**Simulator de comenzi**

Documentatie

Cîrciu Mihnea Teodor

Grupa 30227

**Cuprins**

1. Obiectiv
2. Analiza problemei
3. Implementare
   1. Diagrama UML
   2. Baza de date –SQL-
   3. Pachete si clase
   4. Metode si impementare
   5. JavaDoc
   6. GUI
4. Testare
5. Concluzii
6. Bibliografie
7. **Obiectiv**

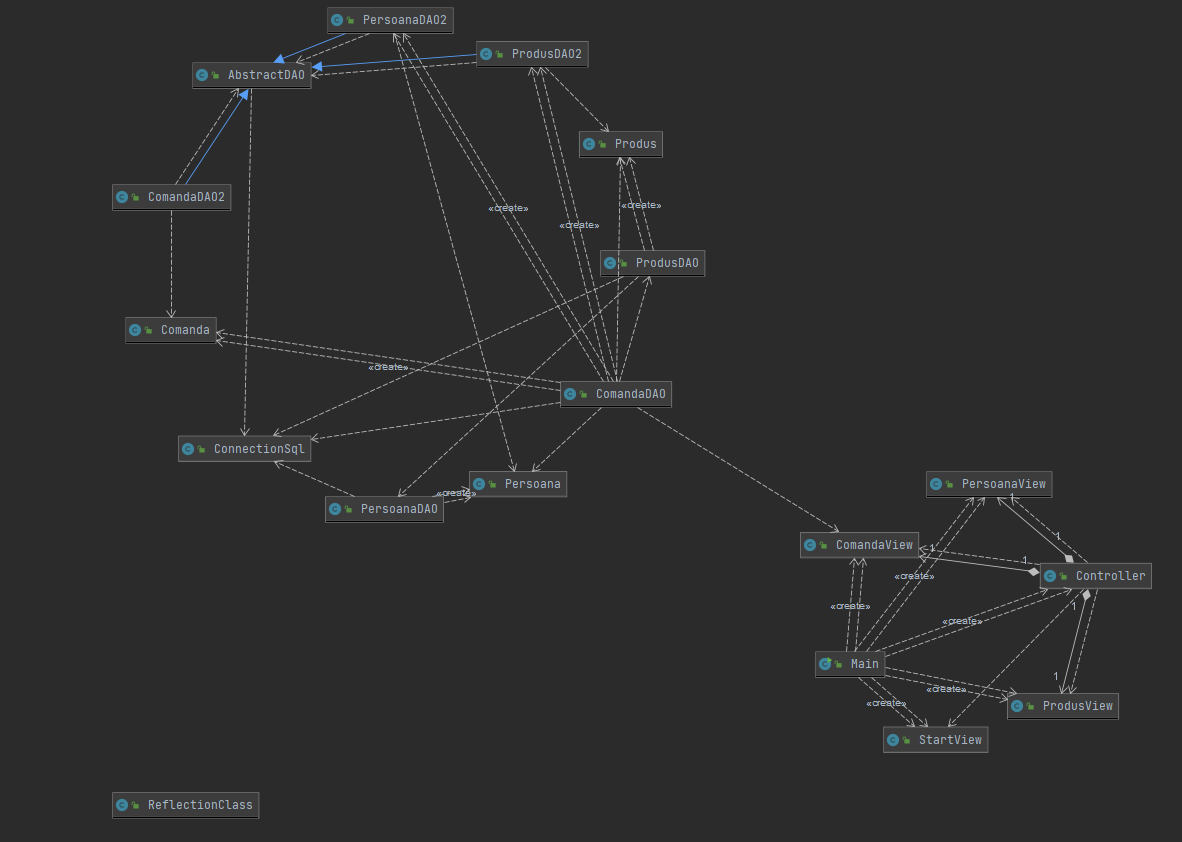
Cerinta acestei teme este reprezentata de un simulator de comenzi bazat pe baze de date. Baza de date va contine trei tabele: client, produs si comanda. In aplicatie vor fi implementate operatii pe cele trei tabele (ex: cautare, setare, stergere, vizualizare, inserare) in care sa fie folosita si tehnica de „reflection”. De asemenea aplicatia trebuie sa fie impartita in pachete care sa fie impartite intr-o „layered architecture”. In final vor fi generate documentele JavaDoc pentru fiecare clasa.

1. **Analiza problemei**

Initial se va crea baza de date in MySQL, asupra careia se va implementa aplicatia din tema. Aplicatia va contine interfete grafice pentru „fereastra de start” din care se pot accesa „fereastra pentru clienti”, „fereastra pentru produse” si „fereastra pentru comenzi” din care se vor putea utiliza operatiile pe baza de date, corespunzatoare fiecarui tabel. Aplicatia se va imparti in mai multe pachete care sa reprezinte structura de „layered architecture”. Utilizatorul va crea comenzi care sa contina o persoana si un produs, produsul fiind in stoc intr-o anumita cantitate iar comanda trebuie sa respecte acest aspect. Persoana client va avea un ID, un nume si o adresa de email, produsul va avea un ID, un nume si o cantitate in stoc iar comanda va contine ID-ul ei, ID-ul, numele si adresa de email ale clientului, ID-ul, numele si cantitatea ceruta ale produsului. In final se vor genera documente HTML JavaDoc care sa contina informatii despre fiecare element al aplicatiei.

1. **Implementare**
   1. **Diagrama UML**

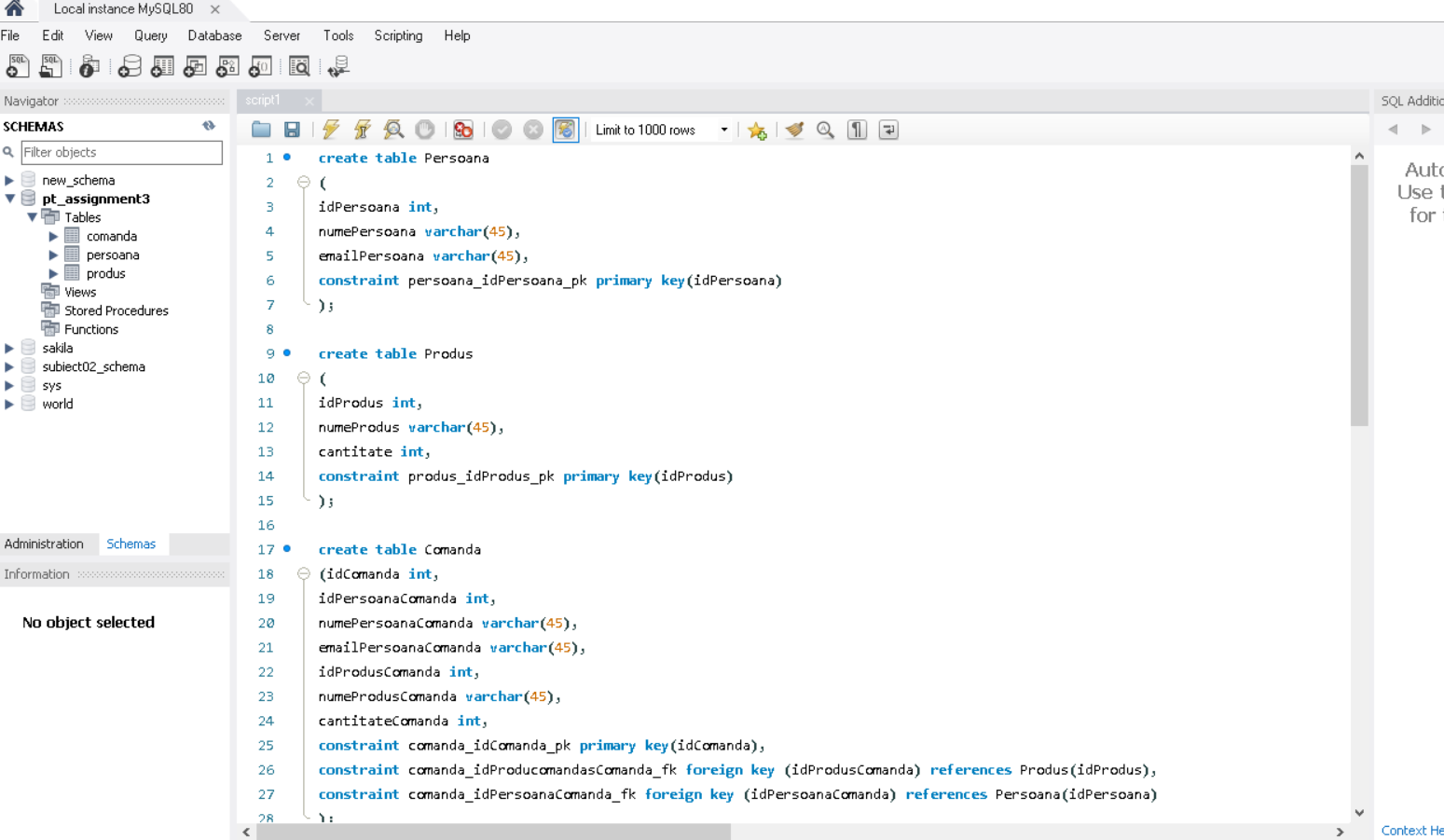
Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele si specificatii pentru software. Pentru reprezentarea acesteia am folosit generatorul implicit de diagrame UML din Intellij.

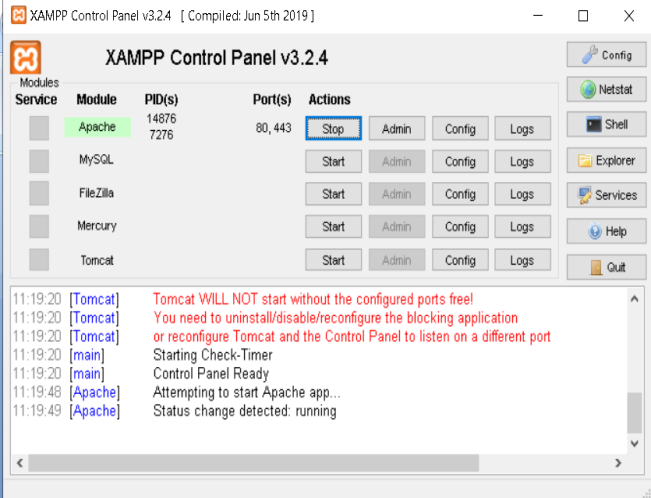


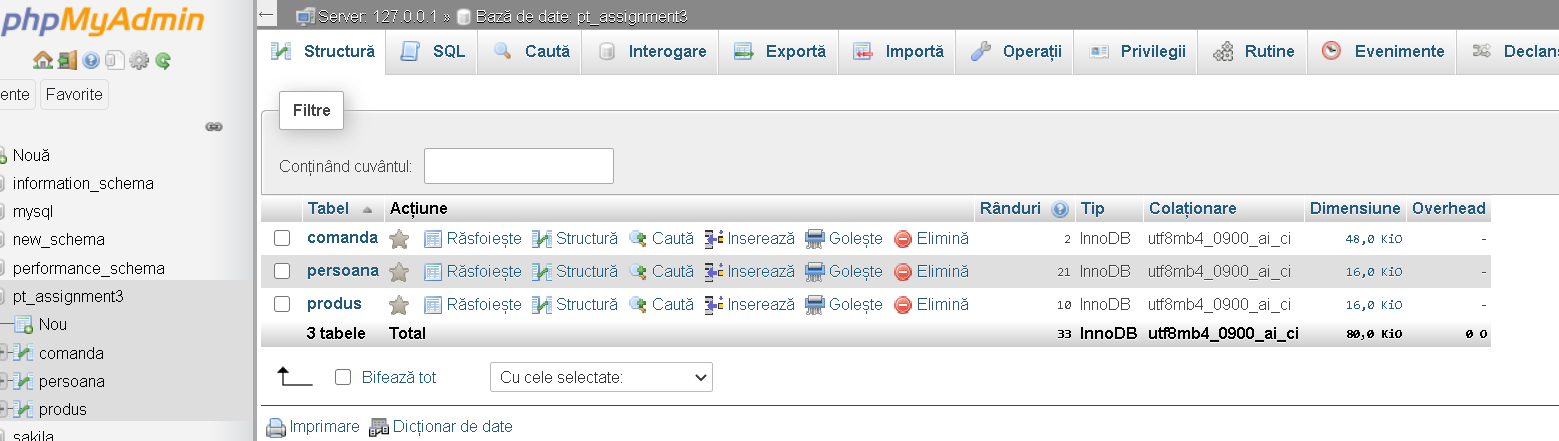
* 1. **Baza de date –SQL-**

Baza de date pe care aplicatia lucreaza este creeata in MySQL Workbench. „Schema” bazei de date se numeste „pt\_assignment3”. Aceasta contine trei tabele: Persoana (care reprezinta clientul), Produs si Comanda. Tabelul Persoana are trei coloane: ID, nume si email, tabelul Produs are tot trei coloane: ID,nume si cantitate in stoc, iar tabelul Comanda prezinta o combinatie intre celelalte doua: ID-uri pentru persoana - produs, nume pentru persoana – produs, adresa de email, cantitate ceruta si ID-ul comenzii care este si cheia primara.

Baza de date va functiona cu ajutorul programului XAMPP, activand Apache. Sunt prezentate mai jos capturi de ecran din MySQL, browser si aplicatia XAMPP:







* 1. **Pachete si clase**

Pachetul „connection” prezinta o singura clasa: „ConnectionSQL”, aceasta face legatura intre aplicatie si baza de date. Contine mai multe variabile instanta: LOGGER de tipul Logger care face legatura la baza de date, String DRIVER care reprezinta driver-ul java pentru baze de date, String-ul DBURL care contine link-ul la baza de date in localhost impreuna cu port-ul sau, String-urile USER si PASS care sugereaza numele de utilizator si parola cu care aplicatia se conecteaza la baza de date in MySQL.

Pachetul „start” prezinta doar clasa „Main”, aceasta deschide aplicatia si afiseaza interfata grafica.

Pachetul „view” contine 4 clase de tip „View”. „StartView” reprezinta prima fereastra a aplicatiei. Din aceasta se pot accesa ferestrele care sa lucreze pe unul dintre cele trei tabele. Clasa „PersoanaView” este clasa care creeaza fereastra grafica pentru operatiile pe tabelul Persoana. Clasa „ProdusView” este clasa care creeaza fereastra grafica pentru operatiile pe tabelul Produs, iar in cele din urma, clasa „ComandaView” este clasa care creeaza interfata grafica pentru operatiile pe tabelul „Comanda”.

Pachetul „model” contine clase pentru fiecare tabel existent in baza de date, acestea fiind trei. Clasa „Persoana” prezinta variabile instanta care sa corespunda cu coloanele tabelului Persoana: intregul „idPersoana”, String-ul „numePersoana” si String-ul „emailPersoana”. Clasa „Produs” contine si ea variabile instanta care sa corespunda coloanelor tabelului Produs: intregul „idProdus”, String-ul „numeProdus” si intregul „cantitate” din stoc. Clasa „Comanda” prezinta si ea de asemenea variabile instanta corespunzatoare coloanelor din tabel: intregii „idComanda”, „idPersoanComanda”, „idProdusComanda” si „cantitateComanda”, String-urile „numePersoanaComanda”, „numeProdusComanda”, „emailPersoanaComanda”.

