SISTEM DE PROCESARE A COMENZILOR

TEHNICI DE PROGRAMARE

UNIVERSITATEA TEHNICA CLUJ-NAPOCA

FACULTATEA DE AUTOMATICA SI CALCULATOARE

SPECIALIZAREA CALCULATOARE SI TEHNOLOGIA INFORMATIEI

Student Urcan Denisa-Teodora

Grupa 30226 CTI-Romana seria B

Cuprins

1. Obiective

1.1. Obiectiv principal

1.2. Obiectiv secundar

2.Analiza problemei

2.1. Analiza problemei

2.2. Modelarea

2.3. Scenarii

2.4. Cazuri de utilizare

3. Proiectare

3.1. Decizii de proiectare

3.2. Diagrame UML

3.3. Structuri de date

3.4. Proiectare clase

3.5. Algoritmi

3.6. Interfata utilizator

4. Implementare

5. Rezultate

6. Concluzii

7. Bibliografie

1. Obiective
   1. Obiectivul principal

Subiectul temei consta in propunerea, proiectarea si implementarea unui sistem de procesare a comenzilor in vederea determinarii produselor alese de un client. Cerintele aferente acestei teme sunt respectarea paradigmelor programarii orientate pe obiecte, structurate in pachete, clase cu numar maxim 300 linii de cod si metode cu numar maxim 30 de linii. Implementarea temei respecta conventiile de numire Java, si implicit stilul de scriere camel case.

De asemenea, se respecta proiectarea sugerata din punct de vedere al arhitecturii, si anume Layered Architecture, aplicatia continand mai multe pachete, printre care cele mai relevante fiind dataAccess, businessLayer, model, presentation. Se utilizazea tehnica reflexiei pentru a prelua anumite proprietati ale obiectelor, precum si tehnici abstracte pentru a evita duplicarea codului.

Pentru a permite unui client sa efectueze operatii asupra mai multor produse, se utlizeaza patru tabele, cel de-al patrulea furnizand detalii asupra unei comenzi( tehnica detaliata ulterior).

Baza de date reprezintă o modalitate de stocare a unor informații și date cu posibilitatea extinderii ușoare și a regăsirii rapide a datelor. Bazele de date relaționale sunt un tip de baze de date în care datele, văzute ca și atribute, sunt stocate în tabele și sunt legate între ele prin relații.

Arhitercura Layered presupune proiectarea aplicației folosind cel puțin patru pachete de clase, și anume: **dataAccessLayer,** pachet care conține legătura cu baza de date - MySql în acest caz-, **businessLayer** este pachetul care conține clasele în care se regăsește implementarea metodelor necesare pentru efectuarea operațiilor, pachetul **model** conține clasele care mapează tabelele din baza de date, iar pachetul **presentationLayer** conține clasele care definesc interfața grafică.

Acest proiect simulează operațiile de adăugare, editare, ștergere și afișare pentru clienți, produse și comenzi. Aplicația este destinată firmelor de livrări sau depozitelor, cu scopul de a ține o evidență a produselor și clienților și, de asemenea, a comenzilor efectuate.

1.2 Obiective secundare

|  |  |  |
| --- | --- | --- |
| Obiectiv secundar | Descriere | Capitol |
| Folosirea structurilor de date convenabile | Structurile de date sunt alese astfel incat sa faciliteze adaugarea si extragerea clientilor precum si mentinerea unui tip de data sortat. | 3 |
| Impartirea pe pachete | Fiecare pachet este format din mai multe clase care indeplinesc sarcini inrudite | 3 |
| Impartirea pe clase | Fiecare clasa se ocupa de manipularea unui singur obiect, respectiv a unei singure operatii. | 3 |
| Algoritmi utilizati | Algoritmii utilizati pentru a realiza diferitele operatii sunt cei matematici. | 3 |

1. Analiza problemei
   1. Analiza problemei

Analiza problemei presupune identificarea claselor proiectului, functionalitatile pe care acesta le implementeaza, legaturile dintre ele si modelul impus din punct de vedere al structurii. Programarea orientata pe obiecte ofera strategia de dezvoltare „Top-Down”, prin care utilizatorului ii sunt dezvaluite doar informatiile de suprafata. Aplicatia, care are ca si scop principal efectuarea operatiilor asupra clientilor si produselor este structurata in clase, fiecare detinand o functionalitate, implementate ulterior.

