DOCUMENTATIE TEMA 3

ORDER MANAGEMENT

Nume student: Buzilă-Gârda Andra-Maria

Grupa: 30229

Profesor de laborator: assist. Antal Marcel

Cuprins:

1. Cerinte functionale 3
2. Obiective 4
   1. Obiectiv principal 4
   2. Obiective secundare 4
3. Analiza problemei 4
4. Proiectare 6
   1. Alegerea structurilor de date 6
   2. Diagrama de clase 6
5. Implementare 7
6. Verificare 12
7. Concluzii si dezvoltari ulterioare 13
8. Bibliografie
9. **Cerinte functionale:**

Bazele de date relationale sunt folosite pentru a inmagazina produse, clienti si comenzi pentru

acestia. Aplicatia trebuie structurata in pachete folosind o arhitectura stratificata sugerata in prezentarea suport si trebuie folosite anumite clase in crearea acestei aplicatii, clase precum: Model classes, Business Logic classes, Presentation classes, Data access classes etc.

Pentru a rezolva cerinta trebuie sa respectati urmatoarele criterii de notare:

* Folositi un limbaj de programare orientat pe obiecte avand clase care sa contina maxim 300 linii si

metode cu maxim 30 linii, exceptie facand clasele folosite pentru interfata cu utilizatorul;

* Folositi javadoc pentru documentarea claselor si generati fisierele javadoc;
* Folositi baze de date relationale pentru a stoca datele pentru aplicatie, avand minim 3 tabele: Client, Produs si Comanda
* Creati o interfata grafica care ca includa:
* O fereastra pentru operatiile legate de tabelul client cu urmatoarele operatii: adaugare de client nou, stergerea unui client, editarea informatiilor despre un anumit client si afisarea tuturor clientilor intr-un tabel (cu JTable);
* O fereastra pentru operatiile legate de tabelul produs cu urmatoarele operatii: adaugare de produs nou, stergerea unui produs, editarea informatiilor despre un anumit produs si afisarea tuturor produselor intr-un tabel (cu Jtable);
* O fereastra pentru crearea de noi comenzi- utilizatorul va putea selecta un produs existent, un client existent si sa insereze cantitatea dorita din acel produs. Daca acea cantitate este mai mare decat cantitatea valabila din produsul respectiv se va afisa un mesaj de eroare. Dupa ce s-a facut comanda se va decrementa cantitatea produsului.
* Folositi tehnica de reflexie pentru a crea metode care primesc liste de obiecte si genereaza tabelele corespunzatoare cu tot cu header-ele corespunzatoare si populati valorile din tabel cu elementele din lista.
* Sa aveti arhitectura structurata( aplicatia va contine cel putin 4 pachete: dataAccessLayer, businessLayer, model, presentation);
* Creati „nota de plata” in momentul in care se creeaza comanda, reprezentata printr-un pdf sau text file
* Folosind tehnica reflexiei, creati clase generice care sa faca legatura cu baza de date, reusind astfel sa se editeze obiecte, sa se stearga sau sa se introduca. Interogarile sunt generate cu tehnica reflexiei.

1. **Obiective** 
   1. **Obiectiv principal**

Proiectarea si implementarea unei aplicatii care lucreaza cu baza de date, avand date anumite

cerinte de implementare.

* 1. **Obiective secundare**

|  |  |  |
| --- | --- | --- |
| **Obiectiv** | **Descriere** | **Capitol** |
| Dezvoltarea de use care-uri si scenarii | Dorim sa cunoastem ce se intampla in momentul apasarii butoanelor aflate in interfata grafica, cea vazuta de utilizator | 3 |
| Alegerea structurilor de date | Modul de reprezentare in memorie | 4 |
| Impartirea problemelor pe clase | Cream clasele de care avem nevoie | 4 |
| Implementarea solutiilor | Descrierea modului in care am creat fiecare clasa si fiecare metoda in parte | 5 |
| Verificare | Cu ajutorul validatorilor | 6 |

1. **Analiza problemei**

Utilizatori:

* Manageri sau detinatori de magazine online de haine;
* Manageri dau detinatori de magazine de orice fel;
* Manageri dau detinatori de restaurante.

Pre-conditii:

* Utilizatorul a introdus corect id-ul clientului in interfata pentru client;
* Utilizatorul a introdus corect numele clientului in interfata pentru client;
* Utilizatorul a introdus corect adresa clientului in interfata pentru client;
* Utilizatorul a introdus corect email-ul clientului in interfata pentru client;
* Utilizatorul a introdus corect vartsa clientului in interfata pentru client;
* Utilizatorul a introdus corect id-ul produsului in interfata pentru produs;
* Utilizatorul a introdus corect numele produsului in interfata pentru produs;
* Utilizatorul a introdus corect cantitatea produsului in interfata pentru produs;
* Utilizatorul a introdus corect id-ul comenzii in interfata pentru comanda;
* Utilizatorul a introdus corect id-ul clientului in interfata pentru comanda;
* Utilizatorul a introdus corect id-ul produsului in interfata pentru comanda;
* Utilizatorul a introdus corect cantitatea de produs in interfata pentru comanda.