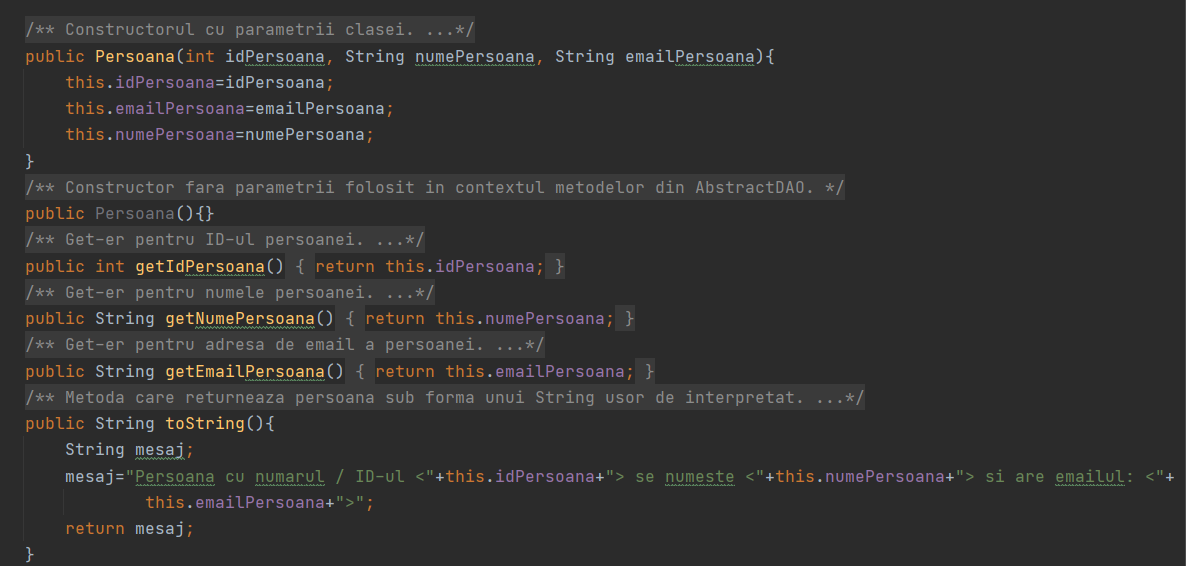
Pachetul „dao” contine mai multe clase. Fiecare contine metode care reprezinta operatii pe baza de date. Clasa „PersoanaDAO” prezinta ca variabile instanta String-uri care sugereaza interogarea pentru baza de date (de exemplu „SELECT \* FROM persoana WHERE emailPersoana = ?” – pentru operatia de cautare in functie de adresa de email a unei persoane). Clasa „ProdusDAO” prezinta si ea variabile instanta de tip String care sugereaza interogari pe baza de date (de exemplu „SELECT \* FROM produs WHERE numeProdus = ?” – pentru operatia de cautare in functie de numele produsului). De asemenea clasa „ComandaDAO” prezinta si ea variabile instanta String care sa reprezinte interogari pe tabelul corespunzator (de exemplu „INSERT INTO comanda VALUES (?, ?, ?,?,?,?,?)” – pentru inserarea unei noi comenzi). Clasa „AbstractDAO” contine variabila instanta care va fi folosita sa gaseasca clasa corespunzatoare pentru metode care folosesc tehnica de „reflection”. Fiecare din clasele prezentate mai sus din acest pachet au si o variabila instanta care reprezinta „logger”-ul pentru conectarea la baza de date.

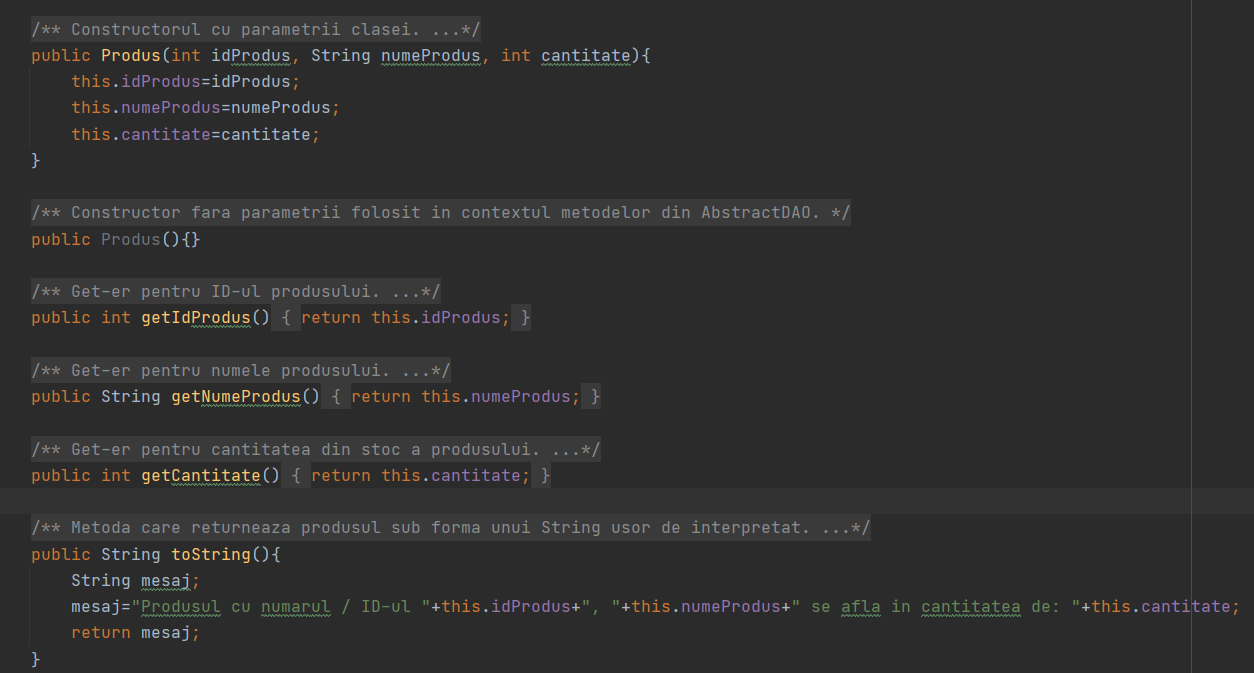
Acest pachet contine cateva clase care mostenesc clasa „AbstractDAO”. Acestea sunt: „PersoanaDAO2”, „ProdusDAO2” si „ComandaDAO2” care folosesc metodele si operatiile clasei parinte, fiind mai eficiente intrucat acestea utilizeaza tehnica de „reflection”. Ultima clasa prezenta in acest pachet este clasa „ReflectionClass” care, de asemenea, cu ajutorul tehnicii de „reflection” prezinta metode care pot afisa intr-un JTable orice obiect sau lista de obiecte pentru oricare tabel din baza de date.

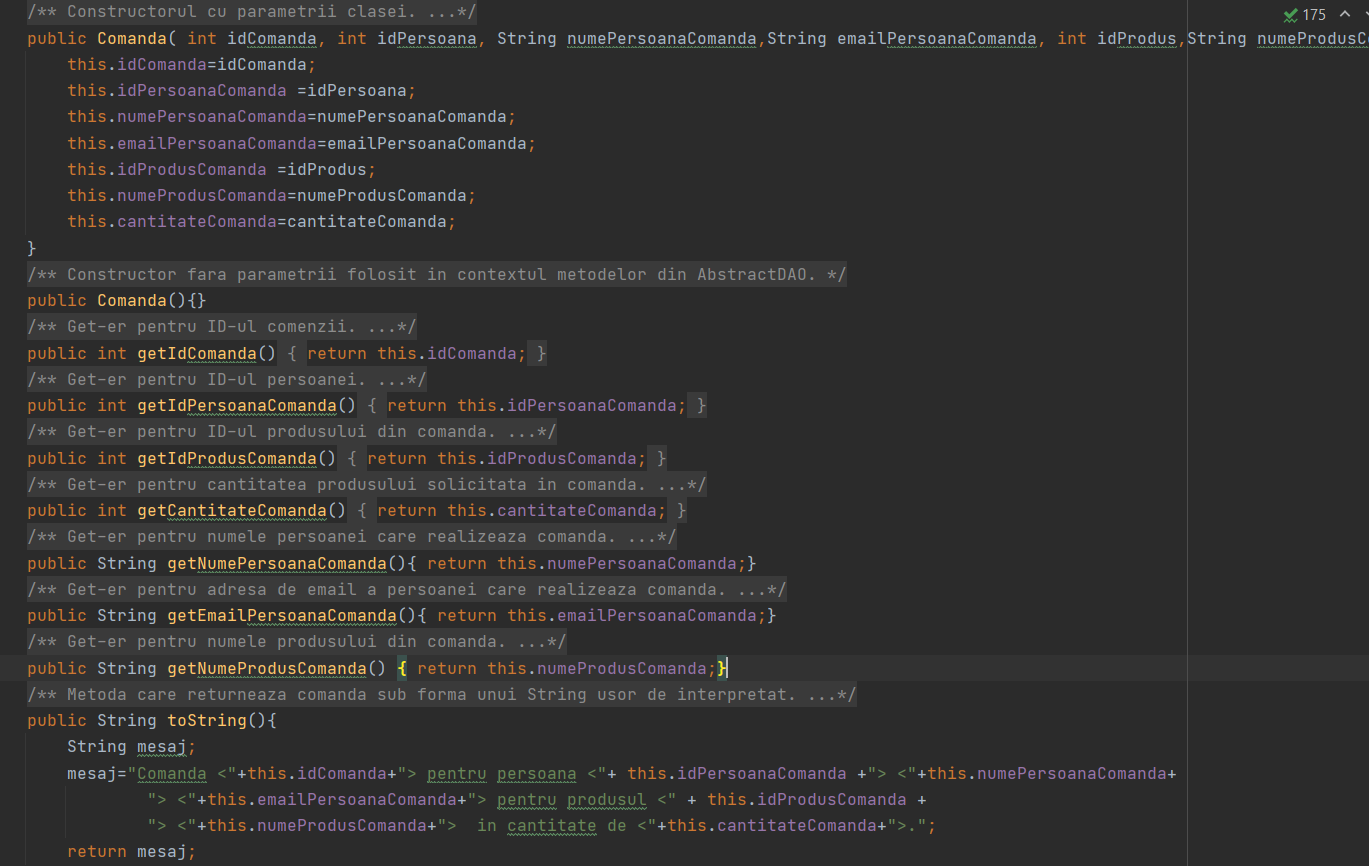
Pachetul „controller” contine clasa „Controller” care reprezinta legatura dintre interfetele grafice „StartView”, „PersoanaView”, „ProdusView” si „ComandaView” si datele cu operatiile pentru tabelele din baza de date. Aceasta clasa prezinta trei variabile instanta, cate una pentru fiecare insterfata grafica corespunzatoare tabelului sau(„PersoanaView”,”ProdusView”, ”ComandaView”). Clasa contine mai multe clase imbricate pentru a realiza actiuni la apasarea butoanelor din ferestrele aplicatiei.

* 1. **Metode si implementare**

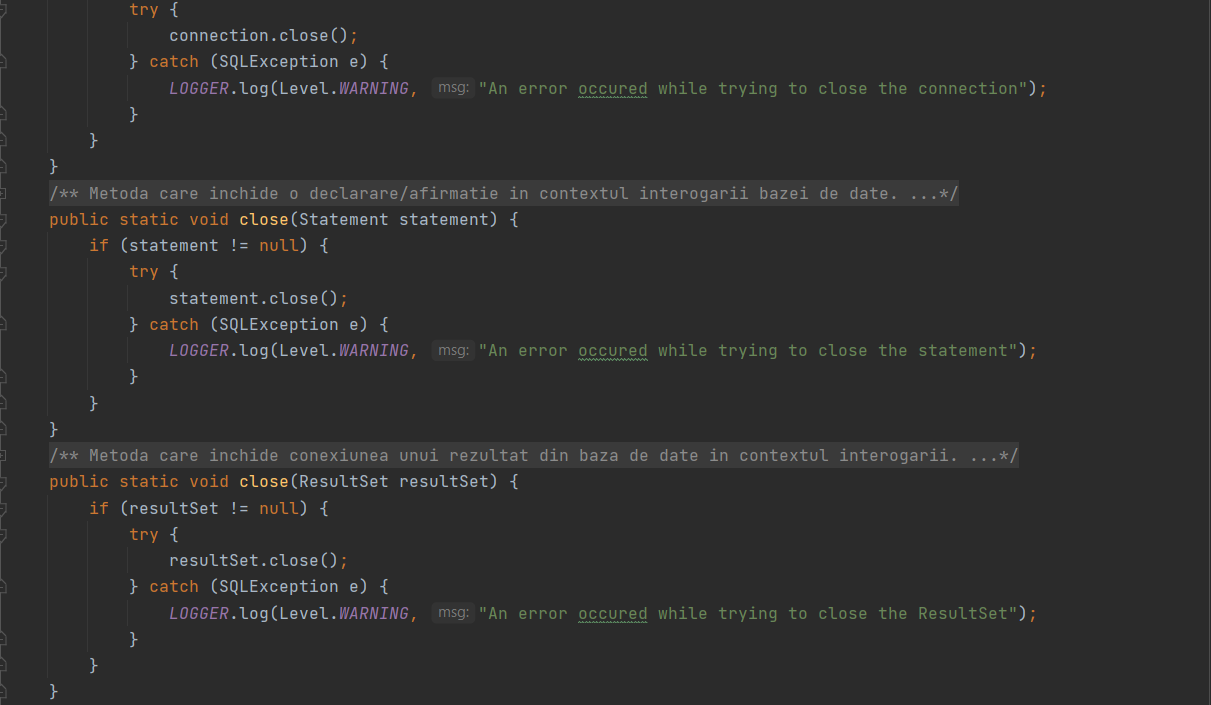
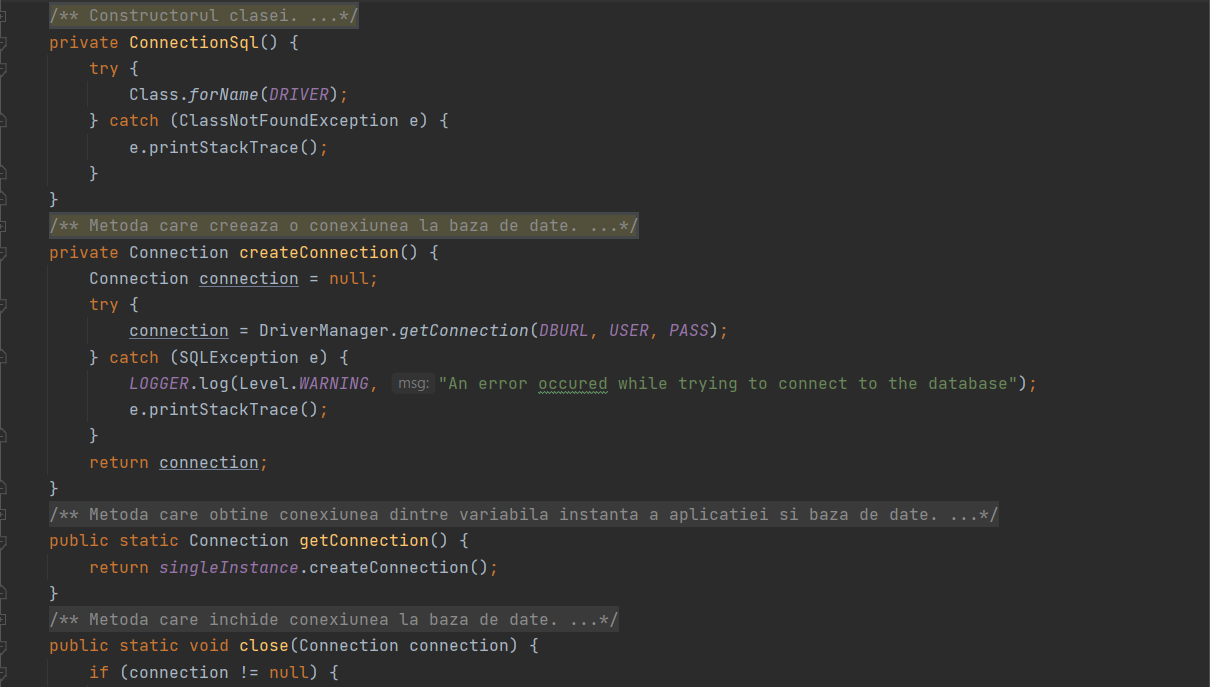
Metodele claselor din pachetul „model”: „Persoana”, „Produs”, „Comanda”. Fiecare din acestea prezinta get-ere pentru variabilele proprii, constructori cu parametrii si fara paramentrii si metoda „toString” care returneaza un String cu fiecare variabila instanta afisata intr-un mod usor de inteles pentru utilizator.





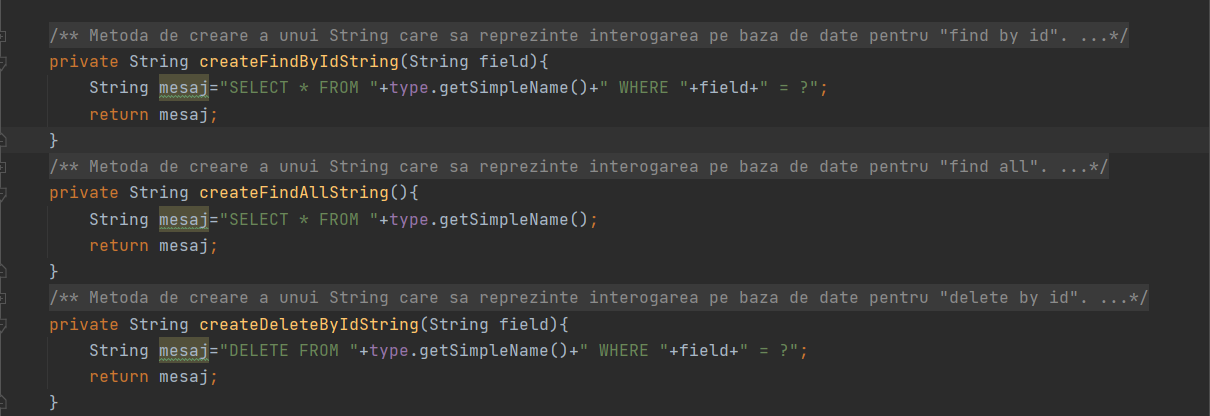


Metodele clasei „ConnectionSQL” din pachetul „connection” sunt reprezentate de: crearea si obtinerea conexiunii la baza de date si inchiderea unei conexiuni, unei interogari si unui rezultat al interogarii.

