DOCUMENTATIE

TEMA 3

ORDER MANAGEMENT

Varvari Alexandru

Grupa 30225

CUPRINS

1. Obiectivul temei

-Se va prezenta obiectivul principal al temei printr-o fraza si un tabel sau o lista cu obiectivele secundare. Obiectivele secundare reprezinta pasii care trebuie urmati

pentru indeplinirea obiectivului principal. Fiecare obiectiv secundar va fi descris si

se va indica in care capitol al documentatiei va fi detaliat.

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

-Modelare, scenarii, cazuri de utilizare . Se va prezenta cadrul de cerinte functionale formalizat si cazurile de utilizare ca si diagrame si descrieri de use-case.

Descrierile use-case-urilor se vor face sub forma unui flow-chart ori sub forma unei liste continand pasii executiei fiecarui use-case.

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

-Se va prezenta proiectarea OOP a aplicatiei, diagramele UML de clase si de pachete, structurile de date folosite, interfetele definite si algoritmii folositi (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator).

4. Implementare

-Se va descrie fiecare clasa cu campurile si cu metodele importante. Se va descrie implemantarea interfetei utilizator.

5. Rezultate

-Se vor prezenta scenariile pentru testare cu Junit sau alt framework de testare.

6. Concluzii

-Se vor prezenta concluziile, ce s-a invatat din tema, dezvoltari ulterioare.

7. Bibliografie

-Se vor mentiona resursele bibliografice care au fost folosite pentru dezvoltarea temei.

1. Obiectivul temei

Obiectivul acestei teme este implementarea unei aplicatii OrderManagement care proceseaza comenzile clientilor pentru un depozit. Se folosesc baze de date relationale pentru a stoca produsele, cleintii si comenzile. De asemenea aplicatia trebuie sa fie structurata pe pachete folosind o arhitectura stratificata. Si ar trebui sa urmeze minimal urmatoarele clase:

* Clasele model – modelele de date ale aplicatiei ;
* Clasele pentru logica de afaceri – implementeaza logica aplicatiei ;
* Clasele de prezentare – implementeaza input-ul / output-ul user-ului ;
* Clasele pentru acces de date – implementeaza accesul la baza de date ;

Obiective secundare*:*

* Dezvoltarea de use-case-uri
* Alegerea corecta a structurilor de date
* Impartirea pe clase
* Dezvoltarea algoritmilor
* Implementarea solutiilor
* Testarea programului

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

Aplicatia trebuie sa permita procesarea comenzilor dintr-un fisier text dat ca argument in linia de comanda, sa efectueze operatiile cerute, sa salveze modificarile efectuate in baza de date si sa genereze rapoarte in format pdf. Alte clase si pachete pot fi adaugate pentru a implementa functionalitatea completa a aplicatiei.

De asemenea , trebuie implementat si un parser pentru citirea comenzilor din fisier in layer-ul de prezentare ( in loc de interfata grafica standard ) si un generator de fisiere pdf care sa genereze rapoartele.

Operatiile pe care aplicatia le poate procesa:

* Adaugarea unui client in baza de date ;
* Stergerea unui client din baza de date ;
* Adaugarea unui produs in baza de date ;
* Stergerea unui produs din baza de date ;
* Crearea unei comenzi pentru client ;

1. Proiectare

* Decizii de proiectare:

Se va folosi tehnica “ reflection “ atat pentru accesul si operatiile cu baza de date cat si pentru scrierea in fisiere. **Java Reflection** este procesul de analizare si modificare a tuturor capabilitatilor unei clase la runtime.

* Structuri de date folosite

Am folosit o structura de tip List < T > , generica, pentru a retine rezultatele operatiilor de select in clasa DAO indiferent ce tip ar avea acestea.

Diagrama Use-Case:

A close up of text on a white background

Description automatically generated

Diagrama UML de pachete:

A picture containing clock, bird

Description automatically generated

Diagrama UML de clase:

In continuare voi prezenta clasele si pachetele pe care le-am folosit in acest proiect, dar si metodele cele mai importante din fiecare clasa.