Post-conditii;

* Se afiseaza tabelul cu clientii in interfata grafica pentru client;
* Se face insert-ul unui client;
* Se face delete-ul unui client;
* Se face update-ul unui client;
* Se afiseaza tabelul cu produsele in interfata grafica pentru produs;
* Se face insert-ul unui produs;
* Se face delete-ul unui produs;
* Se face update-ul unui produs;
* Se afiseaza tabelul cu comenzile in interfata pentru comanda;
* Se creeaza o comanda noua si se face update la cantitatea produsului din clasa produs.

Modul de functionare:

* Utilizatorul introduce de la tastatura datele necesare crearii unui nou client apoi apasa pe butonul insert.

Se va crea un client nou si se va introduce in baza de date.

* Utilizatorul introduce datele necesare stergerii unui client si apoi apasa pe butonul delete. Se va sterge

din baza de date clientul respectiv.

* Utilizatorul va apasa pe butonul view all pentru a vedea tabelul cu toti clientii in interfata.
* Utilizatorul va introduce datele necesare editarii unui client si apoi apasa pe butonul update. Clientul

respectiv va avea datele modificate.

* Utilizatorul introduce de la tastatura datele necesare crearii unui nou produs apoi apasa pe butonul

insert. Se va crea un produs nou si se va introduce in baza de date.

* Utilizatorul introduce datele necesare stergerii unui produs si apoi apasa pe butonul delete. Se va sterge

din baza de date produsul respectiv.

* Utilizatorul va apasa pe butonul view all pentru a vedea tabelul cu toate produsele in interfata.
* Utilizatorul va introduce datele necesare editarii unui produs si apoi apasa pe butonul update. Produsul

respectiv va avea datele modificate.

* Utilizatorul apasa pe butonul create din interfata pentru comanda dupa ce a introdus toate datele necesare

si se va crea o noua comanda care va fi introdusa in baza de date.

* Utilizatorul apasa pe butonul view all din interfata pentru comanda pentru a vedea tabelul cu toate

comenzile.

client GUI

view all

user Display client table

insert

update

delete product GUI

view all Display product table

insert

update

delete order GUI

view all

create Display order table

Figura 1: REPREZENTARE DIAGRAMA USE CASE

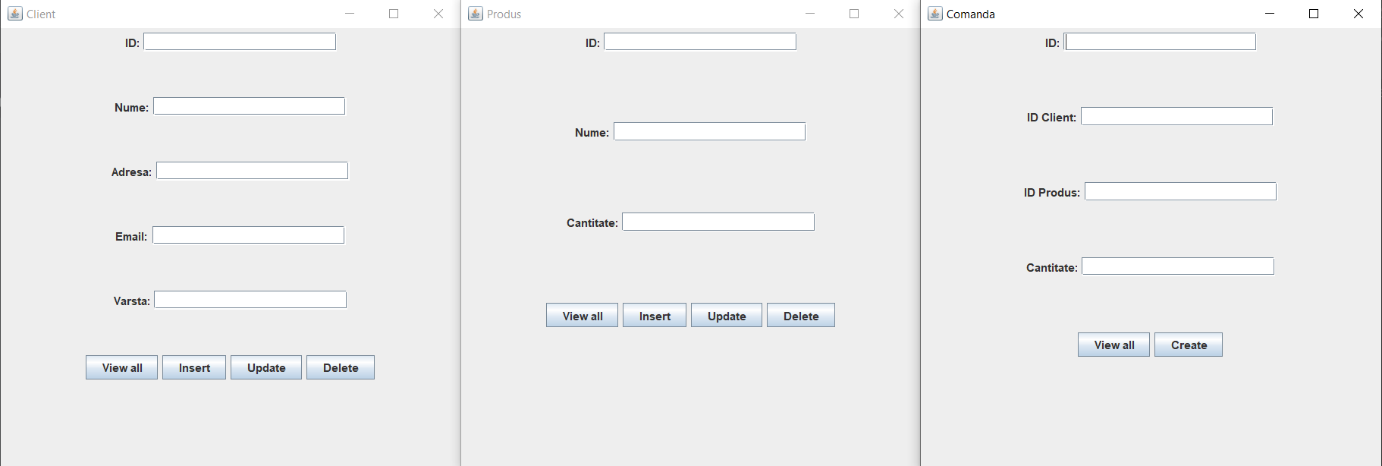


Figura 2: INTERFETELE VAZUTE DE UTILIZATOR

1. **Proiectare**
   1. **Alegerea structurilor de date**

Am folosit liste in cea mai mare parte a programului. Avem liste de validatori in clasele din Business

Layer. Metodele care ne ajuta la generarea rezultatelor din tabele vor returna la randul lor niste liste de obiecte. In clasele din view am siruri de String-uri iar in controller exista, de astemenea, mai multe liste, de ex liste in care am pus campurile pe care le vom modifica cand se executa update, mai sunt diferite liste de produse, iar in clasele de afisare de date din controller se gasesc liste de siruri de String-uri.

* 1. **Diagrama de clase**

Unified Modeling Language (prescurtat UML) este un limbaj standard pentru descrierea de modele și

specificatii pentru software. UML ofera o larga gama de diagrame pentru modelarea diferitelor situatii in cadrul unui proiect de dezvoltare software.

Diagrama de clasa este folosita pentru reprezentarea vizuala a claselor ai a interdependentelor, taxonomiei si a relatiilor de multiplicitate dintre ele. Aceste diagrame sunt folosite si pentru reprezentarea concreta a unor instante de clasa, asadar obiecte, si a legaturilor concrete dintre acestea.