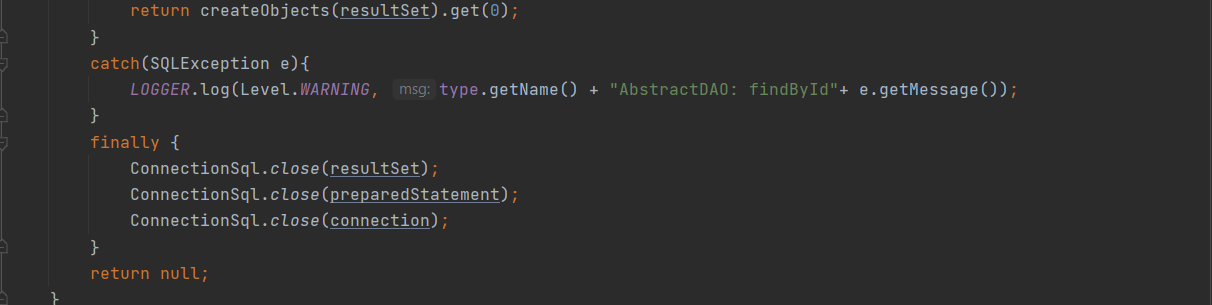
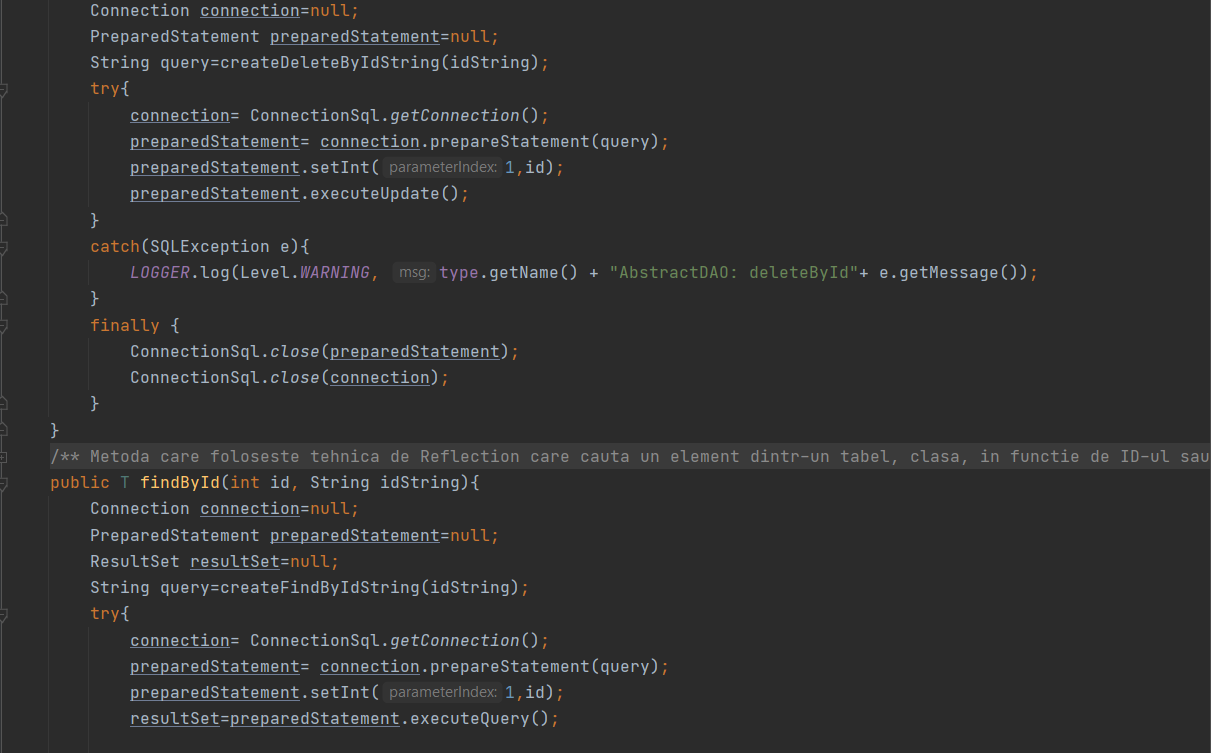


Metodele claselor din pachetul „dao”:

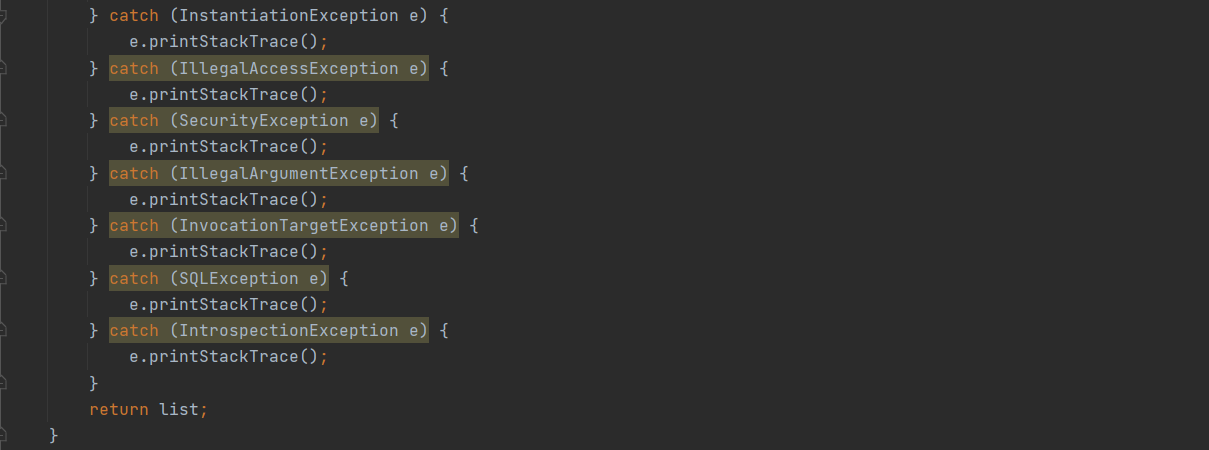
In clasa „AbstractDAO” exista 3 metode de obtinere a unor interogari sub forma unor String-uri pentru a putea fi folosite in operatiile pe tabel:



Sunt prezente de asemenea si metode care realizeaza operatiile de cautare dupa ID („findById”), stergere dupa ID („deleteById”), obtinere a tuturor liniilor dintr-un anumit tabel:



Ultima metoda din aceasta clasa este cea care creeaza o lista de obiecte dintr-un anumit tabel, in functie de rezultatul („ResultSet”) unei interogari.



Fiecare metoda din aceasta clasa este implementata folosind tehnica de „reflection”.

Acestea sunt implementate astfel:

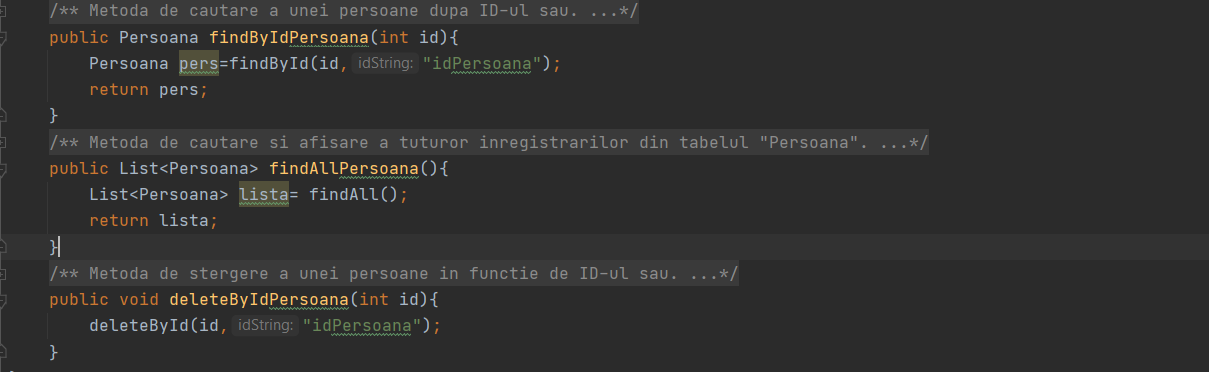
Se initializeaza variabile de tipul Connection (pentru conectarea la baza de date), PreparedStatement (pentru interogare) si String (care este de fapt interogarea, „query”-ul, propriu-zis si este obtinut folosind o metoda de obtinere a String-ului respectiv), cu valoarea „null”. In cazul in care metoda trebuie sa returneze valori, se initializeaza si o variabila de tipul ResultSet.

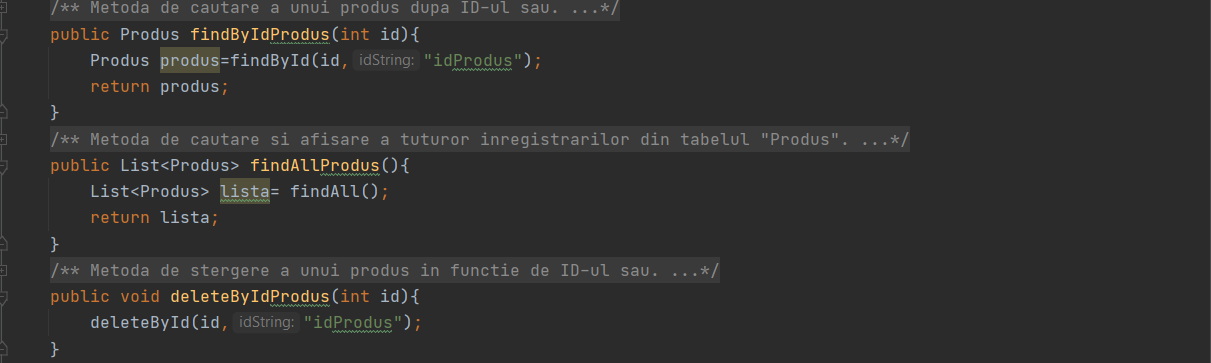
Se incearca apoi conectarea la baza de date, lansarea interogarii si in caz de cautare, obtinerea rezultatelor. Se apeleaza apoi metoda de creare de obiecte („createObjects”) care creeaza o lista de elemente din tabelul cerut. In caz de eroare (de tipul SQLException) se afiseaza un mesaj corespunzator: „AbstractDAO: <metoda>”.

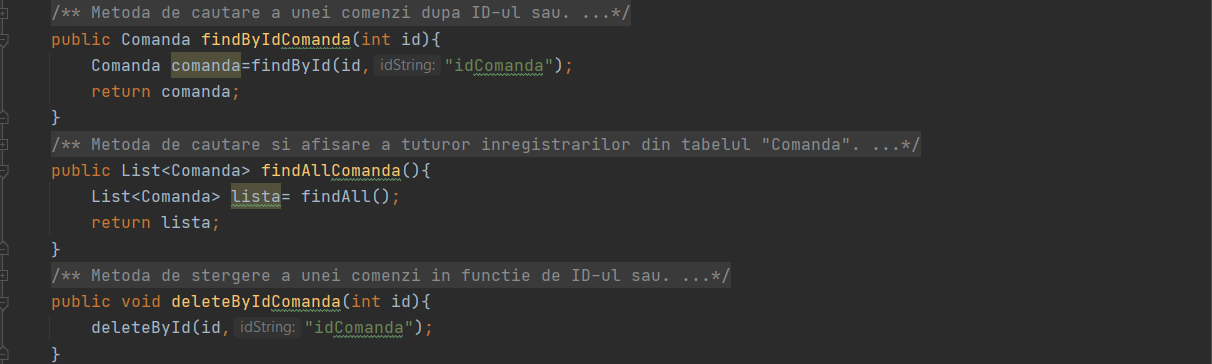
Ultimul pas al metodei este sa inchida conexiunea la rezultatul interogarii, la interogare si la baza de date.

Urmatoarele trei clase: „PersoanaDAO2”, „ProdusDAO2” si „ComandaDAO2” sunt asemanatoare, intrucat toate mostenesc clasa „AbstractDAO”, folosindu-i metodele pentru operatiile specifice pe tabel: cautare dupa ID, stergere dupa ID si obtinerea tuturor liniilor ca lista („List<T>”).

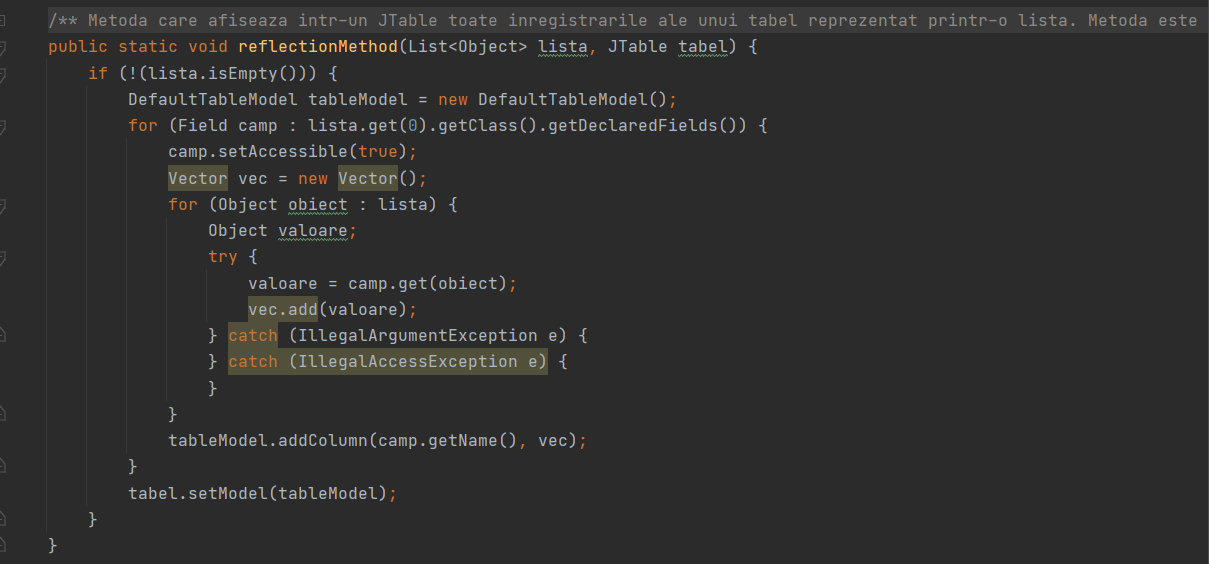
PersoanaDAO2 :



ProdusDAO2:

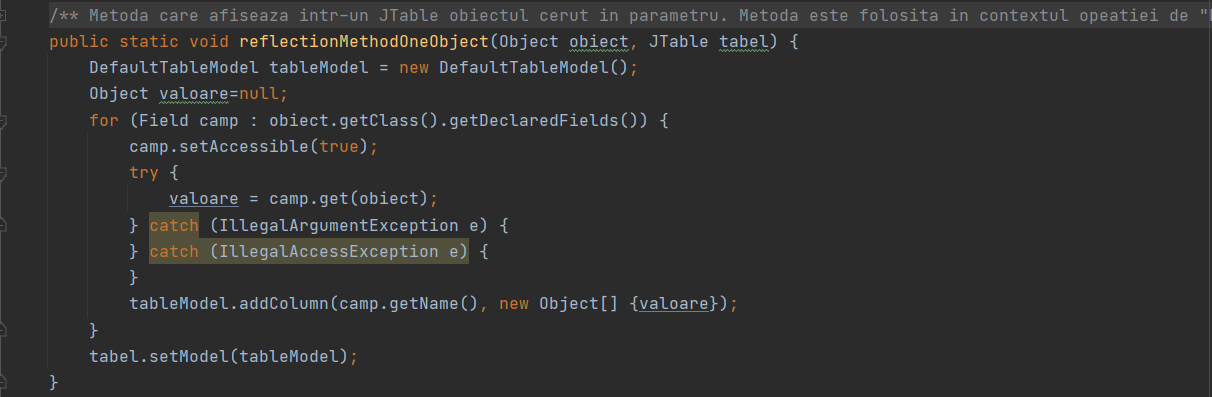
ComandaDAO2:

Urmatoarea clasa este de asemenea una care foloseste in metodele sale tehnica de „reflection”, intrucat pot fi apelate pe oricare tabel din baza de date. Clasa „ReflectionClass” prezinta doua metode: una pentru afisarea intr-un tabel („JTable”) a unei linii cautate in tabel, si una pentru afisarea intr-un tabel a tuturor liniilor care exista in tabelul corespunzator parametrului din metoda.



Aceasta prima metoda primeste doi parametrii: o lista de obiecte de o anumita clasa, ADICA toate liniile care exista intr-un tabel (reprezentat de clasele din pachetul „model”) din baza de date, si un tabel in care sa poata afisa lista de obiecte.