 ConnectionFactory

ConnectionFactory este o clasă care stabilește legătura cu baza de date, adoptând pattern-ul Singleton. Pattern-ul Singleton este o tehnică de design software ce permite crearea unui singur obiect al unei clase, fiind util când se impune coordonarea acțiunilor în cadrul sistemului printr-un singur obiect. Clasa ConnectionFactory este încadrată în pachetul DataAccess, furnizând conexiunea cu baza de date. Clasa conține driver-ul MySQL pentru baza de date, userul, parola și locația bazei de date. Un atribut major al clasei este singleInstance, care, împreună cu constructorul, asigură accesul la propriul obiect al clasei fără a necesita instantierea.

 Model

Pachetul Model găzduiește clasele ce reprezintă tabelele din baza de date, mapează toate atributele pe coloanele respectivelor tabele și conține constructori, seteri și getteri.

Clasa Client include următoarele atribute: id-ul clientului (incrementat automat la introducerea acestuia în baza de date și folosit ca cheie primară în tabelul Clients), numele clientului, adresa sa, adresa de e-mail și vârsta clientului.

Clasa Order constă din următoarele atribute: id-ul fiecărei comenzi, id-ul clientului care a plasat comanda, id-ul produsului dorit și cantitatea dorită (un număr întreg).

Clasa Product conține următoarele atribute: id-ul produsului (incrementat automat la introducerea acestuia în baza de date și folosit ca cheie primară în tabelul Products), numele produsului, cantitatea disponibilă în stoc și prețul pe unitate al produsului (un număr real).

 BusinessLogic

Pachetul BusinessLogic conține logica aplicației, deci clasele care implementează toate metodele și operațiile pe baza de date. Clasa DAO utilizează tehnica reflection și are ca atribute un logger, un obiect de tip ReportGenerator pentru a permite apelarea funcțiilor de scriere în PDF și un tip, generic. În constructor, se preia clasa tipului generic T cu care se creează obiectul de tip DAO.

Pentru fiecare operație necesară asupra bazei de date în urma primirii unei comenzi din fisierul de intrare txt, clasa conține o metodă privată pentru a returna sub forma de string query-ul SQL corespunzător. Se folosește un obiect de tip StringBuilder și se apelează metoda append() pentru a concatena string-ului datele necesare. De exemplu, metoda createSelectQuery() returnează un obiect de tip string de forma: "select \* from" + "where". Numele tabelului este găsit prin intermediul parametrului type și a apelării metodei getSimpleName(), iar condiția pentru select este dată de cei doi parametri ai funcției, field-ul care ne inter

1. Rezultate

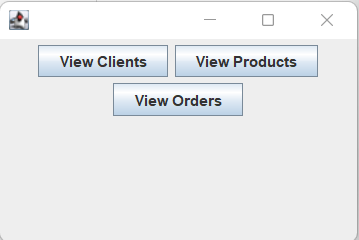
Se foloseste o baza de date initial goala. In momentul in care se citesc pe rand comenzi din fisier, aceasta sufera modificari datorita metodelor apelate.

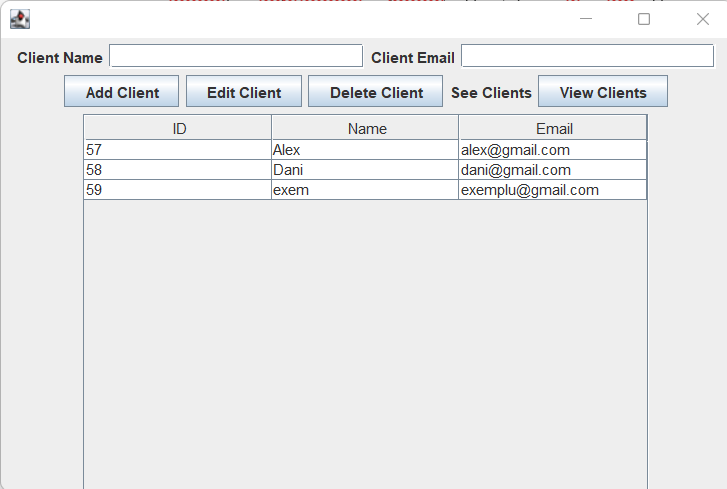
**Insert client** contine numele si adresa clientului care se doreste a fi inserat in baza de date.

