

Text

Description automatically generated

Metoada findAll() returneaza o lista de obiecte creata prin metoda createObjects(). Majoritate codului este strict pentru conectarea si formarea legaturii cu baza de date, spre care este transmis ca parametru queryul format de metoda createSelectAllQuery();

Text

Description automatically generated

Logica constructiei metodei findById(int id) este identica cu cea a metodei findAll() doar ca in acest caz este introdus ca parametru id ul obiectului iar in loc de o lista metoda returneaza obiectul gasit sau null in caz contrar.

Text

Description automatically generated

Metoda insert(T t) primeste ca parametru un obiect abstract de tipul T. Sunt parcurse toate campurile din tabelul respectiv si se incearca introducerea datelor in baza de date in functie de tipul fiecarui camp.

Text

Description automatically generated

Metoda update(T t) este scrisa analog, la final pentru marcarea noilor date se foloseste functia predefinita executeUpdate();

Analog si in cazul functiei de delete sunt sterse toate campurile dorite in functie de id.

1. GUI

In pachetul de gui marcat de presentation am scris clasele Controller, InputWindow, ClientWindow, ProductWindow, OrderWindow.

GUI este reprezentată de Graphical User Interface, un termen folosit nu numai în [Java,](https://ro.eferrit.com/ce-este-java/) ci și în toate limbajele de programare care suportă dezvoltarea de GUI-uri. Interfața grafică de utilizator a unui program prezintă un afișaj vizual ușor de utilizat pentru utilizator. Acesta este alcătuit din componente grafice (de exemplu butoane, etichete, ferestre) prin care utilizatorul poate interacționa cu pagina sau [aplicația](https://ro.eferrit.com/exemplu-cod-java-pentru-a-construi-o-aplicatie-simpla-gui/) .

Un GUI include o serie de elemente de interfață cu utilizatorul, comenzile de intrare, cum ar fi butoanele, listele derulante, casetele de selectare și câmpurile de text.

Interfața grafică cu utilizatorul este creată prin clasa InputWindow și legată de restul codului prin Controller.

Clasa Controller contine un constructor care initializeaza view ul principal. Tot in clasa Controller este implementata metoda de tip ActionPerformed prin care este setata vizibilitatea ferestrei pe care vrem sa muncim la momentul apasarii butonului.

Text

Description automatically generated

Prin intermediul interfetei grafice utilizatorul este capabil sa interactioneze cu aplicatia si sa vada modificarile efectuate intr-un mod mult mai comfortabil. Totodata interfata grafica ofera posibilitatea de afisare dinamica a datelor de iesire

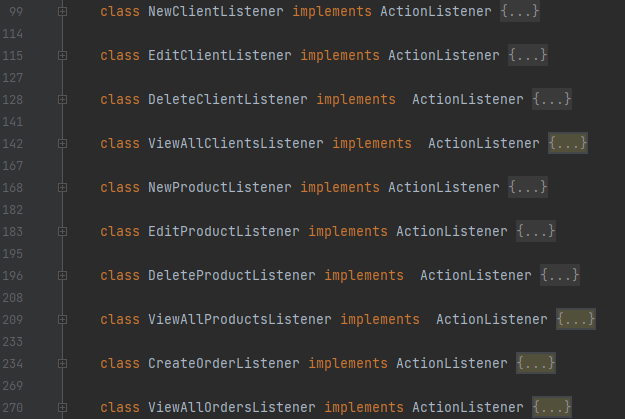
Graphical user interface, application

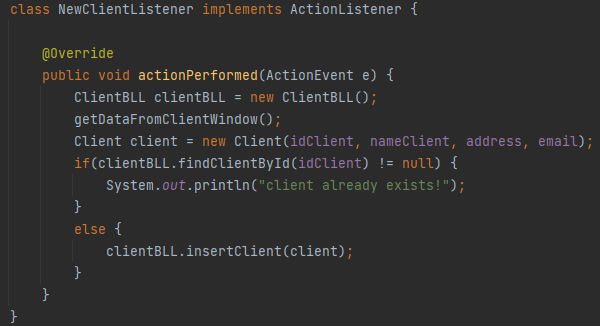
Description automatically generated

La apasarea unui buton din fereastra principala se deschide o noua fereastra specifica operatiolor pe care dorim sa le efectuam. Clasele ClientWindow, ProductWindow contin butoane (cate unul pentru fiecare operatie : insert, update, delete, view) si textFielduri pentru introducrea datelor. Totodata in partea dreapta, am aduagat un Jpanel care sa ocupe loc pentru un tabel de tipul Jframe. Pentru butoane am creat metode cu parametrul un ActionListener, iar pentru textField-uri am creat getter-e și setter-e.