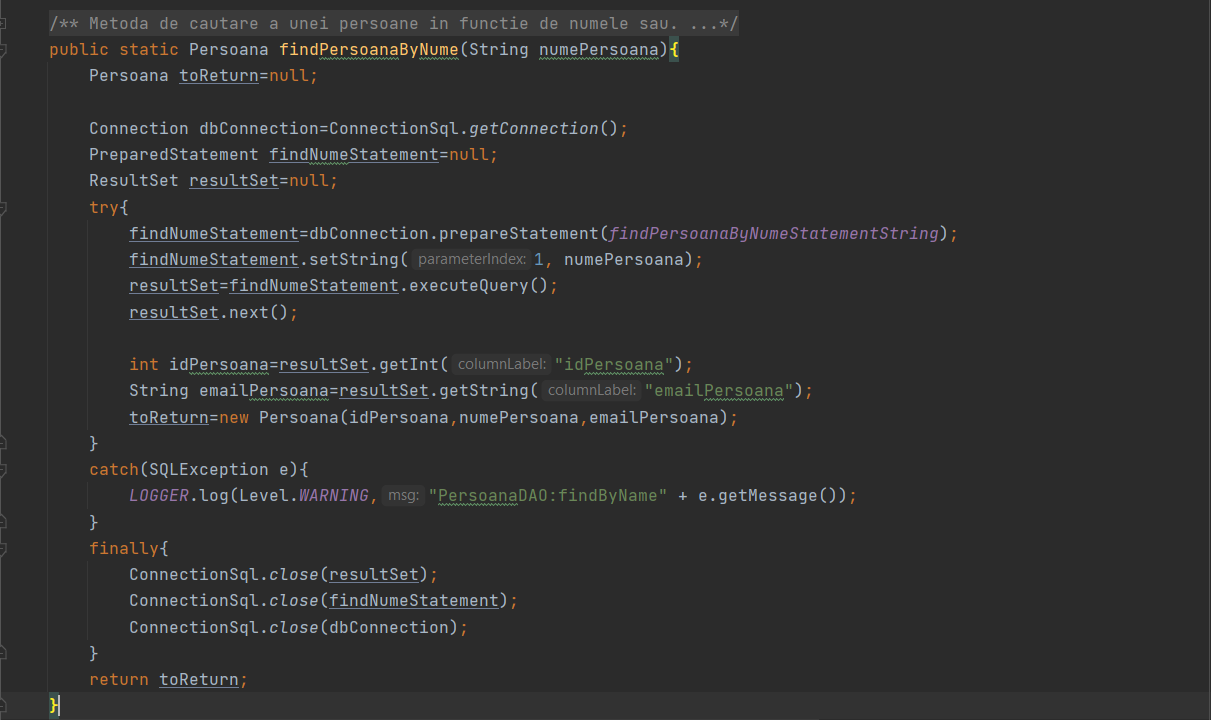
Metoda verifica initial daca lista este goala, in caz afirmativ scrie coloanele tabelului dar nu completeaza nicio linie, in caz contrar creeaza un model de tabel „DefaultTableModel” si apoi acceseaza fiecare camp (coloana a tabelului) a caror valori le introduce intr-un vector si le adauga la „tableModel” ca fiind o coloana. In cazul in care operatiile din „try” esueaza, erorile sunt prezente in clauza „catch” pentru a nu afisa nimic, astfel tabelul ramanand necompletat.



Aceasta metoda este asemenatoare celei precedente, insa primul parametru este doar un obiect (din oricare tabel) ca va fi afisat pe o singura linie in tabelul dat ca al doilea parametru. Sunt urmati aceeasi pasi ca in metoda pentru lista, insa campurile (coloanele tabelului) sunt parcurse doar pentru unicul obiect dar ca parametru. In cele din urma este adaugat in „tableModel” si afisat pe prima linie a tabelului.

Clasa „PersoanaDAO” prezinta operatii tabelul „Persoana” din baza de date (inserare, cautare dupa nume, email, stergere dupa nume, email si modificare a valorilor de ID,nume sau email al unor persoane) reprezentate prin metode, fara a se folosi tehnica de „reflection”.

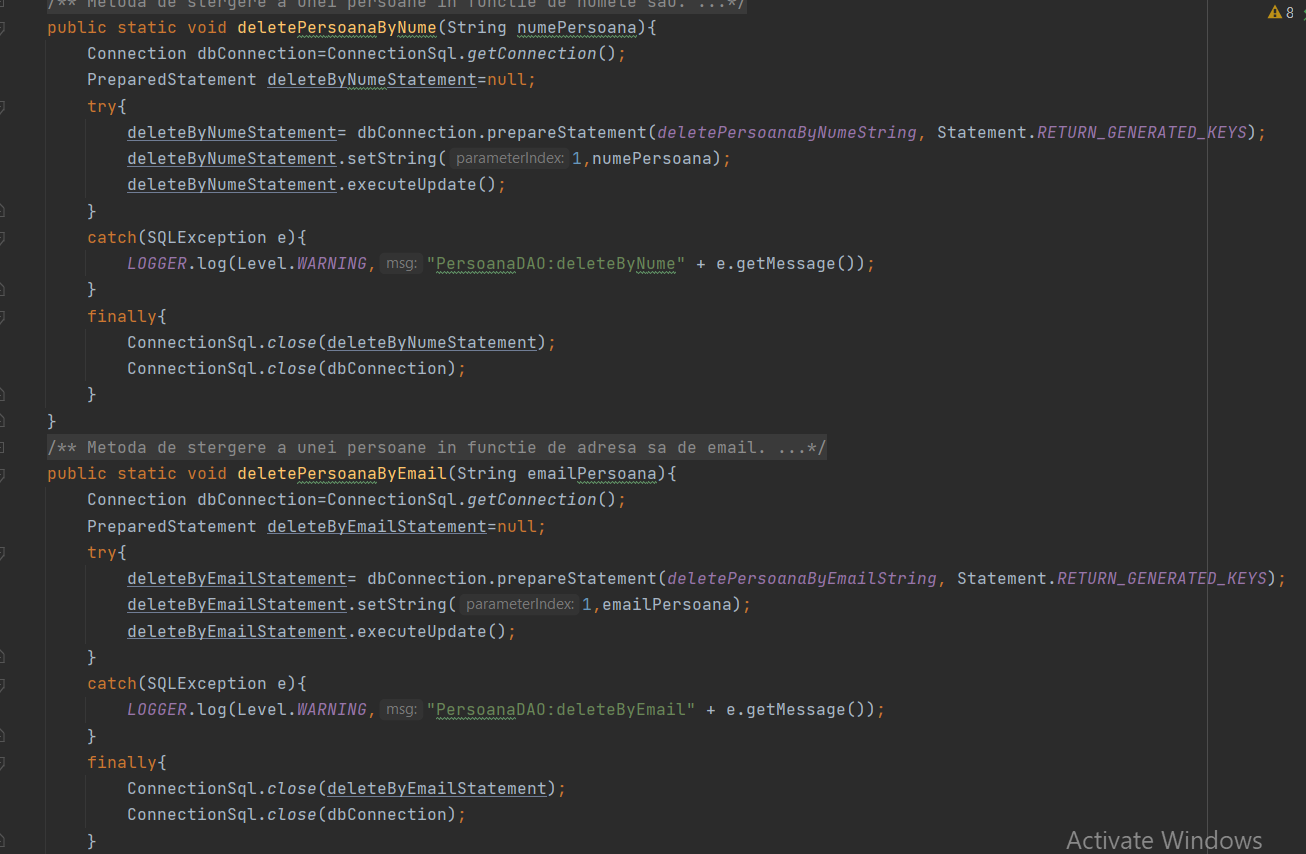
Cautare dupa numele sau adresa de email a clientului:



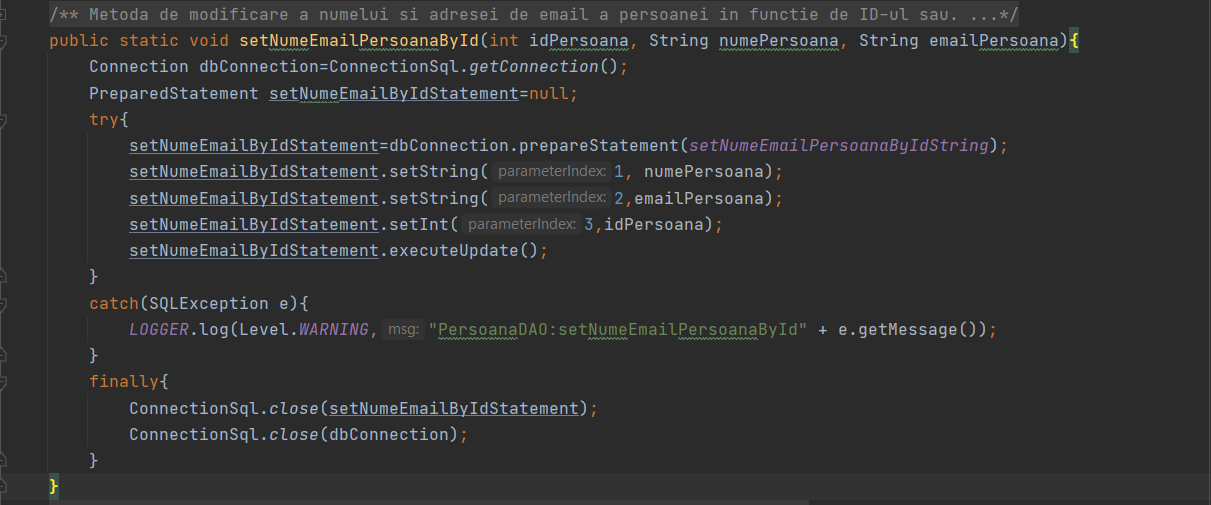
Acestea functioneaza exact in acelasi fel cu cautarea din clasa „AbstractDAO”, insa campurile rezultatelor (coloanele tabelului) trebuie specificate in functie de tabelul in care se lucreaza. Se creeaza in final o noua persoana („new Persoana()”) cu valorile campurilor specificate si este returnata.

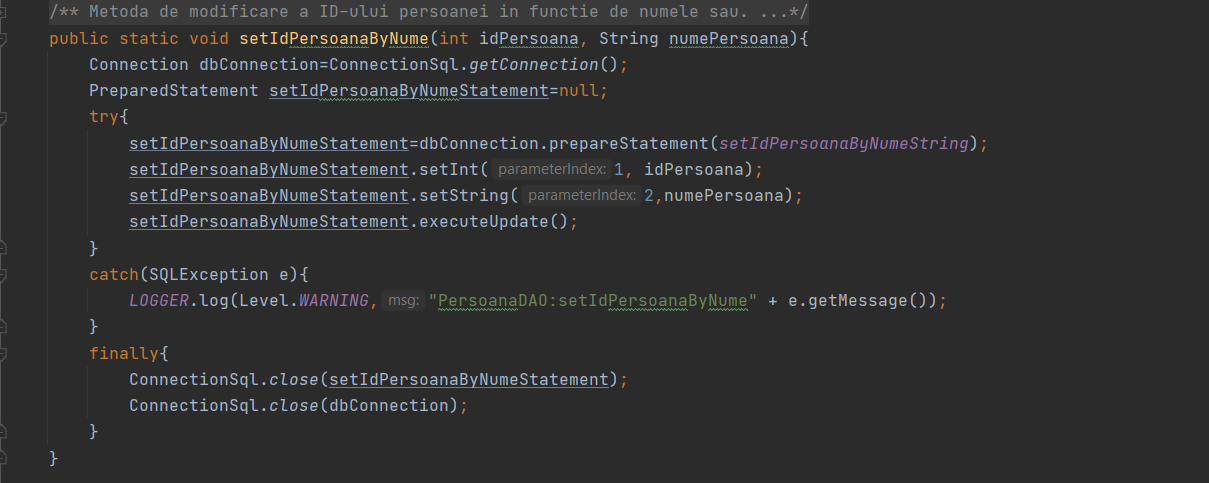
Celelalte metode sunt foarte asemanatoare ca structura, codurile acestora sunt prezentate in continuare:

Stergere dupa nume sau dupa adresa de email a clientului:



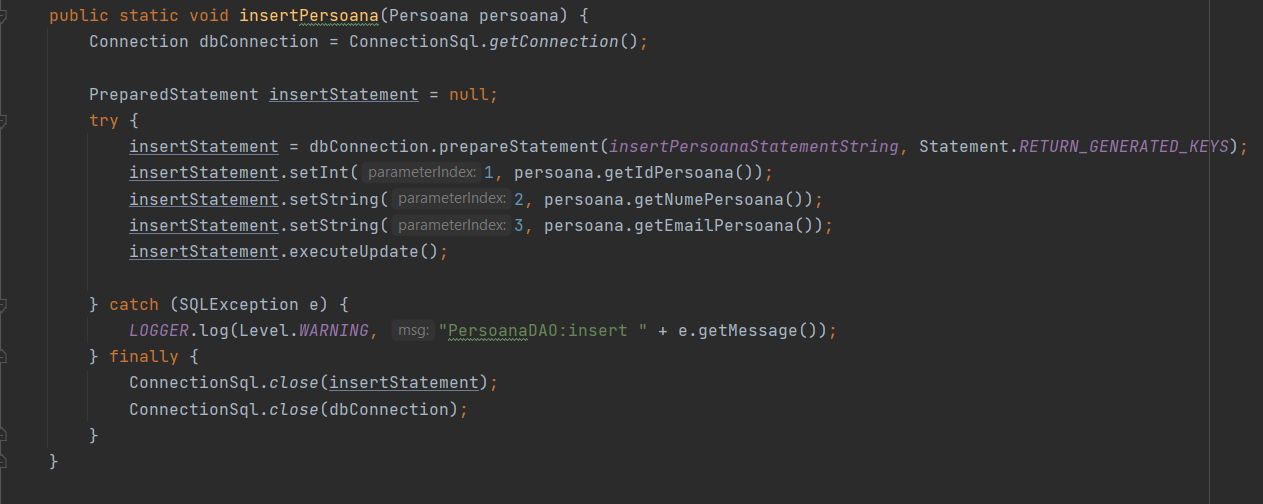
Setare a campurilor cu anumite valori in functie de ID-ul sau numele persoanei:





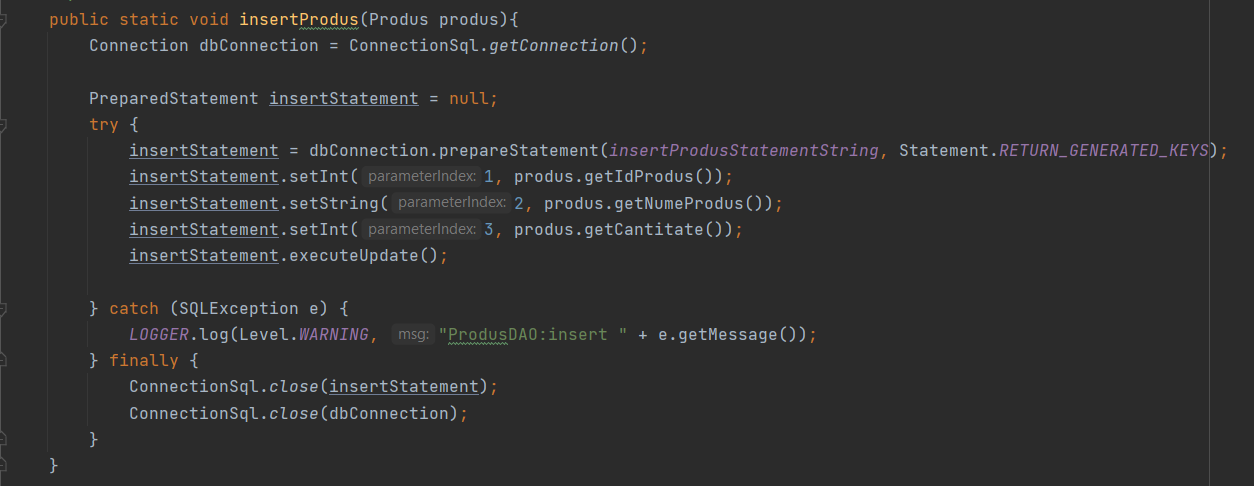
Prima metoda modifica numele si adresa de email a unui client in functie de ID-ul acestuia, pe care a doua metoda modifica ID-ul unui client in functie de numele sau.