Am create aceasta aplicatie luamndu-ma dupa structura Model-View-Controller (MVC) dar avem si alte

pachete precum dao si bll in felul urmator: am facut 3 clase de tipul ClassView (pentru fiecare clasa din model) in care am schitat imaginea de ansamblu a simulatorului, unde utilizatorul introduce datele, contine butoane, textField-uri, label-uri, tabele si un textPane in care se va afisa rezultatul pe parcursul evolutiei. Am construit apoi clasa ClassController, unde am manipulat datele introduse prin clasa ClassView, iar Modelul este reprezentat de clasele Client, Comanda si Produs si am creat, de asemenea, clasa Main si FileData. Mai avem si clasele din Business layer: clientBLL, comandaBLL, produsBLL si clasele din dao: clientDao, produsDao, comandaDao dar si abstractDao.

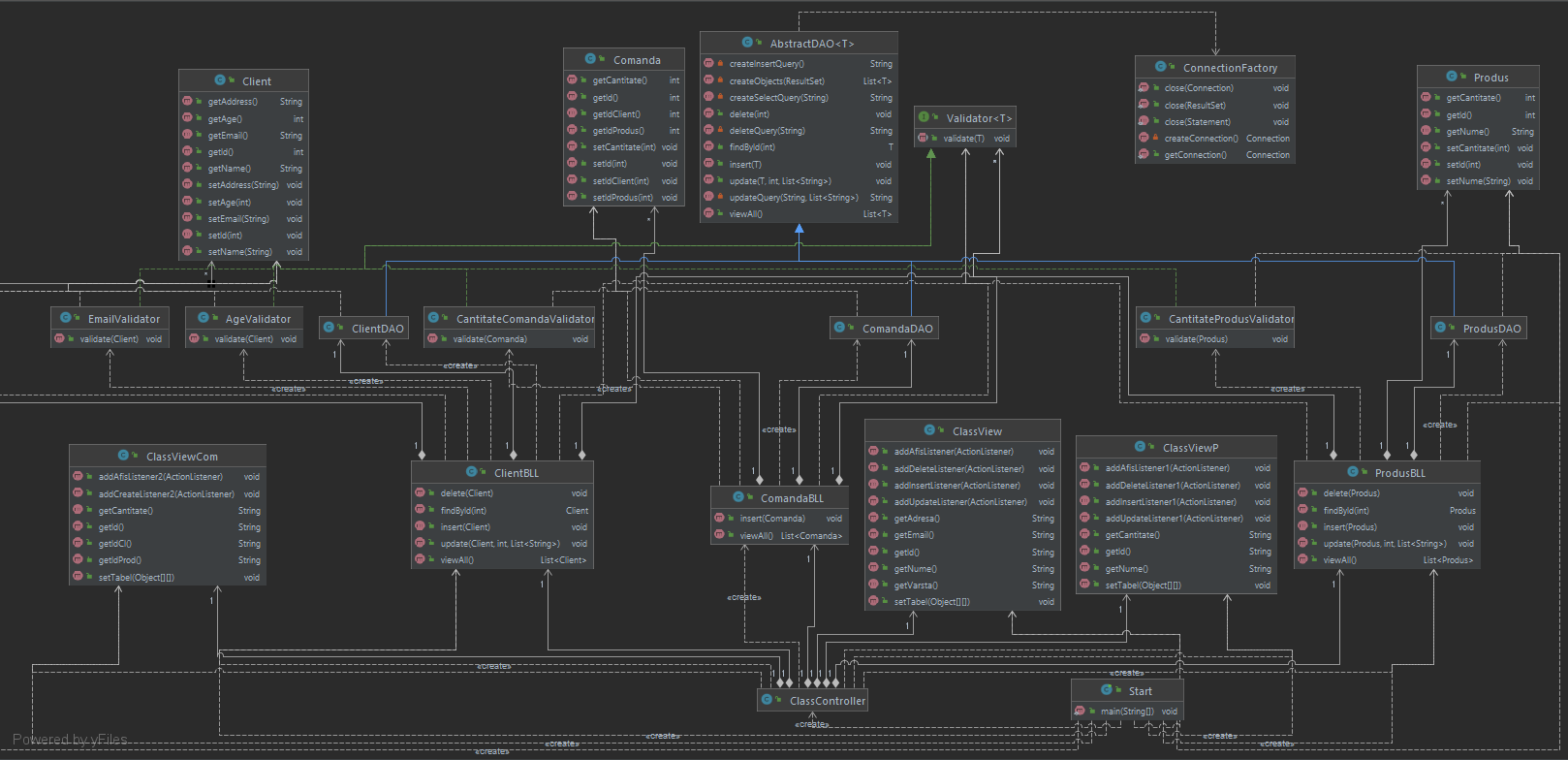
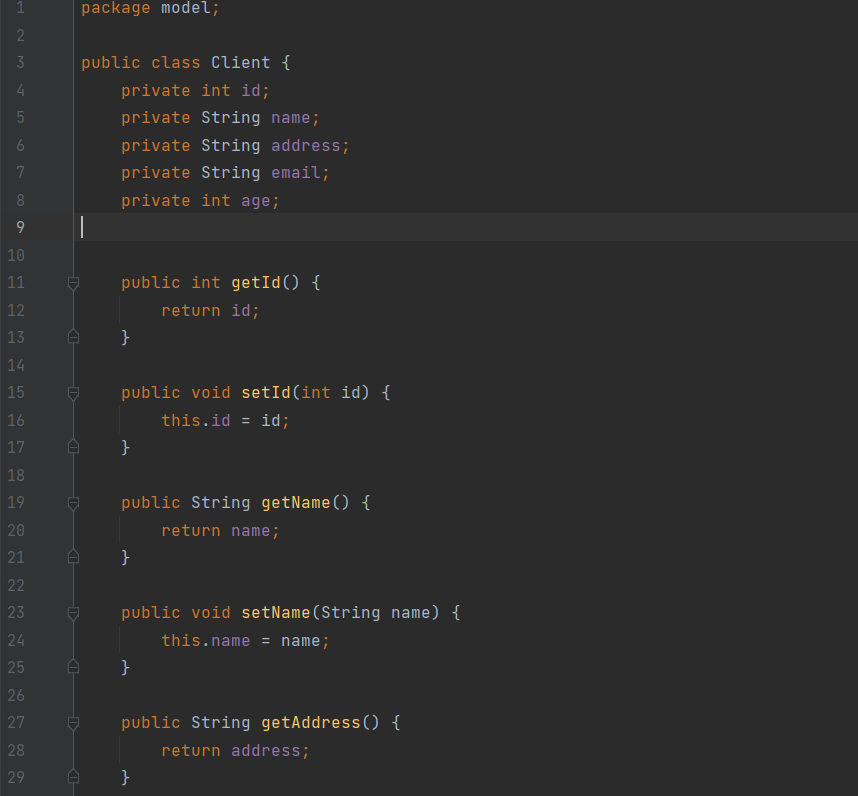