Datorită strategiei oferite de programarea orientate pe obiecte, se pot identifica clasele, denumite prin substantive, și funcționalitățile, denumite cu ajutorul verbelor.

Utilizarea bazei de date MySQL ne permite stocarea informațiilor sub forma de tabele.

Operatia de adaugare a unui client respectiv a unui produs presupune introducerea datelor despre obiectele respective, respectiv nume, adresa, adresa de email, numar de telefon pentru clienti, nume, cantitate, pret pentru produse. Tabelele care contin informatii legate de comenzi si detaliile acestora sunt formate automat, clientul alegand anumite produse. Operatia de editare presupune introducerea id-ului clientului/ produsului de editat, si completarea a cel putin unui camp legat de atributele celor doua obiecte. Operatia de stergere se realizeaza pe baza id-ului.

2.2 Modelarea

Din punct de vedere al modelarii aplicatiei, s-au implementat diverse clase, fiecare operand asupra unui singur obiect. Clasa Client contine datele specifice fiecarui client : id, nume, adresa, adresa de email, numar de telefon. Clasa Product contine atribute definitorii produselor, precum id, nume, cantitate si pret. Clasele View si Controller sunt responsabile de crearea interfetei grafice si comunicarea cu utilizatorul. Clasa Order detine informatii despre numele clientului si un id, acestea realizand legatura cu detaliile comenzii (clasa orderDetail), care contin id-ul comenzii si al produsului, acestea din urma reprezentand cheia primara a tabelului aferent, precum si cantitate. Al doilea pachet, **dataAccess,** reprezintă legătura cu baza de date MySql, în care avem reținute tabelele sub aceeași formă în care au fost definite în clasele pachetului model: Customer, Order și Product. Acesta este compus din cinci clase, și anume: **CustomerAccess, ProductAccess, OrderAccess**, ConnectionFactory și Reflection, reprezentând legătura bazei de date cu tabela clienților, Customer, legătura cu tabela produselor, Product, legătura cu tabela comenzilor, order, definirea conexiunii cu baza de date MySql și, respectiv, implementarea tehnicii reflexiei.

Următorul pachet, **bussinesLogic,** este alcătuit din trei clase: **CustomerLogic, ProductLogic** și **OrderLogic.** Aceste clase implementează operațiile care se efectuează asupra bazei de date apelând metodele din clasele corespunzătoare pachetului dataAccess.

Ultimul pachet**, presentation**, conține 25 clase specifice pentru interfața cu utilizatorul, astfel: clasa View, reprezentând fereastra principală, unde avem butoane pentru a deschide ferestrele destinate clienților, produselor și comenzilor.

2.3 Scenarii

Utilizarea corecta a aplicatiei presupune parcurgerea drumului pentru a intra in magazin(->). Inauntrul magazinului, se regasesc diverse coduri pentru diverse operatii. Astfel, la apasarea unui click asupra unui produs se deschide fereastra aferenta produselor, la apasarea unui click asupra clientilor se deschide fereastra aferenta clientilor, la apasarea unui click asupra cosului de cumparaturi se deschide fereastra care permite selectarea unui client si mai multor produse. Ultima operatie, la apasarea itemului care si parcurge drumul pana la magazin, se realizeaza factura.

Cazul ideal de utilizare presupune un utilizator care doreste sa simuleze procesarea unor cozi dupa cum urmeaza:

* Utilizatorul parcurge drumul spre usa magazinului, folosind sageata ->
* Utilizatorul alege una dintre operatiile descrise mai sus, prin apasarea unui click in diverse zone ale magazinului
* Programul va simula introducerea, editarea, stergerea clientilor/ produselor/ comenzilor
* La finalul simularii vor fi generate date statistice despre comenzi: factura( prin apasarea itemului care a intrat in magazin)
* Daca se apasa butonul de back, aplicatia va fi resetata si pot fi introduse si simulate date noi de simulare.