Adaugare:

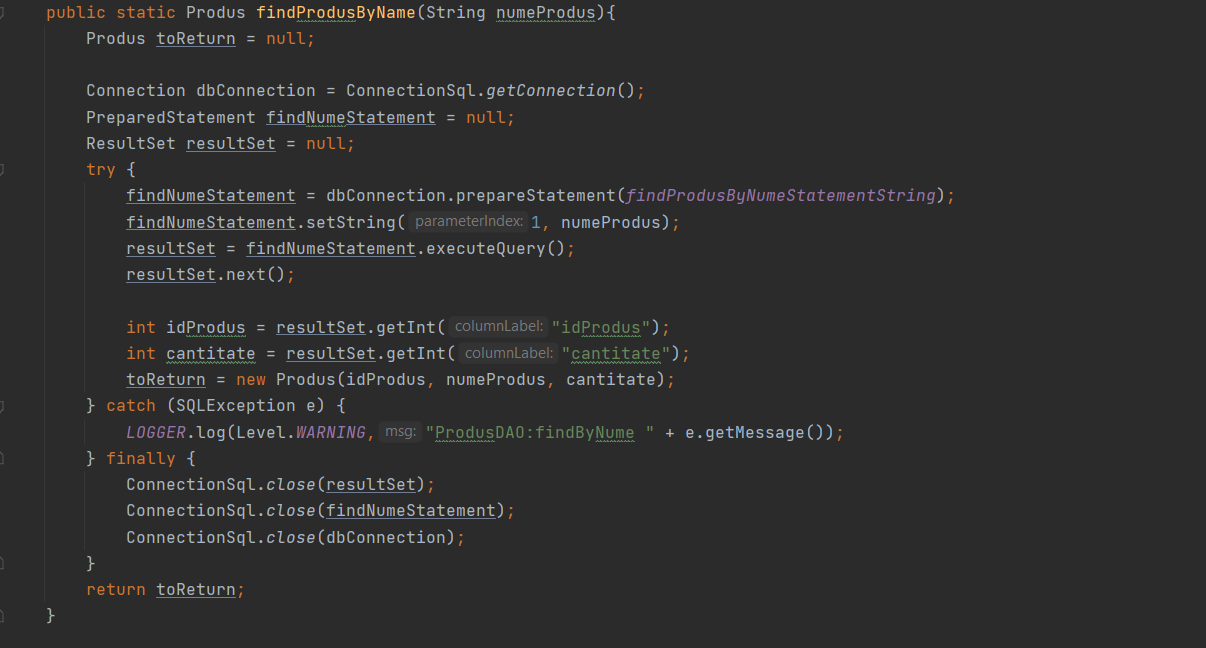


Clasa „ProdusDAO” contine operatii pe tabelul „Produs” din baza de date (adaugare, cautare dupa nume, stergere dupa nume, setare a ID-ului, numelui si cantitatii produsului) reprezentate prin metode, fara a se folosi tehnica de „reflection”.

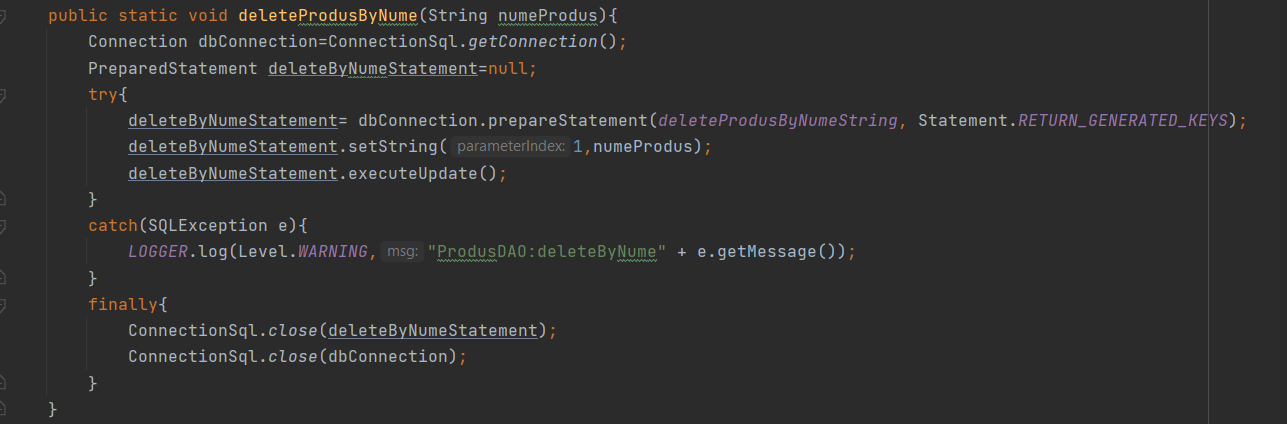
Adaugare:



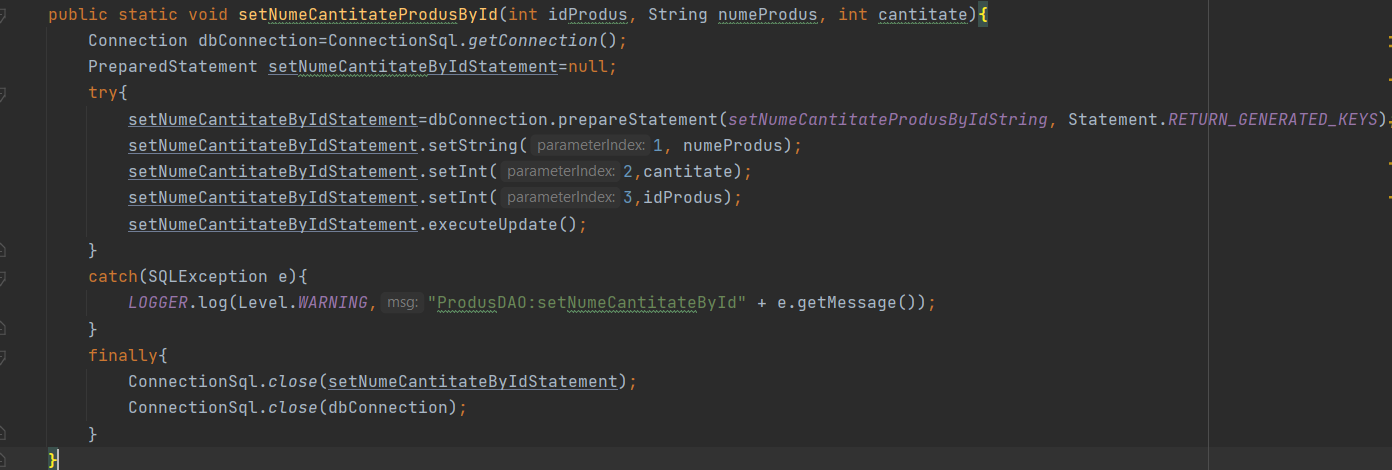
Cautare dupa numele produsului:

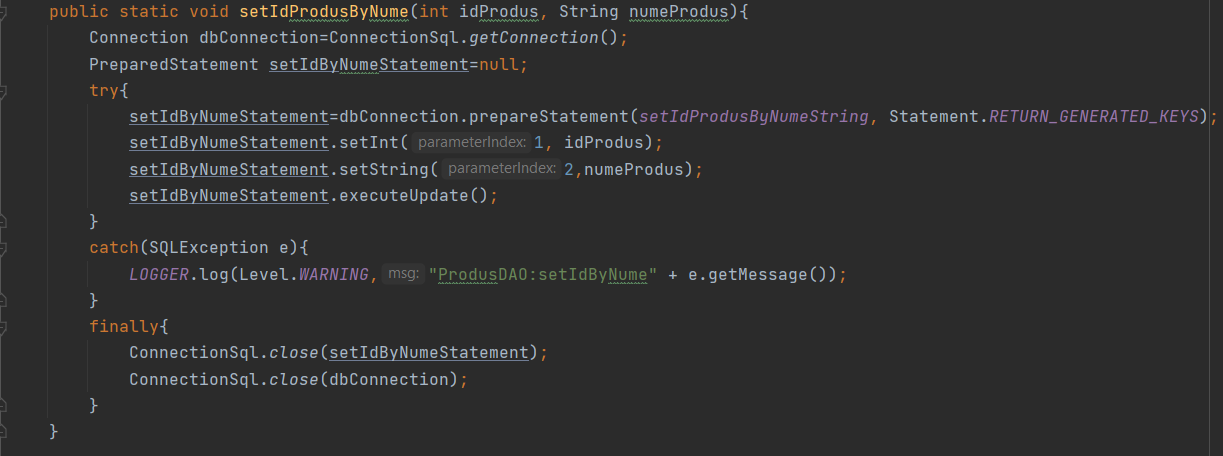


Stergere dupa numele produsului:

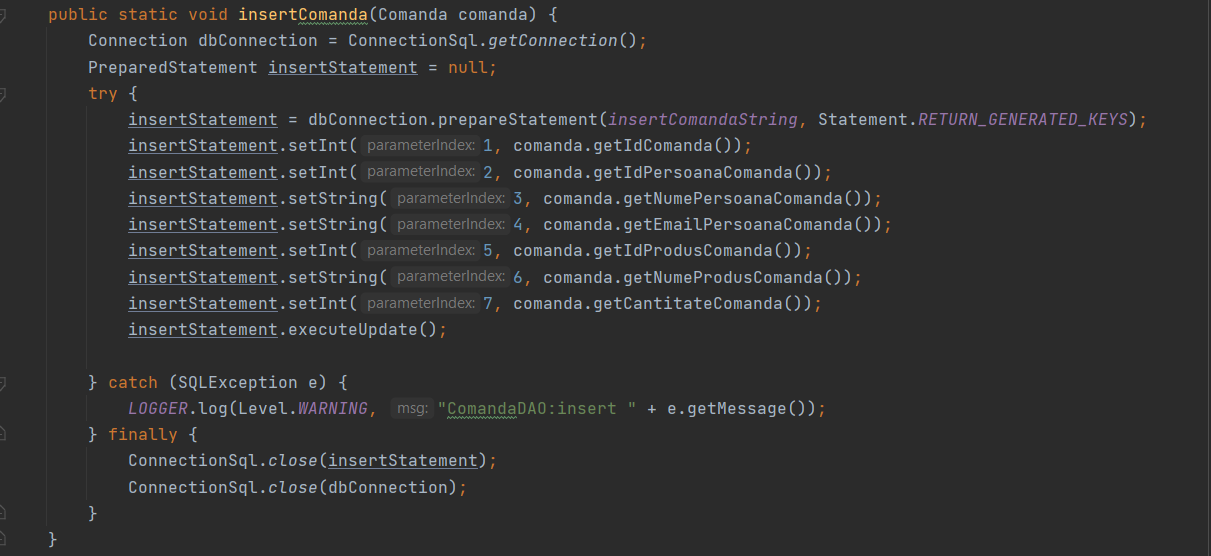


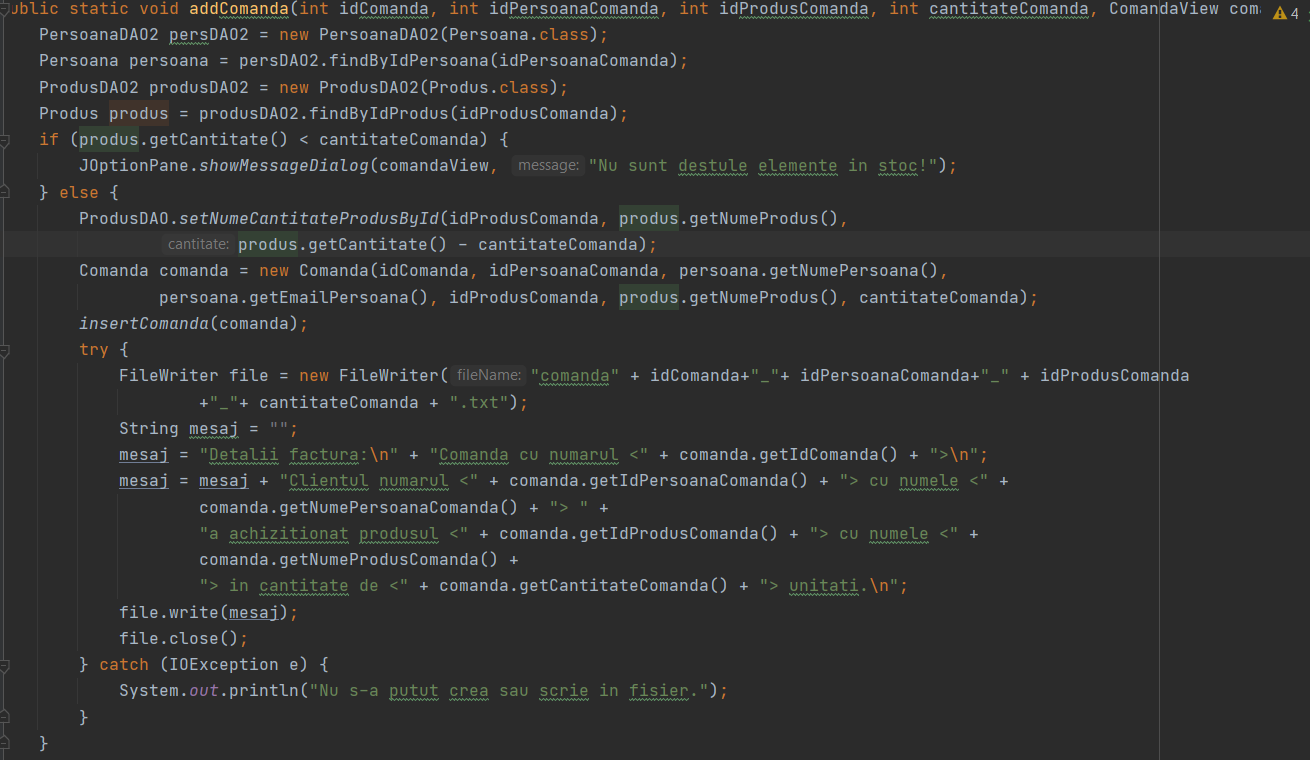
Setare a campurilor cu anumite valori in functie de ID-ul sau numele produsului:





Clasa „ComandaDAO” contine operatia de inserare/adaugare pe tabelul „Comanda” din baza de reprezentate prin metode, fara a se folosi tehnica de „reflection”.



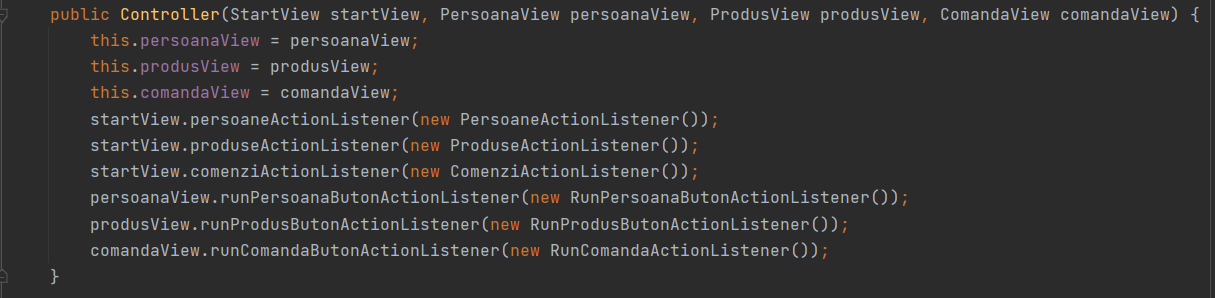


Creearea unei comenzi si adaugarea sa in baza de date se face astfel:

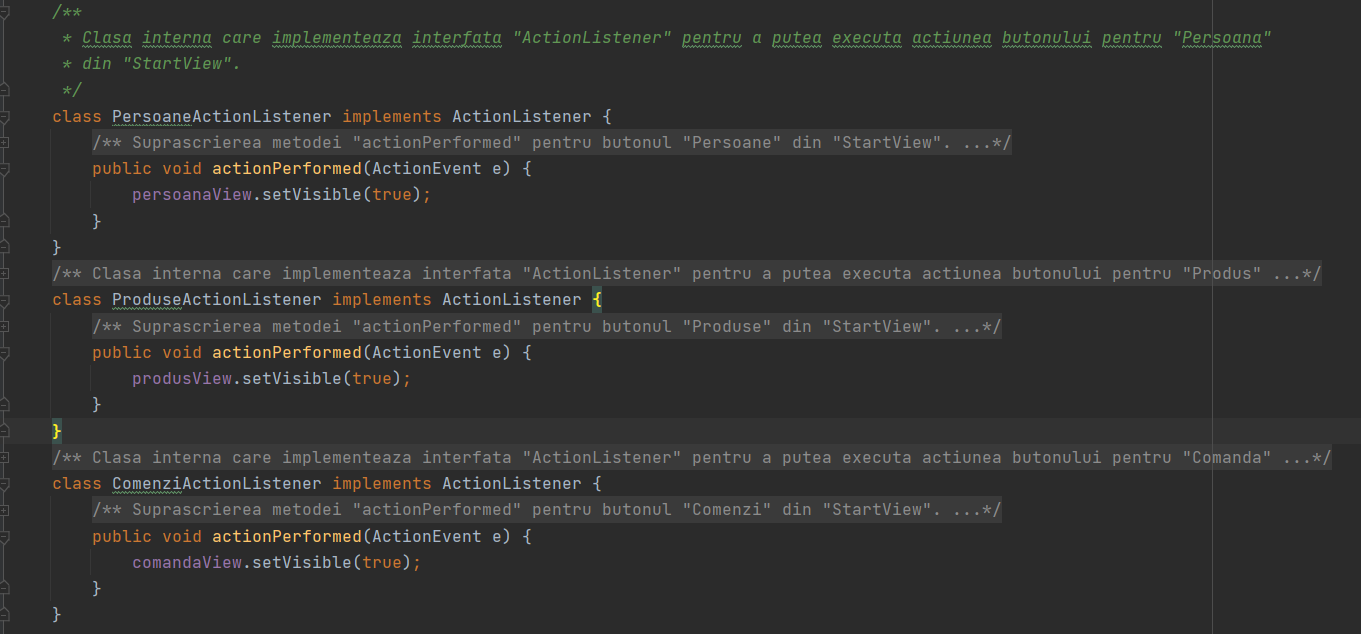
Intrucat pentru a procesa o comanda este nevoie de o persoana client si un produs intr-o anumita cantitate, primul pas este cautarea clientului si apoi a produsului specificati in parametrii in baza de date. In cazul in care cantitatea produsului din comanda este mai mare decat cantitatea din stoc a produsului, se afiseaza un mesaj corespunzator: „Nu sunt destule elemente in stoc!”. In caz contrar se scade din cantitatea din stoc a produsului, cantitatea necesara pentru comanda. Se creeaza apoi un obiect Comanda care primeste in argumentele constructorului detale furnizate in parametrii metodei. Se apeleaza apoi metoda de „insertComanda” care adauga in baza de date comanda procesata si valida. Un alt obiectiv al acestei metode este crearea unei facturi: adica scrierea intr-un fisier corespunzator, datele comenzii: ID-urile necesare, numele necesare, adresa clientului si cantitatea produsului. In cazul in care acest procedeu esueaza, se afiseaza un mesaj corespunzator: „Nu s-a putut crea sau scrie in fisier.” ,a dica factura nu poate fi generata.