Figura 3: DIAGRAMA UML



1. **Implementare**

**Pachetul model:**

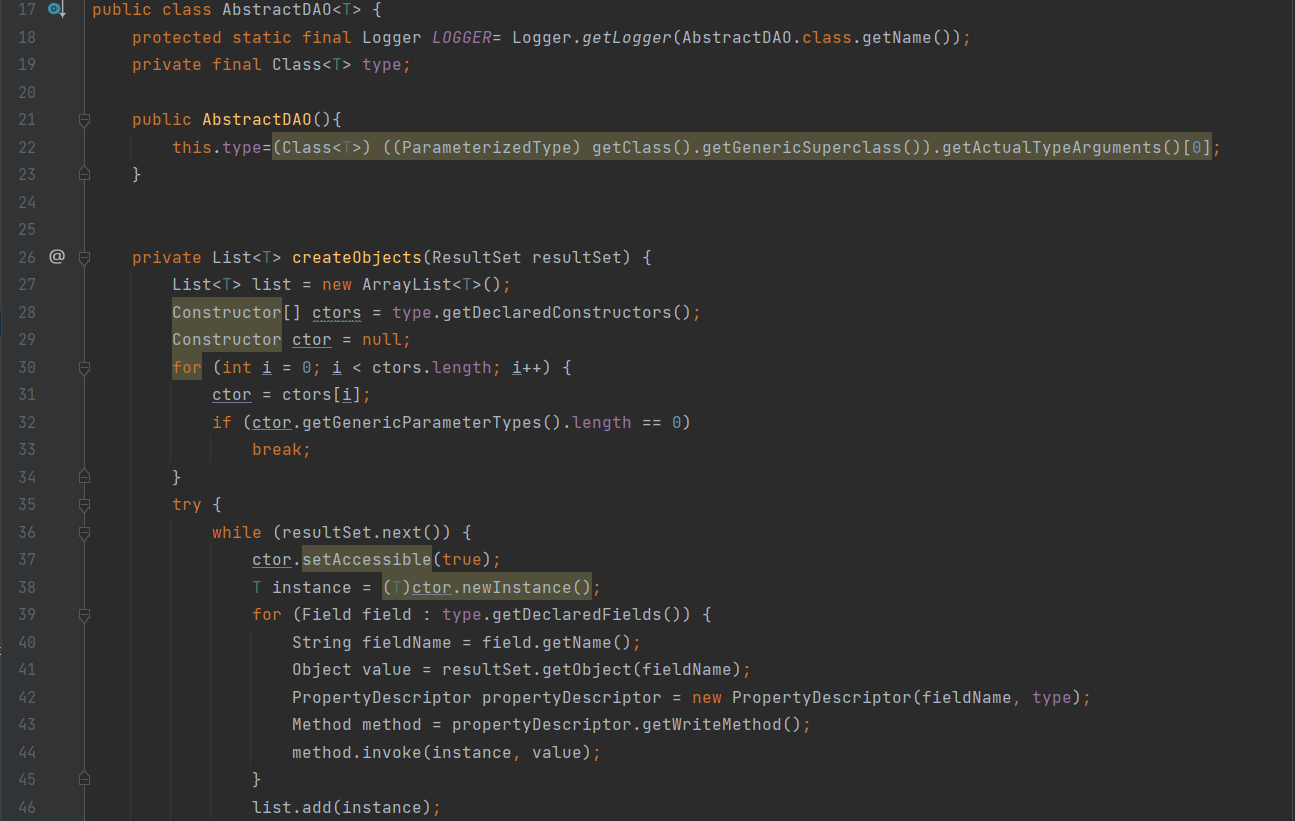
Clasele din pachetul model sunt practic baza intregului program. Acestea se numesc: Client, Produs si

Comanda, iar cu datele extrase din baza de date se vor crea instante de aceste clase. Ele nu contin constructori dar, in schimb, contin getters si setters pentru fiecare variabila instanta a acestor clase. Variabilele instanta din aceste clase trebuie sa poarte aceeasi denumire ca si denumirile coloanelor corespondente din tabelul cu acelasi nume ca si numele clasei.