--------------------------🡪

view

Statistics

Back

view

2.4 Cazuri de utilizare

1. Utilizatorul apasa butonul de start inaintea introducerii anumitor date obligatorii pentru adaugare/editare/stergere. Astfel, va fi afisat un mesaj de eroare.

view

ERROR! BAD INPUT

1. Utilizatorul selecteaza un client si unul/ mai multe produse pentru a realiza o comanda, sau cantitatea care exista pentru produsul respectiv este mai mica decat cea ceruta, acest lucru fiind de asemenea semnalat.

view

ERROR! BAD INPUT

1. Proiectarea

3.1 Decizii de proiectare

Pentru proiectarea aplicației, am folosit patru pachete principale, după structura **Layered Architecture.** Această structură presupune împărțirea claselor pe proiecte în funcție de scopul acestora și o succesiune logică între ele, astfel: baza de date se apelează prin pachetul dataAccess. Implemetarea operațiilor la nivel de cod Java, din bussinesLogic se face prin apelul metodelor din dataAccess. Implemetările din bussinesLogic se vor apela mai apoi în pachetul destinat implementării interfeței grafice cu utilizatorul.

Toate pachetele pot apela pachetul model, fiind pachetul în care sunt reținute datele specific pentru Client, produs și comandă.

Pachetul model contine patruclase: Customer, Product, Order si OrderDetail.

Clasa Client este alcătuită din patru variabile instanță: o variabilă întreagă care reține id-ul clientului și patru variabile de tip String în care sunt reținute numele clientului, email-ul acestuia, adresa și respectiv, numarul acestuia. S-au creat getteri și setteri pentru aceste variabile instanță. De altfel, pentru variabila id, exista o metoda static suplimentară pentru a incrementa automat valoarea acesteia, descrisa in clasa ReflexionExample.

Clasa Product este alcătuită din patru variabile instanță, astfel: o variabilă întreagă care reține id-ul produsului, o variabilă de tipul String în care este stocat numele produsului, o variabilă de tipul Int care reprezintă prețul acestuia și o variabilă intreagă pentru a reține cantitatea totală a produsului din depozit. S-au creat getteri și setteri pentru toate aceste variabile.

Clasa Order continte doar variabilele instanta pentru a retine numarul comenzii si clientul care a comandat.

Clasa OrderDetail este alcătuită din patru variabile instanță, toate întregi, în care sunt reținute id-ul comenzii, id-ul clientului, id-ul produsului și cantitatea comandată. S-au creat getteri și setteri pentru aceste variabile.

Pachetul dataAccess conține clasele specifice pentru fiecare din clasele existente în pachetul model, clasa speccifică pentru conectarea la baza de date și clasa în care este implementată metoda reflexiei.

Clasa ConnectionFactory are 5 variabile instanță statice, care reprezintă datele pentru conexiunea la baza de date MySql, și anume: LOGGER, DRIVER – numele driver-ului, DSURL – locația bazei de date, USER și PASS – user-ul și parola pentru a accesa baza de date.

Această clasă implementează o metodă *getConnection(),* în care se creează conexiunea la baza de date și returnează o vazriabilă în care au fost reținute datele despre această conexiune. De asemenea, s-au implementat metode pentru inchiderea conexiunii, a statement-ului și a rezultatului.

Clasa AbstractDataAccess implementează mai multe metode, printre care *selectAll(List<Customer>)* care returnează toate datele existente despre un obiect, pentru toate obiectele din tabelă. În primă fază, se creează obiectele specific conexiunii, statementului – pentru query – și rezultatul returnat de acest query. Într-un bloc try-catch se încearcă conexiunea la baza de date , se creează interogarea și se execută și se adaugă toate rezultatele într-o listă de tip Object. Metodele realizate in cadrul acestei clase permit refolosirea pentru toate clasele care extind Object,

Pachetul dataAccess mai conține clasele ProductAccess și OrderAccess, care implementează aceleași metode care au fost implementate în clasa CustomerAccess, adaptate.