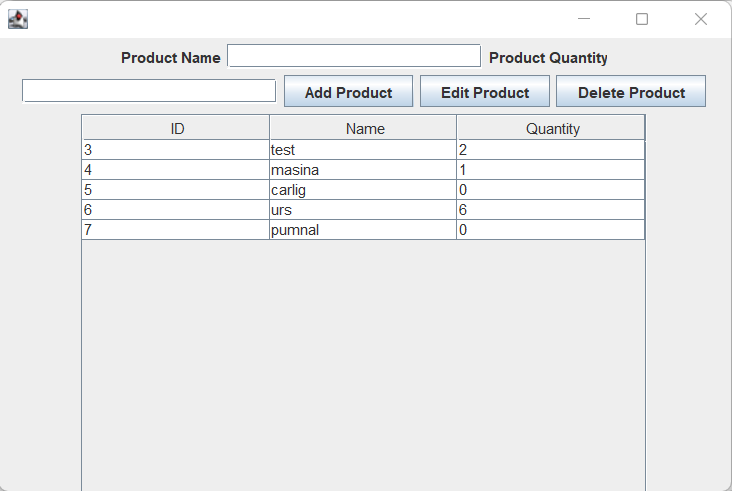
* **Delete client** contine numele clientului care se doreste a fi sters din baza de date a depozitului. In cazul in care clientul are comenzi existente, se sterg si acestea.:
* **Insert product** – se va insera in tabela de produse un produs nou caracterizat prin nume, cantitatea din stoc si pret. In cazul in care se va incerca inserarea unui produs care deja exista, cantitatea acestuia va fi acutalizata prin adunare. Se presupune ca un produs care are acelasi nume cu unul deja existent, are si aceslasi pret. Dupa inserarea a 3 produse, “apple”, “peach” si din nou “apple”

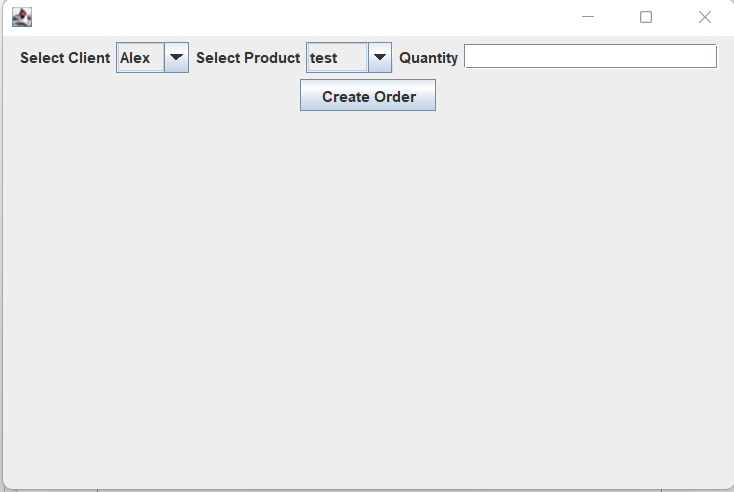
Se observa ca produsul “apple” are ca si cantitate suma celor doua cantitati introduse la momente diferite de timp.

* **Delete product** – sterge din baza de date produsul cu id ul primit.









1. Concluzii

O posibila imbunatatire poate fi adusa la capitolul eficientei. Ca si dezvoltare ulterioara as dori sa introduc o factura in care s-ar pune toate produsele comandate de un client, nu unul singur, cu un total corespunzator.

1. Bibliografie

https://www.javatpoint.com/example-to-connect-to-the-mysql-database

https://www.mysql.com

https://howtodoinjava.com/library/read-generate-pdf-java-itext/

* 1. https://en.wikipedia.org/wiki/Parsing