**Pachetul DAO:**

Acest pachet contine clasele clientDao, produsDao si comandaDao, care extind clasa abstracta AbstractDao, deci ele nu contin nicio metoda, deoarece in momentul cand le apelam, vom apela, de fapt, metodele din AnstractDao. In aceasta clasa avem metodele urmatoare: viewAll, insert, update si delete. Metoda viewAll va returna o lista de obiecte diferite in functie de clasa pe care o apelam. Lista aceea va contine absolut toate valorile variabilelor instanta din toate obiectele de acelasi tip. Se genereaza un query pe care o vom transmite la baza de date, care va arata in felul urmator „SELECT \* FROM ..” apoi va fi titlul tabelului, in functie de clasa. Metoda insert va insera in baza de date un nou client instantiat in aceeasi clasa. De data aceasta query-ul va arata in felul urmator:” INSERT INTO tabel\_name VALUES (?, ?, ?..)”, unde semnul acela al intrebarii va fi inlocuit de valorile propriu-zise din fiecare camp din baza de date. In metoda delete se va proceda in mod asemanator, cu interogarea urmatoare: „ DELETE FROM table\_name WHERE field =?”, unde field-ul este id, id-ul fiind cheie primara in tabel, astfel fiind unic determinat. Deci dupa ce se face conectarea la baza de date se executa acele interogari. Ultima metoda, cea de update va modifica practic in tabel anumite valori introduse de utilizator. Interogarea este:

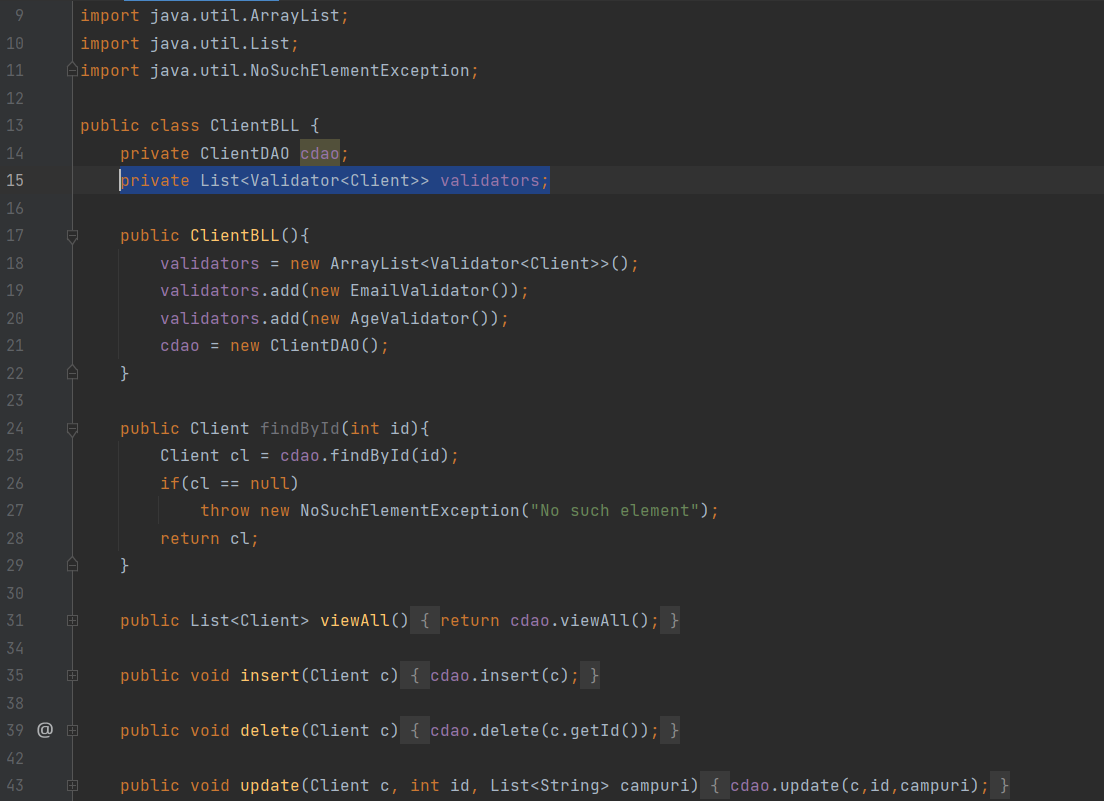
„UPDATE table\_name SET column1=?, column2=?.. WHERE field=?”, iar acele semne de intrebare reprezinta valorile introduse de utilizator, field-ul este id-ul. In toate aceste metode din clasa abstractDao se realizeaza prima data conectarea la baza de date si abia apoi se citesc sau se sciu anumite valori in sau din baza de date.