///reflection

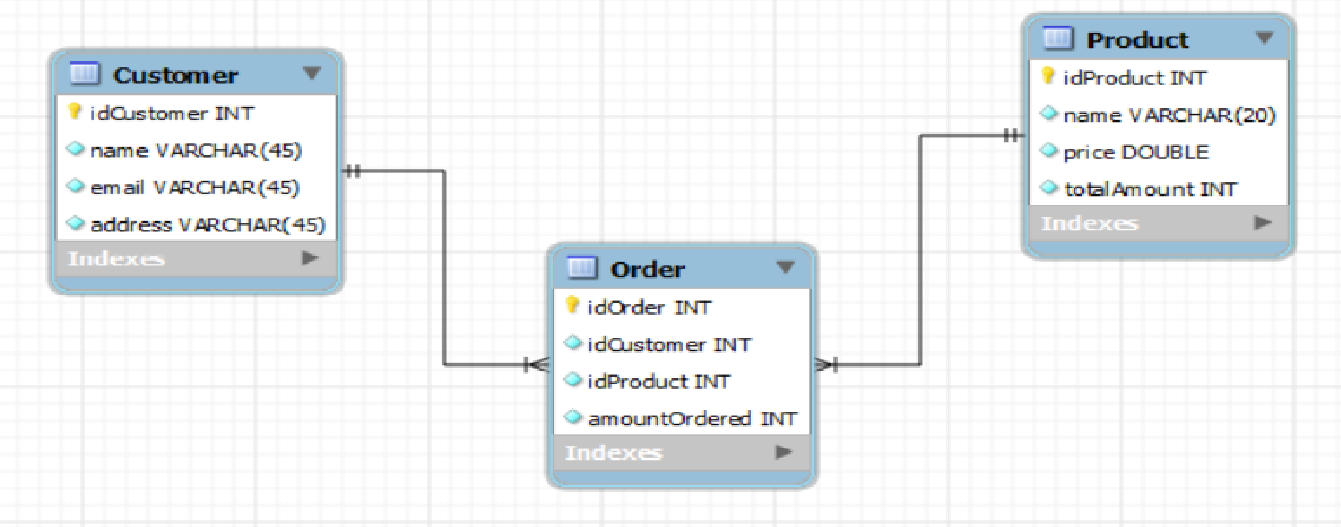
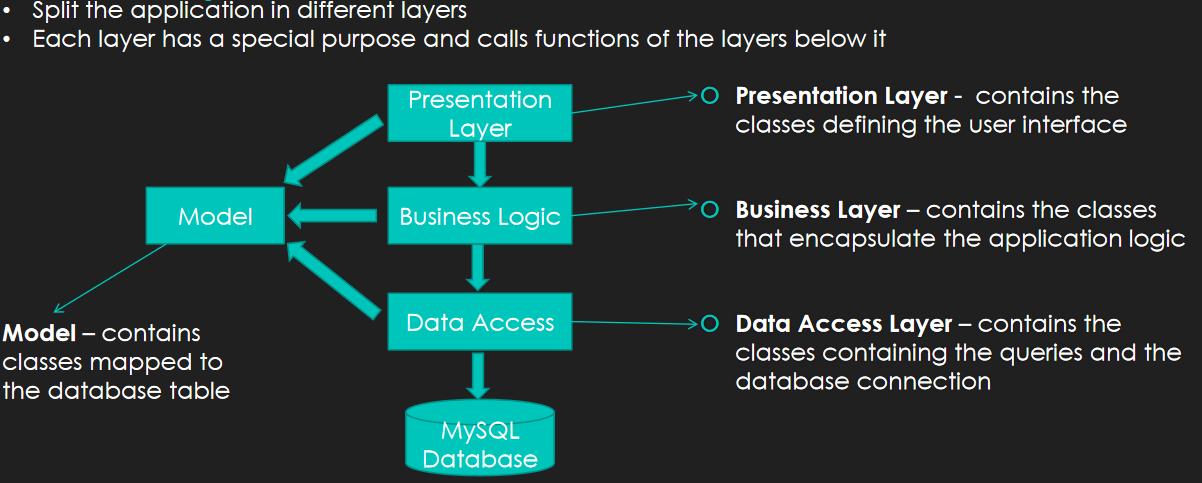


Diagrama bazei de date.

Pachetul bussinesLogic conține trei clase:CleintLogic, ProductLogic, OrderDetailLogic și OrderLogic. Aceste clase conțin fiecare o listă de tipul Cleint, Product, OrderDetail, respectiv Order. Am implementat metode care returnează lista de obiecte, metodă care inițializează lista de obiecte și metodă care adaugă în lista de obiecte.