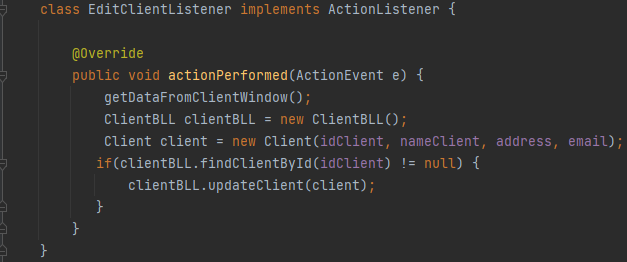
Toate cele 3 tabele sunt populate prin umplerea tabelului cu o matrice de date si un sir cu numele coloanelor.

Tot in clasa Controller sunt implementate alte clase cu rol de a efectua comenzile specifice bazelor de date.

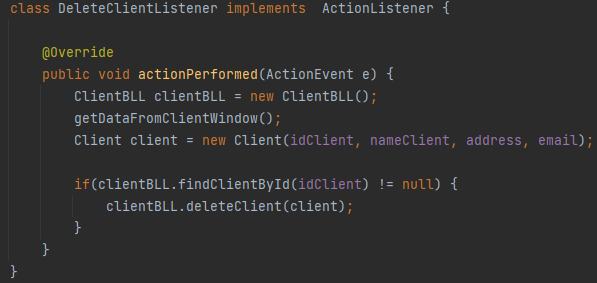




In momentul apasarii butonului, in cazul in care idul nu este folosit de un alt client se insereaza in tabel o noua valoare.



Butonul de edit realizeaza modificare datelor clientului prin invocarea metodei update.

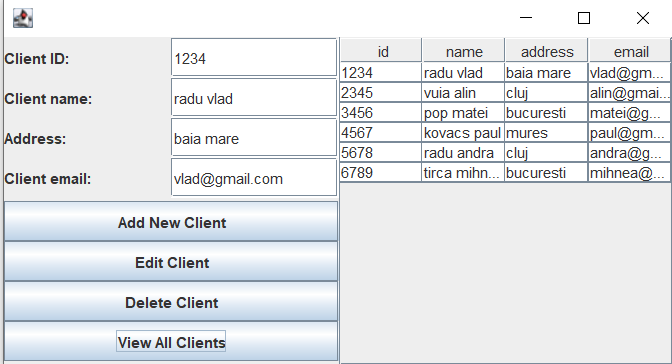


Functia de delete cauta un client dupa id iar in cazul in care clientul se gaseste acesta urmeaza sa fie sters din tabel.

Pentru celelalte tabele metodele se apeleaza in mod analog.

1. Testare

Partea de testare a proiectului s-a realizat prin intermediul GUI. Cu ajutorul lui am putut vizualiza dacă au avut loc inserările, modificările și ștergerile.

Am ales sa exemplific modul in care functioneaza functia de editare. Astfel: avand un client inregistrat in tabel ii scriem date in casutele respective astfel incat valoare ID sa corespunda cu cea din inregistrare, iar la momentul apasarii butonului Edit Client datele ce nu corespund vor fi modificate. In cazul meu am schimbat adresa din tabel cu cea trimisa ca parametru. La Apasarea butonului View All Clients valorile din tabel se reimprospateaza fiind vizibila schimbarea.

1. Concluzii

In urma implementarii acestui proiect am realizat imporatnta functionalitatilor pe care acesta le are in domeniul comertului in magazine si supermaketuri cat si in domeniul industriei fiind capabil sa fie integrat in procedee mult mai complexe. Un astfel de dispozitiv ușurează foarte mult viața oamenilor care utilizează magazinele online.

Pe langa asta, personal, proiectul a reprezentat o sumedenie de avantaje in dezvoltarea stilului meu de programeare. M a ajutat sa imi amintesc anumite concepte de programare in java si respectarea anumitor paradigme de programre. Dar si sa invat noi tehnici si tipuri de clase.

Cu siguranta proiectului ii se pot aduce imbunatatiri pe partea de interfata cu utilizatorul. Un mod mult mai practic si intuitiv de folosire ar fi daca date ar putea fii modificate din interiorul tabelului dand click pe casuta unde dorim sa facem modificarea.

1. BIBLIOGRAFIE:

<https://stackoverflow.com/>

<https://www.geeksforgeeks.org/>

<https://dsrl.eu/courses/pt/materials/A3_Support_Presentation.pdf>