**Pachetul BLL:**

In acest pachet avem urmatoarele clase: ClientBll, ProdusBll si ComandaBll. In aceste clase nu se intampla altceva decat sa se apeleze metodele create in abstractDao dar pentru obiectele potrivite. Se vor crea instante de clase din pachetul dao, corespunzator numelui si din acelea se vor apela.



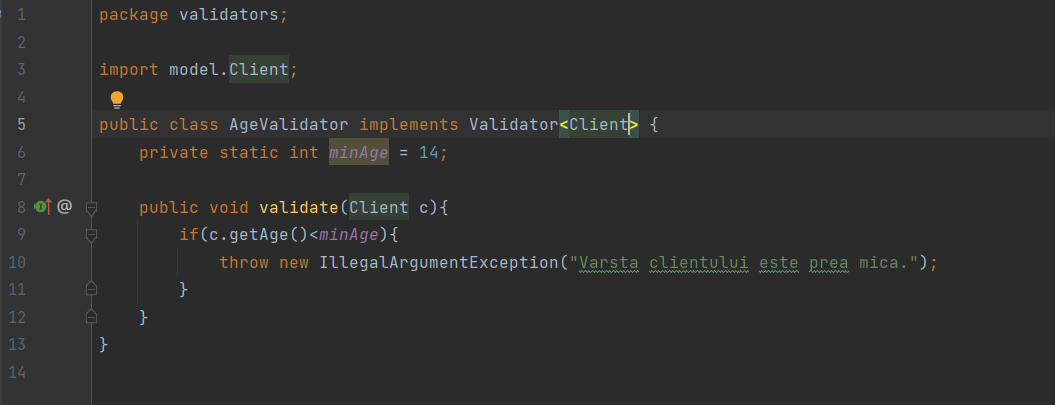
**Pachetul connection:**

Pachetul connection contine doar clasa ConnectionFactory care realizeaza conectarea la baza de date de la 0, adica se introduce user-ul si parola in locurile corespunzatare dupa ce se urmeaza un path pentru a ajunge la baza de date si la conectare.



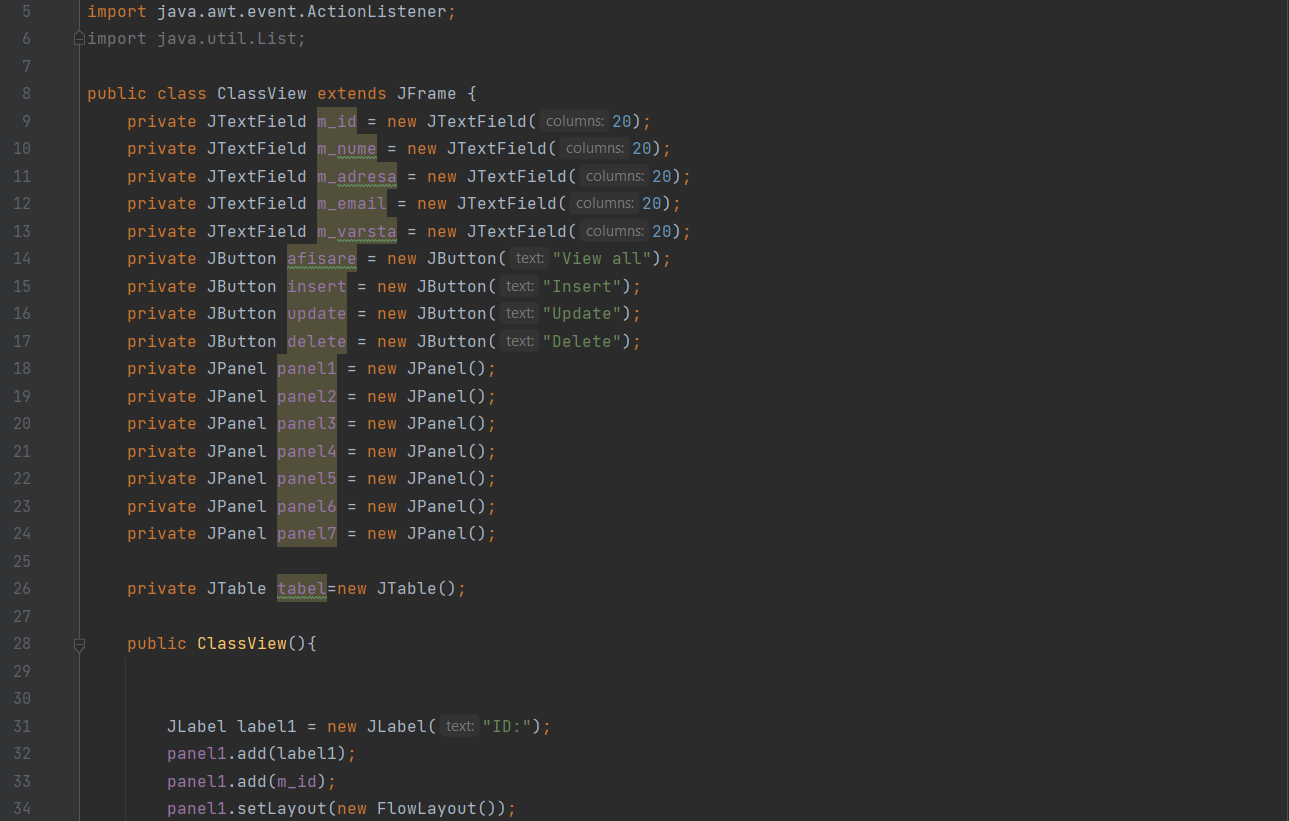
**Pachetul validators:**

In acest pachet exista mai multe clase care ajuta la verificarea corectitudinii introducerii anumitor valori de catre utilizator in interfata grafica. Daca nu sunt respectati acei validatori vor aparea diferite mesaje de eroare. Validatorii acestia se vor verifica in clasele din pachetul bll.