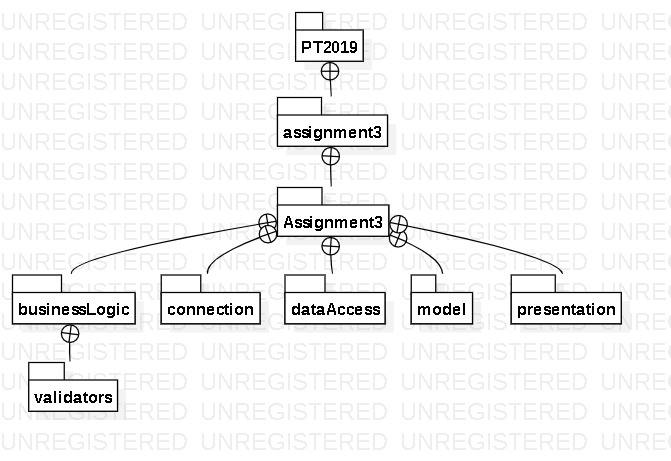
S-au implementat metodele pentru adaugare, editare și ștergere client, produs sau, respective, comandă..

Pachetul presentation conține 25 clase pentru implementarea interfeței grafice cu utilizatorul. Interfața grafică cu utilizatorul (GUI) este o interfață care permite utilizatorului interacțiunea prietenoasă cu aplicația.