Clasa „Controller” implementeaza fiecare actiune a butoanelor din interfata grafica.

In constructuor in initializeaza interfetele grafice pentru „PersoanaView”, „ProdusView”, „ComandaView” si fiecare variabila care sugereaza actiunea butonului din interfata corespunzatoare.



Se creeaza clase imbricate clasei Controller care, la apasarea butoanelor din „StartView” sa afiseze ferestrele corespunzatoare fiecarui tabel din baza de date.



Se implementeaza apoi clase care sa efectueze operatia corespunzatoare butonului „RUN” din fiecare fereastra.

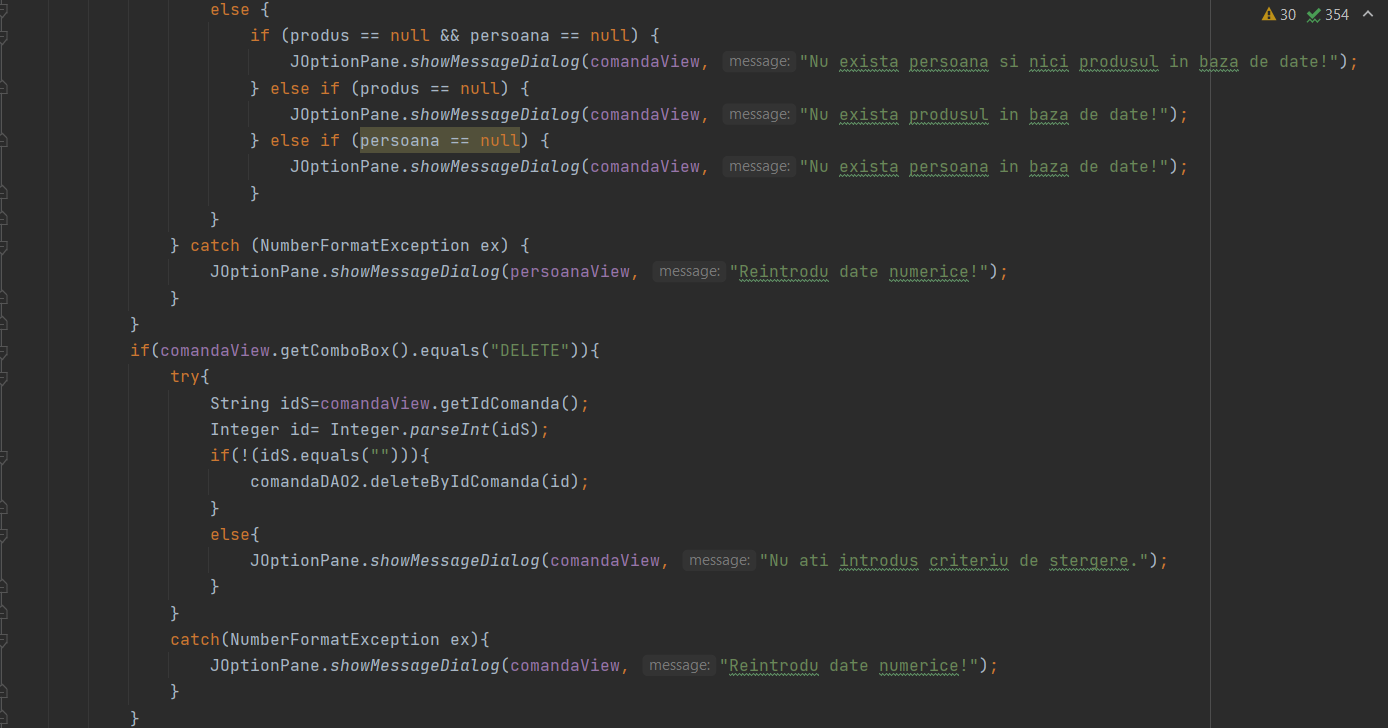
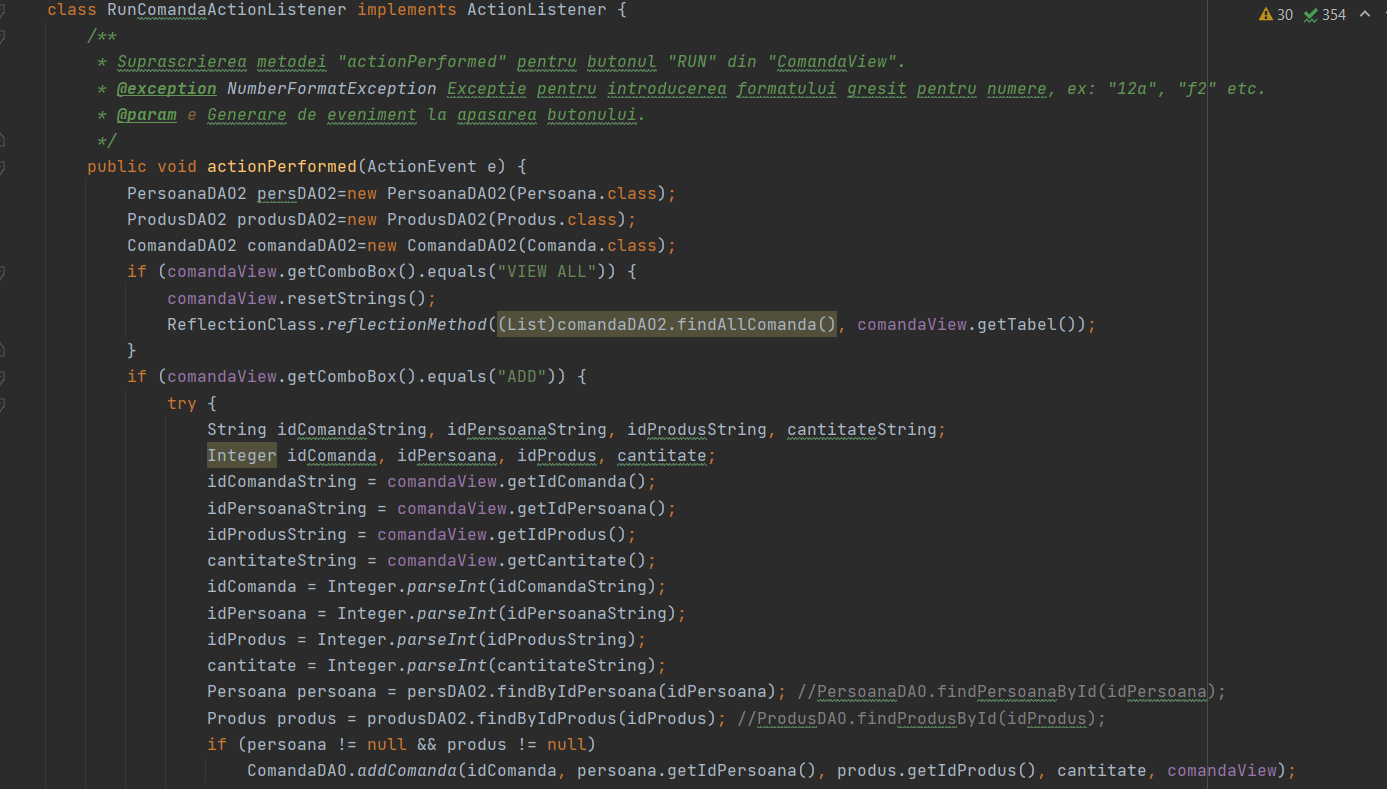
In interfata grafica se va selecta din lista ComboBox operatia care sa fie efectuata la apasarea butonului „RUN”. In functie de operatia selectata, se vor apela metodele necesare pentru tabelul ales.

Explicatiile codului (pentru tabelul „Comanda”):

* pentru alegerea „VIEW ALL” se vor afisa in tabelul ferestrei toate liniile din tabelul „Comanda” din baza de date
* pentru alegerea „ADD” se va adauga in baza de date un nou obiect de tipul „Comanda”
* pentru alegerea „FIND” se va efectua cautarea in tabel a liniei corespunzatoare datelor obiectului
* pentru alegerea „DELETE” se va sterge din tabelul bazei de date linia corespunzatoare datelor obiectului

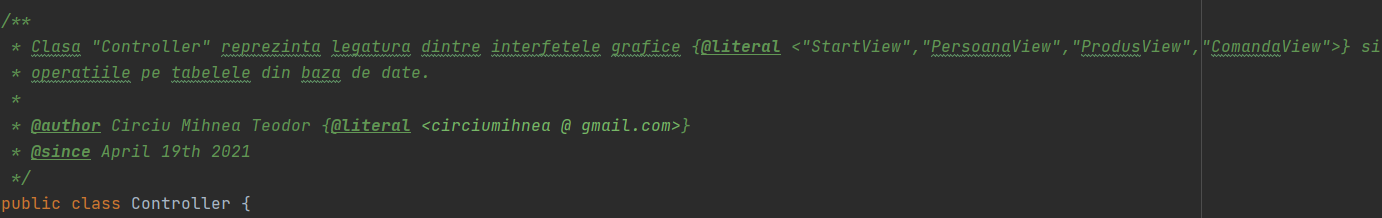
Pentru celelalte tabele (Persoana si Produs), implementarea este asemanatoare.

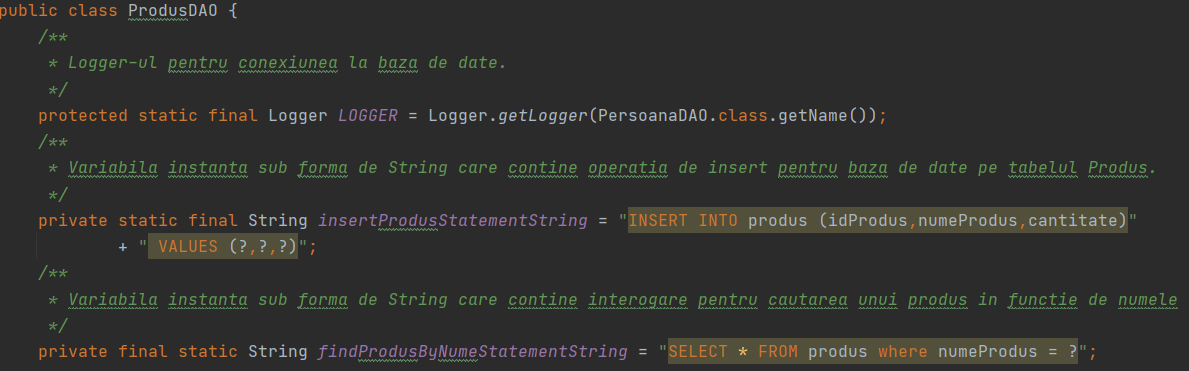
Codul pentru tabelul „Comanda” se afla in pagina urmatoare.

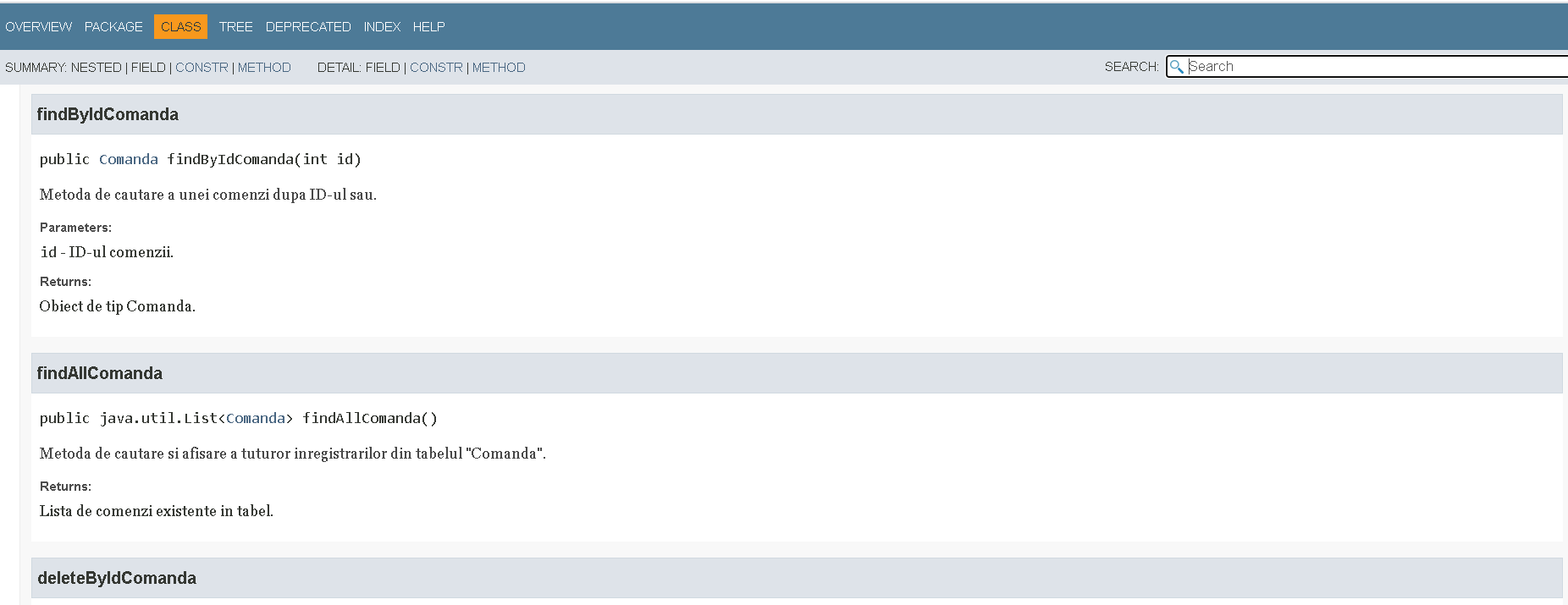


* 1. **JavaDoc**

Un alt obiectiv al acestei teme este generarea documentelor JavaDoc ale claselor aplicatiei.