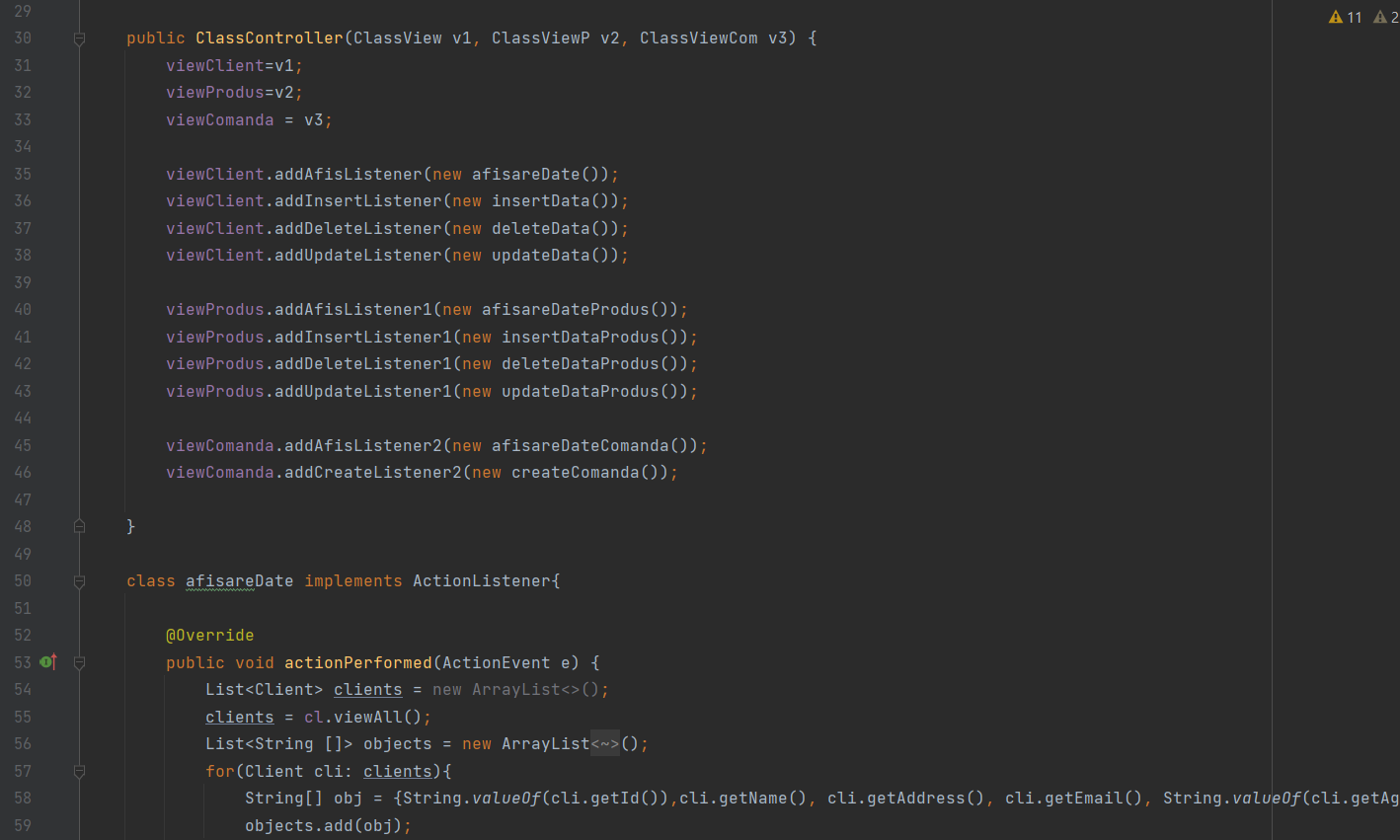
**Pachetul view:**

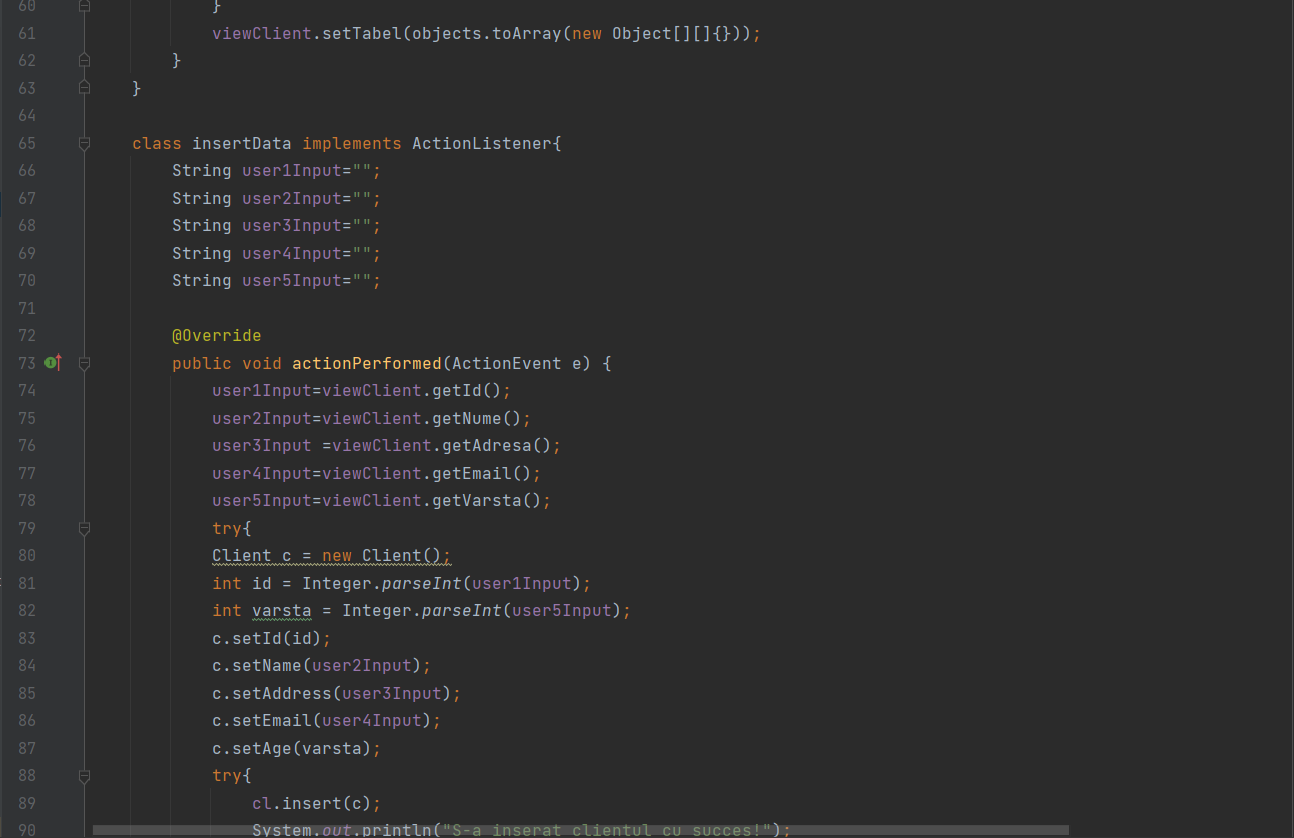
Acest pachet contine 3 clase de forma ClassView, acestea contureaza imaginea de ansamblu pe care utilizatorul o vede in momentul in care deschide interfata. Sunt 3 clase deorece avem 3 clase in pachetul model. Fiecare dintre acestea au in componenta lor panel-uri, jtable-uri, jbuttons, label-uri, toate acestea se vor combina si se vor aseza intr-un panel la sfarsit, acesta fiin principalul panel din interfata. De asemenea, tot aici se gasesc diferite metode care returneaza datele introduse, dar si metode de setter care vor afisa rezultatul in tabelul de pe interfata. Aceste metode sunt prezente in toate 3 clasele.



**Pachetul controller:**

Aici se afla doar clasa controller care, la randul ei, contine mai multe clase pentru fiecare buton apasat. Deci in aceasta clasa controller se efectueaza operatiile din spate, adica ceea ce se intampla in momentul apasarii fiecarui buton. Avem o singura clasa deoarece, avand mai multe clase in aceeasi clasa, nu influenteaza cu nimic daca se amesteca intre ele. In pachetul controller se creeaza instante de clase din pachetul bll, care la randul lor vor crea instante de clase din dao si astfel se realizeaza conectarea si practic tot procesul.