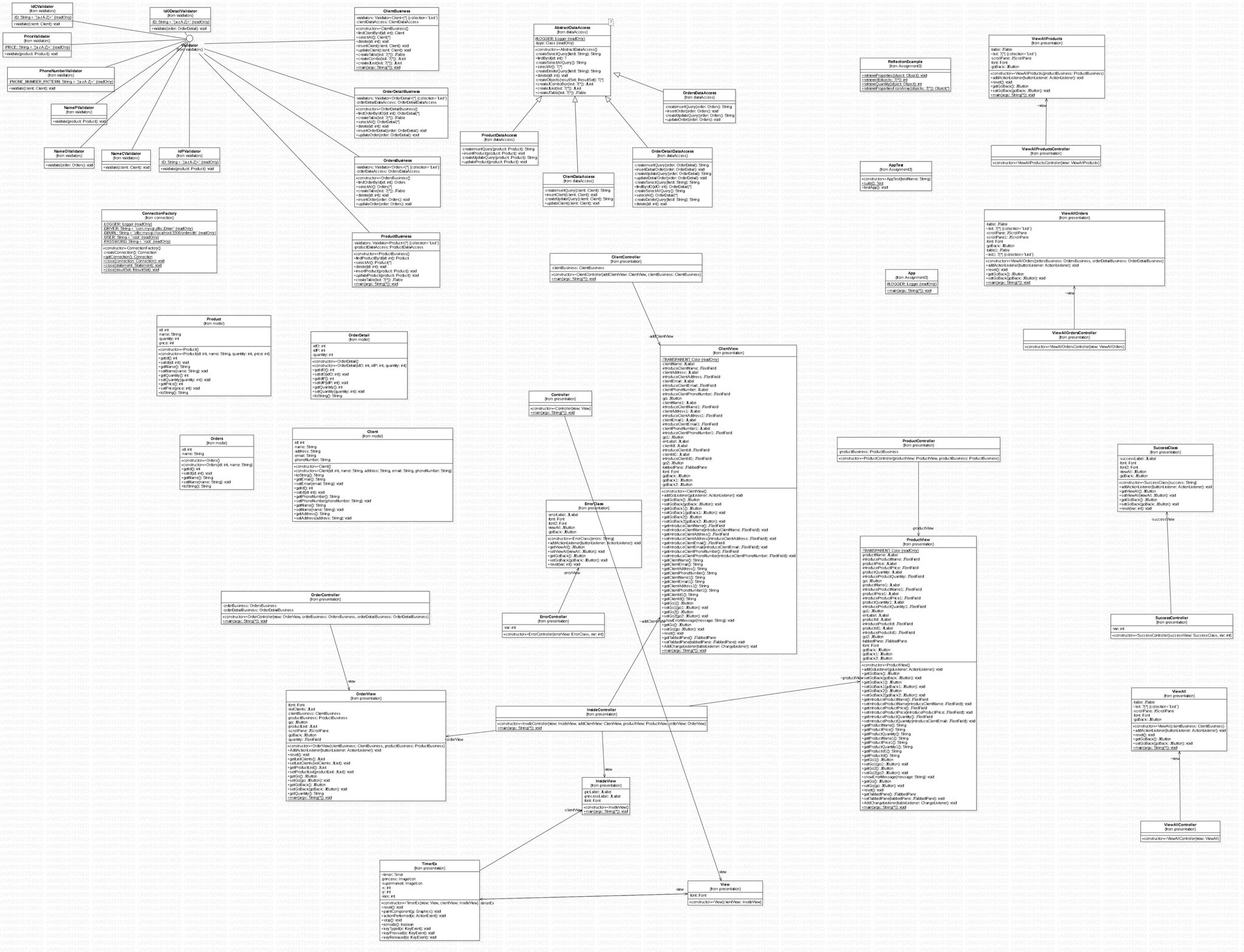


3.2. Diagrame UML

3.2.1. Diagrama de pachete



3.2.2. Diagrama de clase



3.3. Structuri de date

Pentru stocarea datelor despre clienti, produse, comenzi s-a folosit o structura de tip ArrayList, datele fiind introduse ulterior in tabele. Aceste tipuri de date au lungime variabila, care se modifica odata cu operatia realizata. Totodata, datorita faptului ca adaugarea unui client se face la sfarsitul cozii, parcurgerea listei presupune un timp necesar O(n).

3.4. Proiectare clase

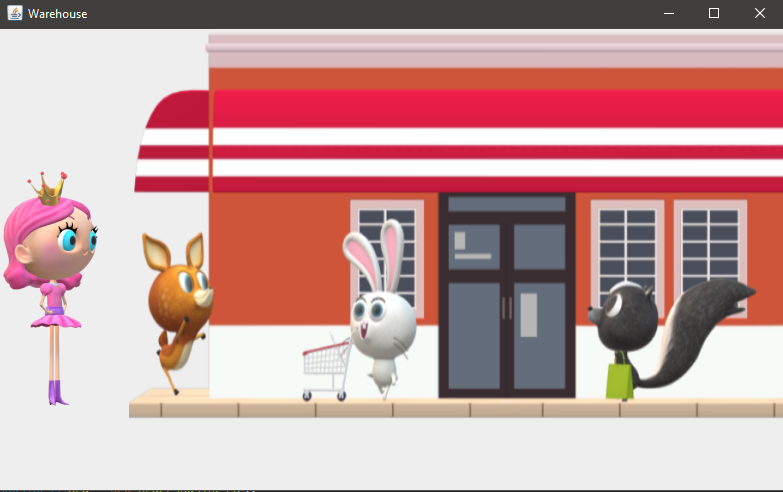
Proiectarea claselor imiplica folosirea unor interfete predefinite in cadrul libariilor Java, implementarea interfetei ActionListener de catre clasa Controller permite „ascultarea” si interpretarea comenzilor date de utilizator.

3.5. Algoritmi

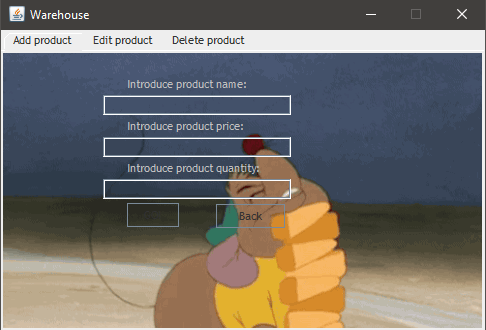
Algoritmul care se remarca face referire la modul de implementare a query-urilor.