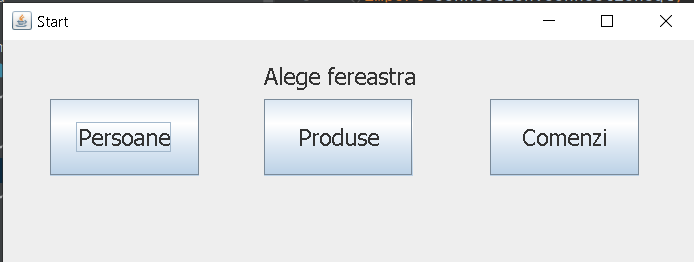




* 1. **GUI**

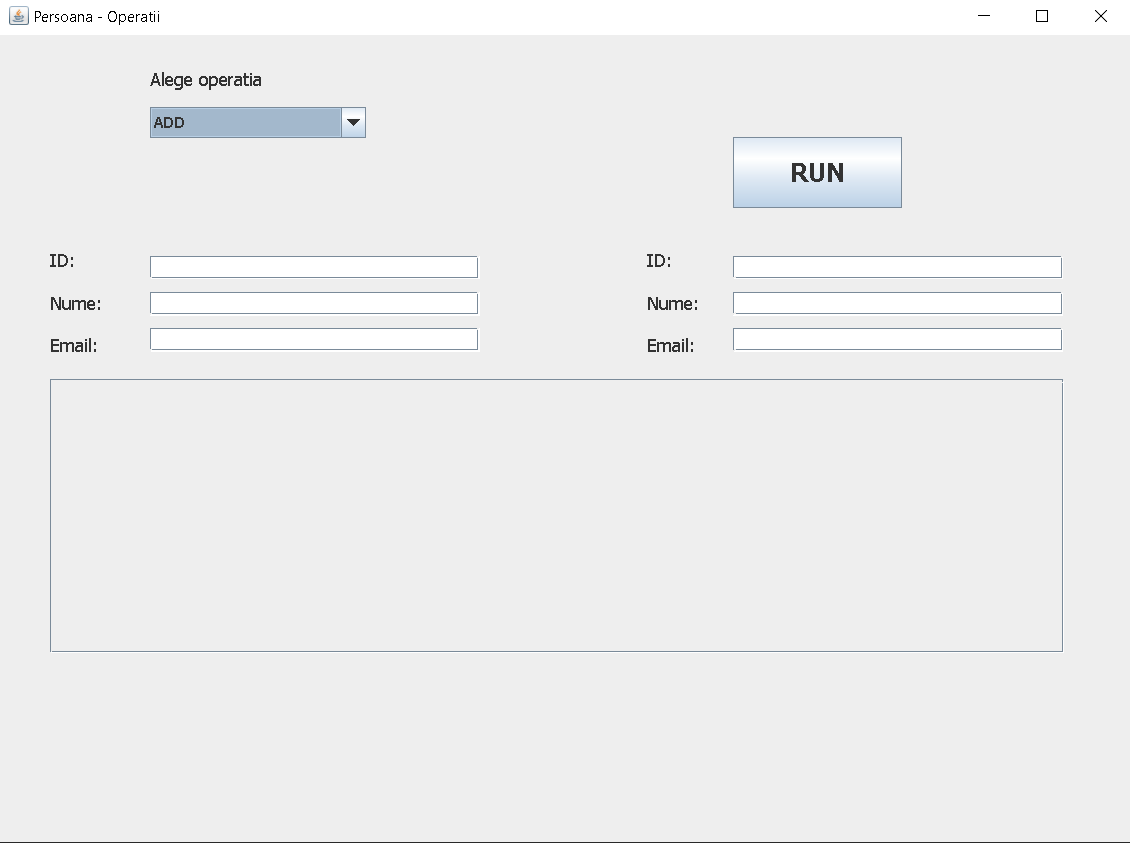
Interfata grafica a aplicatiei reprezinta conexiunea dintre utilizator si program. Aceasta afiseaza elemente grafice pentru ca utilizatorul sa foloseasca programul mai usor.

Fereastra principala este creata cu ajutorul clasei „StartView”



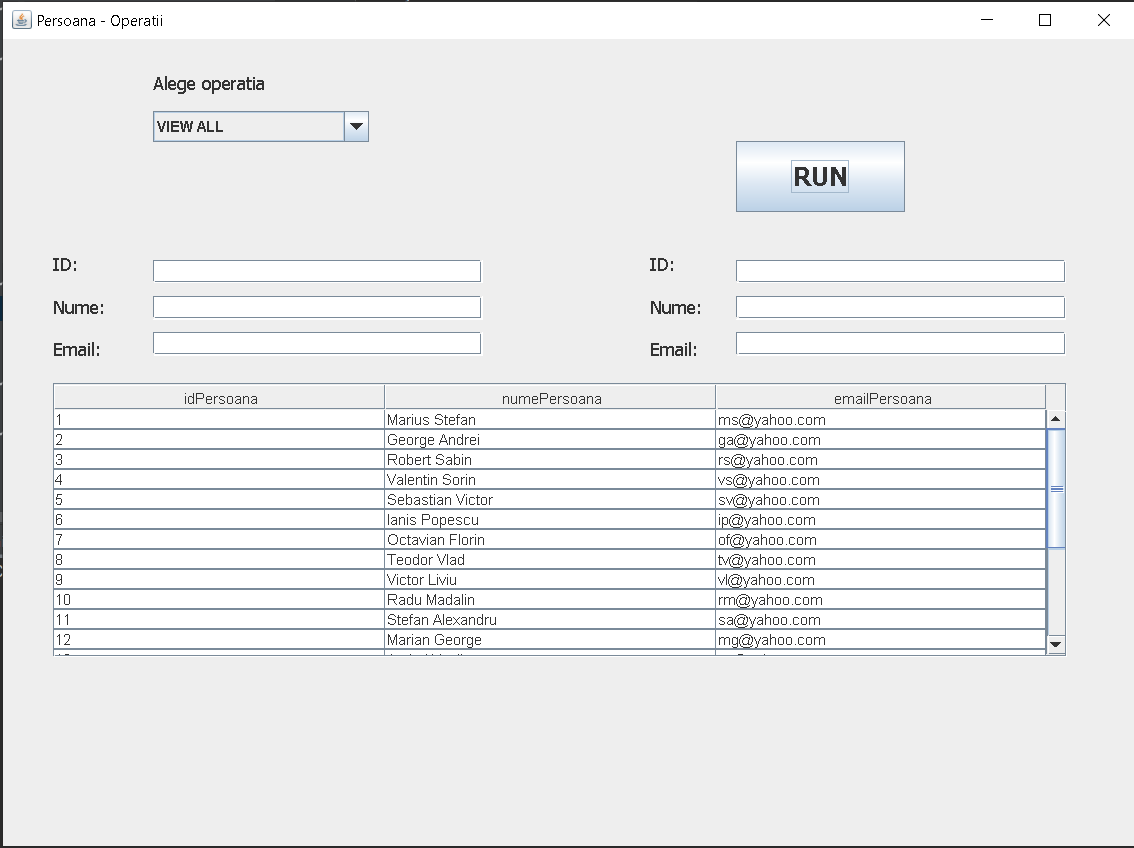
Aceasta contine o eticheta („JLabel”) in care se cere utilzatorului sa aleaga fereastra corespunzatoarea operatiilor pe un anumit tabel. La apasarea unui buton se va deschide fereastra specifica acestuia.

Fereastra corespunzatoare tabelului „Persoana” este generata cu ajutorul clasei „PersoanaView”:

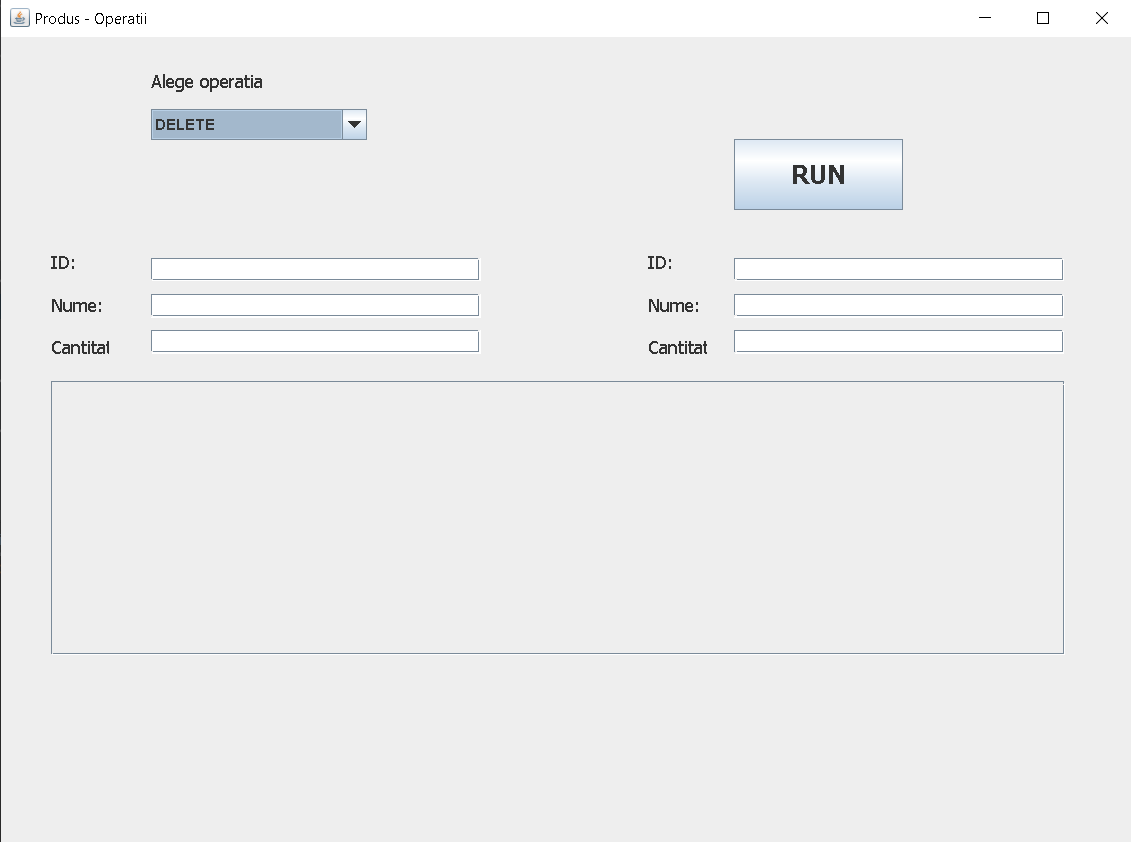


Interfata grafica contine spatiu in care sa vor afisa informatiile cautate din tabelul Persoana, o lista JComboBox din care se poate alege operatia care sa fie efectuata pe tabel, butonul de „RUN” care ruleaza operatia selectata, si , in cele din urma zona de specificatii din stanga si cea din dreapta. In zona din stanga se pot specifica valorile unui obiect care poate fi adaugat, cautat sau sters. Partea din dreapta este utila in contextul in care unei anumite linii i se modifica valorile campurilor.

Pe langa aceasta interfata grafica, clasa „PersoanaView” prezinta metode de curatare a casetelor text si de obtinere a valorilor din casetele text.

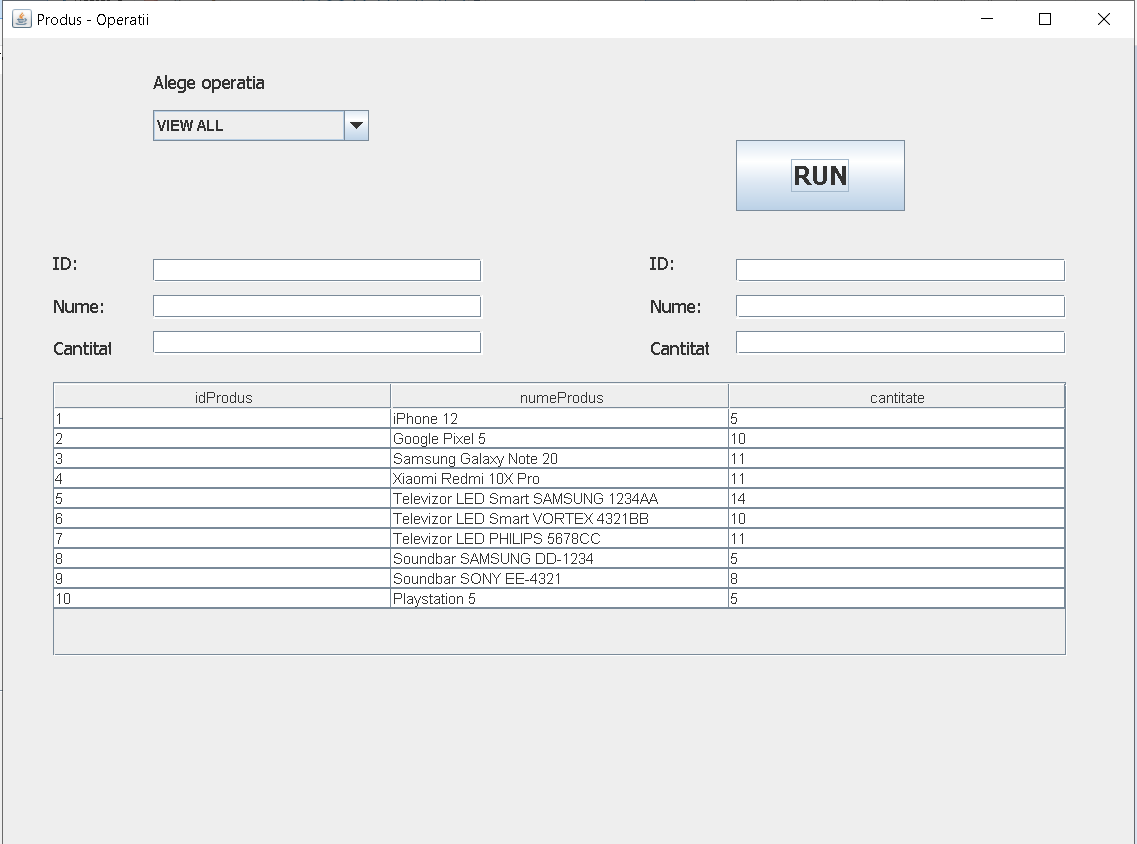


Fereastra corespunzatoare tabelului „Produs” este generata cu ajutorul clasei „ProdusView”:

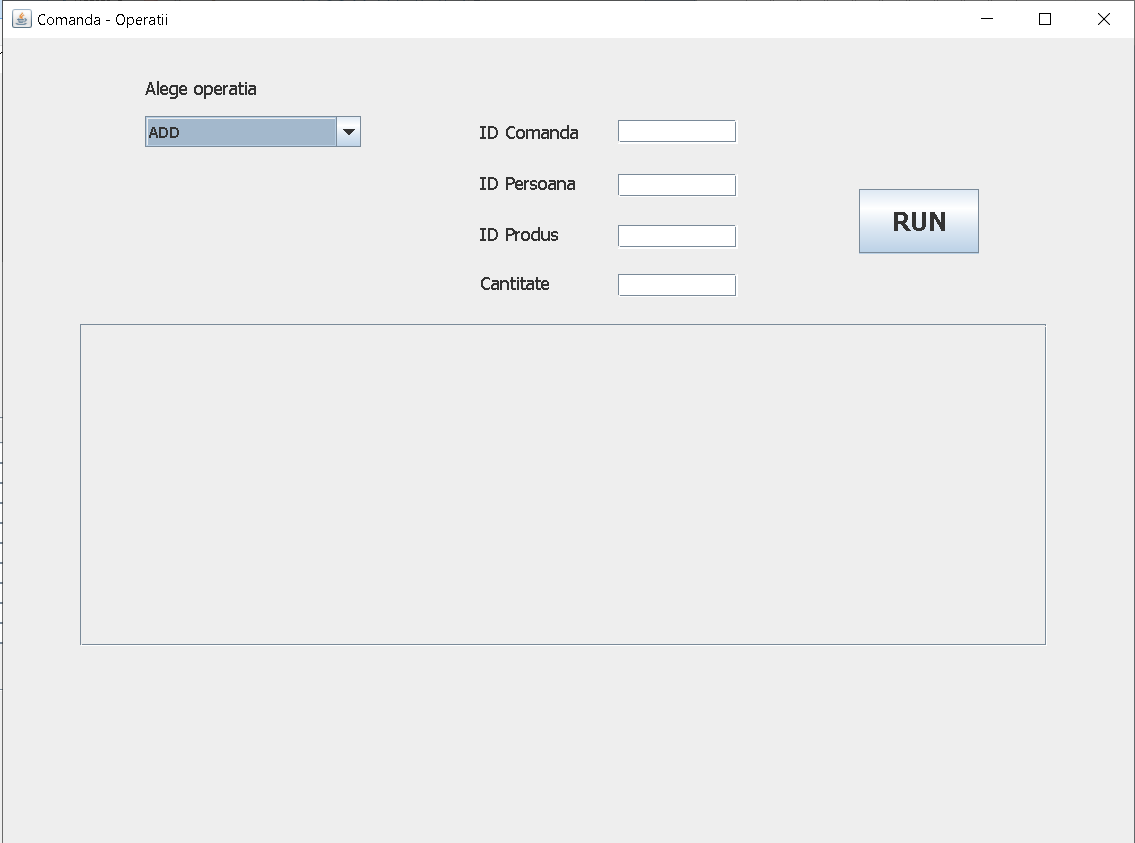


Interfata grafica contine spatiu in care sa vor afisa informatiile cautate din tabelul Produs, o lista JComboBox din care se poate alege operatia care sa fie efectuata pe tabel, butonul de „RUN” care ruleaza operatia selectata, si , in cele din urma zona de specificatii din stanga si cea din dreapta. In zona din stanga se pot specifica valorile unui obiect care poate fi adaugat, cautat sau sters. Partea din dreapta este utila in contextul in care unei anumite linii i se modifica valorile campurilor.