1. **Verificare**

Verificarea introducerii datelor corecte se face in pachetul validator, clase ce vor fi apelate din clasele din pachetul bll. Verificarea corectitudinii functionarii sa face prin exemple.

1. **Concluzii si dezvoltari ulterioare**

Se pot crea mult mai multe clase, precum sediu sau transport, pentru a imbunatatii productivitatea firmei

care foloseste aceasta aplicatie pentru a isi multumi clientii. De asemenea se pot adauga mai multe campuri in tabele, cum ar fi data la care s-a facut comanda pentru a se cunoaste timpul de intarziere in caz ca acesta exista, sau se poate adauga numarul de telefon, pentru a contacta mai usor clientul in cazul in care acesta nu raspunde pe email.

In concluzie, am invatat sa lucrez cu o baza de date in urma cearii acestei aplicatii. Mi-am dezvoltat abilitatile si modul de lucru in acest limbaj de programare. Am invatat cum sa creez mai multe interfete cu utilizatorul si sa le afisez in acelasi timp pe ecran.

1. **Bibliografie**

* Youtube
* [Java JTable - javatpoint](https://www.javatpoint.com/java-jtable)
* [Using Java Reflection (oracle.com)](https://www.oracle.com/technical-resources/articles/java/javareflection.html)
* Materialele de la laborator