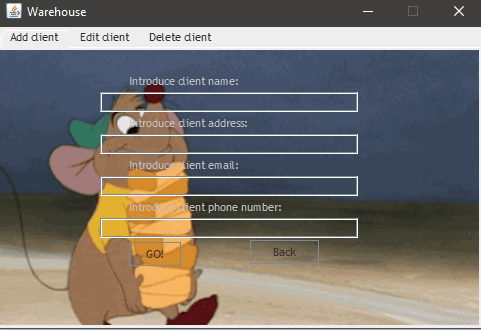
3.6. Interfata utilizator

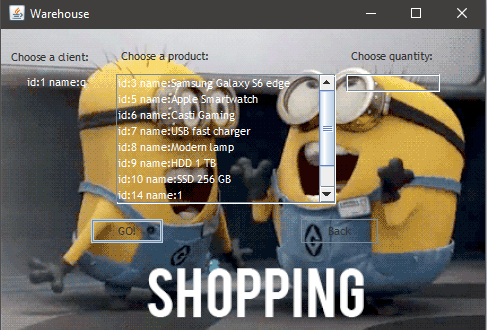
Interfata simplista permite folosirea aplicatiei care simuleaza cozile de catre orice utilizator, cu sau fara cunostinte in unul dintre domeniile informaticii.











1. Implementare
   1. Pachetul Model
      1. Clasa Client

Clasa Client defineste cateva atribute definitorii: ID-ul clientului, numele clientului, adresa, adresa de email precum si numarul de telefon al clientului. Clasa implementeaza un constructor specific, care primeste ca argument atributele definitorii descrise mai sus. Variabilele instanta alea clasei sunt private din motive de protejare si ascundere a informatiilor legate de implementare fata de utilizator, motiv pentru care clasa contine metode accesoare si mutatoare. Aceasta clasa se ocupa si de afisarea „prietenoasa” a unui client, pentru a transpune imaginea acestuia cat mai aproape de realitate. Toate tipurile folosite sunt tipuri de data.

* + 1. Clasa Product

Clasa Product defineste cateva atribute definitorii: ID-produsului, numele produsului, cantitatea precum si pretul produsului. Clasa implementeaza un constructor specific, care primeste ca argument atributele definitorii descrise mai sus. Variabilele instanta alea clasei sunt private din motive de protejare si ascundere a informatiilor legate de implementare fata de utilizator, motiv pentru care clasa contine metode accesoare si mutatoare. Aceasta clasa se ocupa si de afisarea „prietenoasa” a unui produs, pentru a transpune imaginea acestuia cat mai aproape de realitate. Toate tipurile folosite sunt tipuri de data.

* + 1. Clasa Order

Pentru o mai mare modularite, clasa Order contine doar Id-ul comenzii si numele unui client, realizand legatura cu clasa OrderDetail;

* + 1. Clasa OrderDetail

Clasa OrderDetail contine un id al comenzii, al produsului acestea reprezentand cheia primara a tabelului aferent. Am implementat aceasta clasa pentru a permite ca un client sa cumpere mai multe produse. Variabilele instanta alea clasei sunt private din motive de protejare si ascundere a informatiilor legate de implementare fata de utilizator, motiv pentru care clasa contine metode accesoare si mutatoare. Aceasta clasa se ocupa si de afisarea „prietenoasa” a unei comenzi, pentru a transpune imaginea acestuia cat mai aproape de realitate. Toate tipurile folosite sunt tipuri de data

1. Rezultate

Verificarea flow-ului clientilor este realizata in cadrul interfetei grafice, intr-un scrollpanel care permite vizualizarea unui numar mare de clienti/produse/comenzi.

1. Concluzii

Acest proiect m-a ajutat la intelegerea diferitelor concepte ale programarii orientate pe obiecare, care au fost aprofundate pe parcursul dezvoltarii aplicatiei. Structurarea in pachete si intelegerea nevoi de a structura astfel, crearea unei clase care se ocupa de manipularea unui singur obiect si restrangerea metodelor de rezolvare in mai putin de 30 de linii de cod, respectiv evitarea structurilor imbricate la sugestia laborantului au dus in final la descoperirea unor metode de rezolvare mai simpliste, care pot fi urmarite mai usor. De asemenea, a avut un rol major in intelegera concurentei in programare, prin prisma threadurilor care lucreaza simultan, concent nemaiintalnit.

1. Bibliografie

* <https://stackoverflow.com/>
* <http://users.utcluj.ro/~igiosan/>
* inf.ucv.ro/documents/tudori/laborator8\_53.pdf
* coned.utcluj.ro/~salomie/PT\_Lic/marcel99https://www.baeldung.com/java-pdf-creation

<https://giphy.com/gifs/love-funny-cute-10Tpkroztpn2gw>

https://javarevisited.blogspot.com/2016/10/how-to-check-if-resultset-is-empty-in-Java-JDBC.html