Pe langa aceasta interfata grafica, clasa „ProdusView” prezinta metode de curatare a casetelor text si de obtinere a valorilor din casetele text.

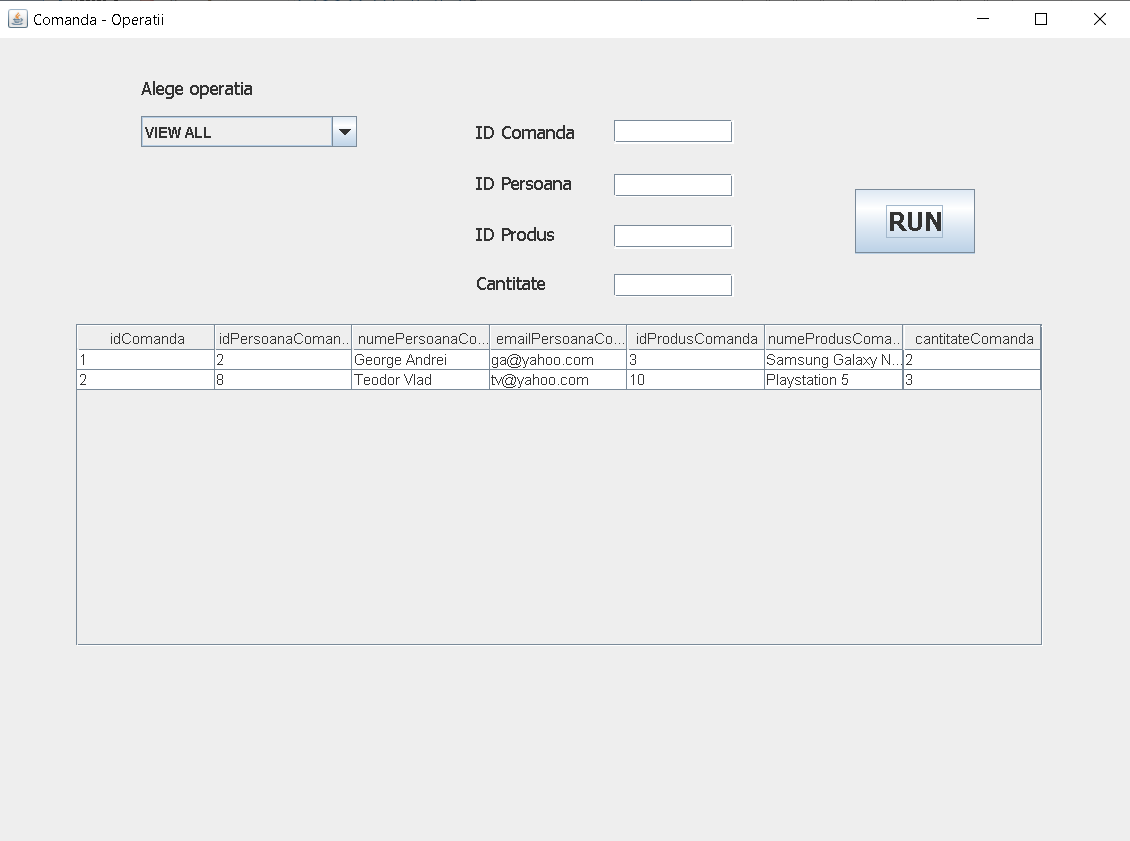


Fereastra corespunzatoare tabelului „Produs” este generata cu ajutorul clasei „ComandaView”:



Interfata grafica contine spatiu in care sa vor afisa informatiile cautate din tabelul Produs, o lista JComboBox din care se poate alege operatia care sa fie efectuata pe tabel, butonul de „RUN” care ruleaza operatia selectata, si , in cele din urma zona de specificatii. In zona de specificatii se pot introduce valorile unui obiect care poate fi adaugat, cautat sau sters.

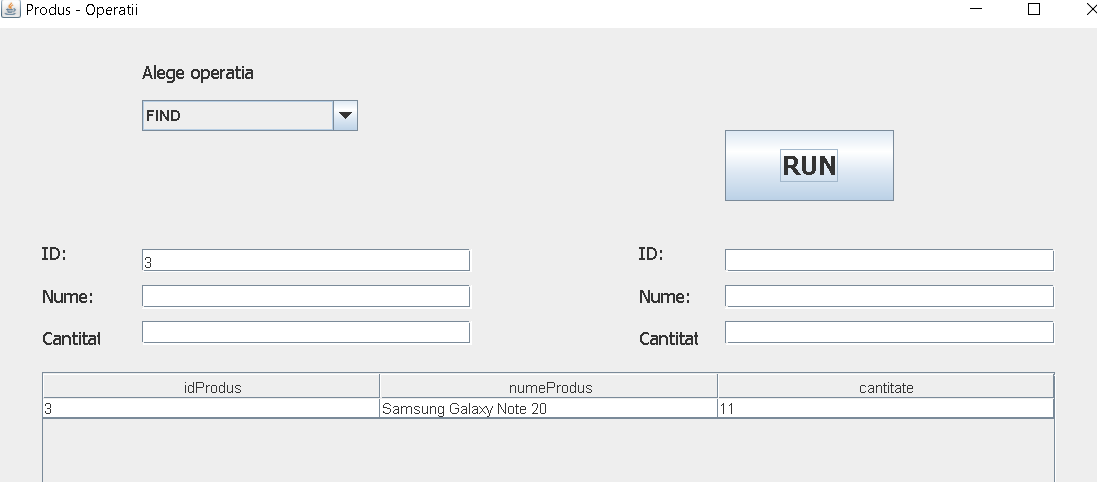
Pe langa aceasta interfata grafica, clasa „ComandaView” prezinta metode de curatare a casetelor text si de obtinere a valorilor din casetele text.



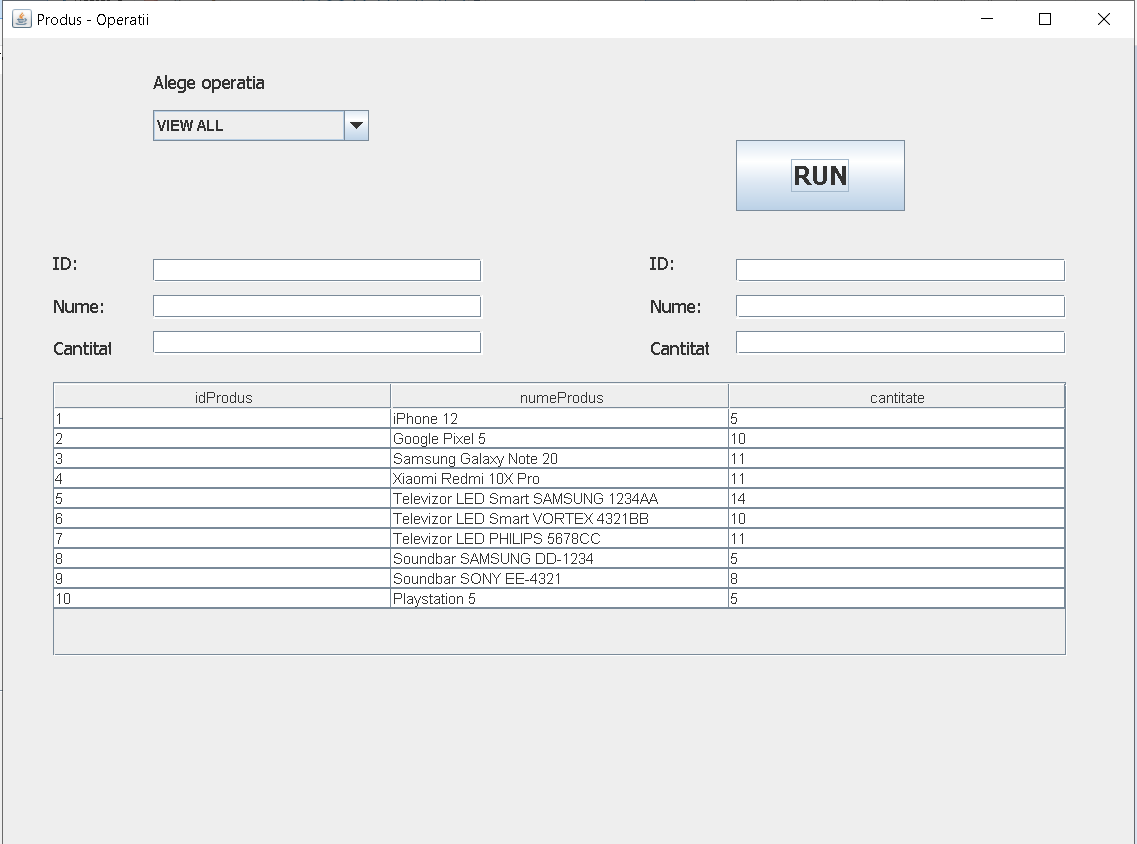
1. **Testare**

Pentru testare vor fi prezentate capturi de ecran din aplicatie.

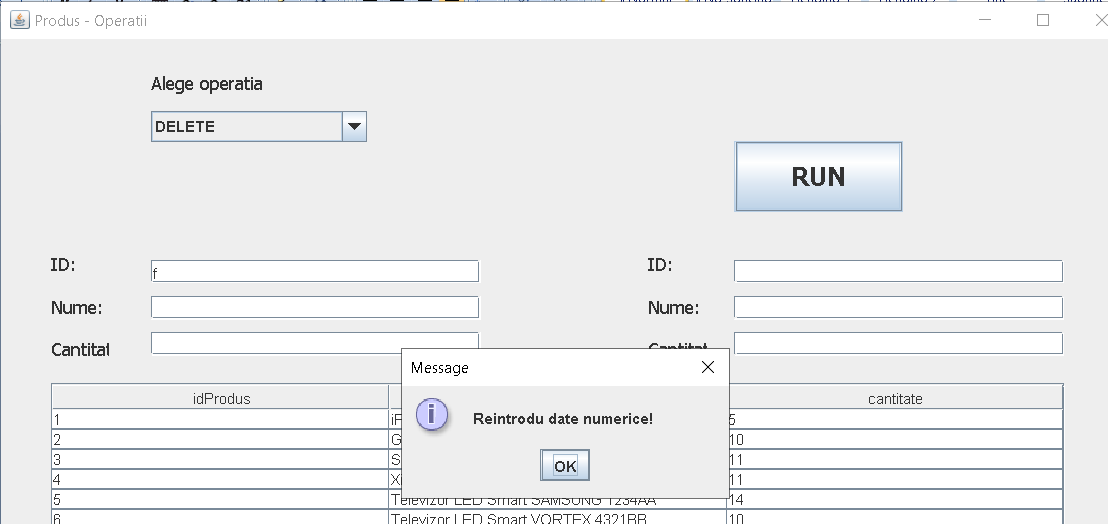
Cautare intr-un tabel:

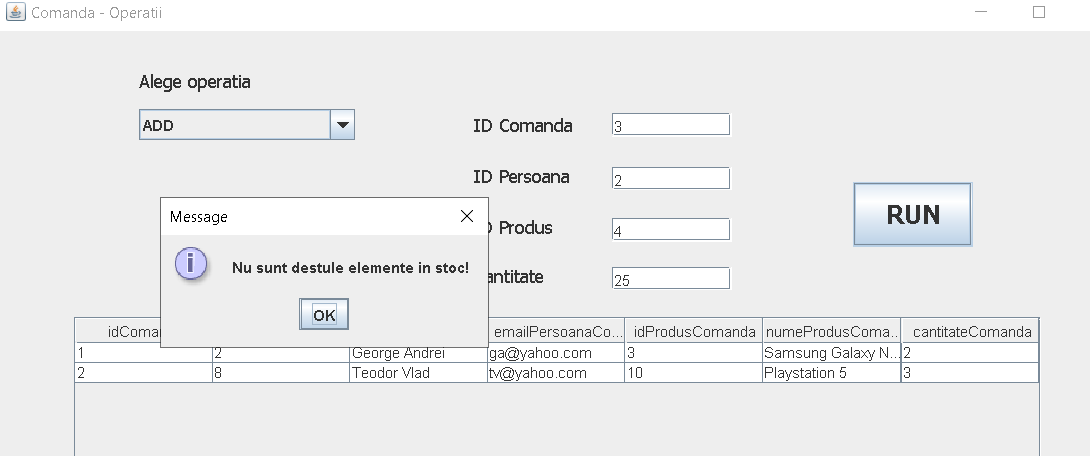


Afisarea tabelului in intregime:

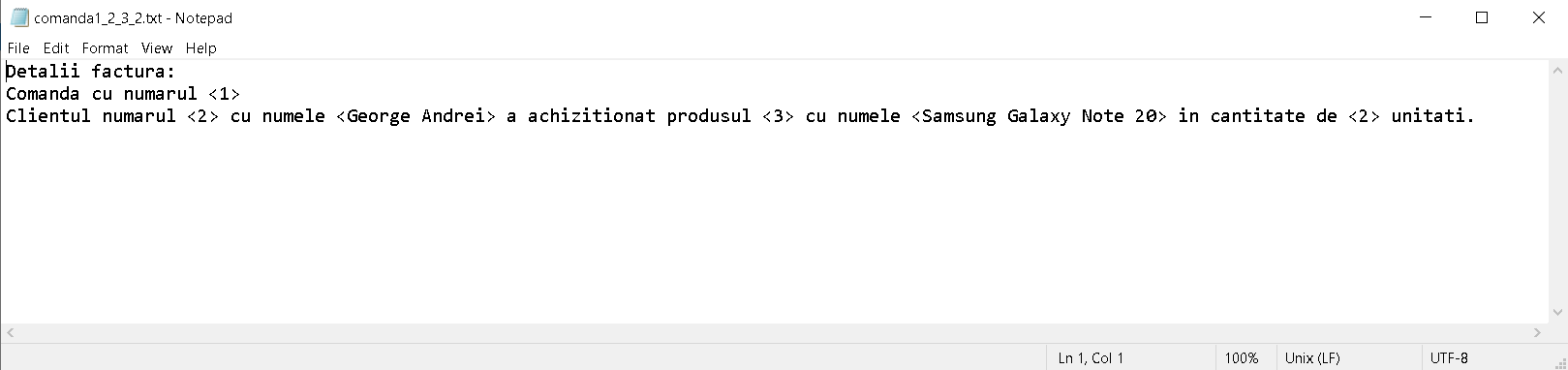


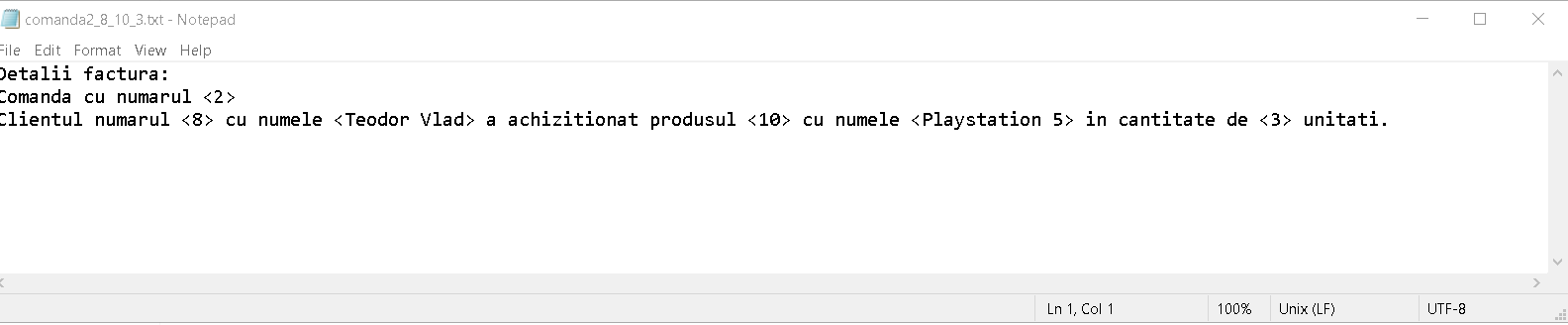
Eroare in aplicatie:





Documentele text, mai precis, facturi:





1. **Concluzii**

In concluzie, acest proiect m-a ajutat sa aprofundez notiunile de Programare Orientata pe Obiect, implicit Java. Consider ca cea mai importanta, complicata dar si interesanta parte a aplicatiei a fost utilizarea bazelor de date, impreuna cu aplicarea tehnicii de „reflection”. De asemenea, un aspect nou pe care nu l-am mai folosit pana acum a fost generarea documentelor JavaDoc. Aplicatia poate beneficia de ulterior de imbunatatiri.

1. **Bibliografie**

<https://en.wikipedia.org/wiki/Javadoc>

<https://gitlab.com/utcn_dsrl/pt-layered-architecture/-/tree/master>

<https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html>

http://tutorials.jenkov.com/java-reflection/index.html