Universitatea Tehnica din Cluj-Napoca

Facultatea de Automatica si Calculatoare

Sectia Calculatoare si Tehnologia Informatiei

Documentatie pentru aplicatie de lucru cu baze de date

Student: Timotei Molcut

Cluj-Napoca

9 Mai 2019

Cuprins

1. Obiectivul temei
2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3. Proiectare
4. Implementare
5. Rezultate
6. Concluzii
7. Bibliografie

Obiectivul temei

Obiectivul principal al temei de fata este de a usura lucrul cu baze de date astfel incat oricine sa poata sa modifice datele existente chiar daca nu are cunostinte despre limbajul SQL.

In acest fel, nu numai programatorii pot avea acces la bazele de date, ci oricine intrebuinteaza aceasta aplicatie.

Pentru a se ajunge la acest obiectiv trebuie bifate inca alte cateva obiective secundare. Unul dintre aceste obiective secundare reprezinta conectarea in limbajul Java la baza de date existenta pe serverul localhost propriu. Modelul acestei conectarii este realizat prin design pattern-ul singleton. Un alt obiectiv secundar reprezinta compunerea dinamica a interogarilor SQL prin care se va modifica baza de date. De asemenea, un obiectiv secundar reprezinta folosirea unei metode generale de extragere a atributelor unui obiect. Aceasta metoda mai are si denumirea de reflection technique. Un nou obiectiv reprezinta organizarea pe mai multe nivele de adancime a aplicatiei. Astfel, interfata grafica se afla la suprafata, iar in „miez” se afla baza de date.

Ca de obicei, un obiectiv secundar ce trebuie implinit pentru dezvoltarea unei aplicatii reprezinta crearea unei interfete grafice prin care utilizatorul sa poata interactiona mult mai usor cu baza de date. Aceasta interfata va fi disponibila pentru a arata dinamic ce se petrece in interiorul bazei de date, ascunzand operatiile pe care aplicatia le desfasoara in interior.

Despre toate aceste obiective se va discuta pe larg in capitolele urmatoare.

Analiza problemei, modelare, scenarii, cazuri de utilizare

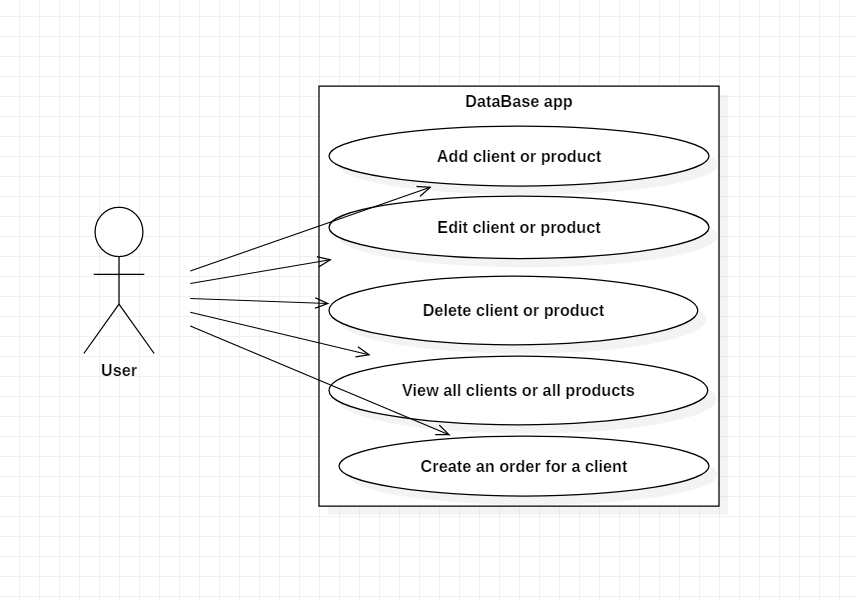
Analizand problema de fata se poate observa ca este nevoie de o organizare pe nivele a aplicatiei. Astfel, toate pachetele importante in care s-a structurat aplicatia vor fi structurate de sus in jos, simbolizand adancimea. La nivelul de la suprafata, adica cel mai inalt nivel se afla prezentarea care tine de interfata grafica. Urmeaza apoi logica de business a aplicatiei. Din business se acceseaza obiectele, care au nevoie de o conectare la baza de date, iar baza de date se afla la cel mai de jos nivel sau miezul aplicatiei.

Pentru conectarea la baza de date este nevoie doar de un obiect care sa faca conexiunea de mai multe ori, asadar se poate folosi modelul de design pattern numit singleton. Acest desing pattern are un concept matematic prin care o multime are un singur element, adica obiectual exista o singura instanta a acestui tip. Prin urmare, aceasta singura instanta va realiza conectarea la baza de date de mai multe ori.

Modelele principale ale acestei aplicatii sunt obiectele de tip Client, Product si Orderr(2 de ‚r’ pentru ca „order” este cuvant deja folosit in limbajul SQL). Aceste tipuri de obiecte au corespondente in tabelele bazei de date cu care se lucreaza, iar atributele fiecarui obiect reprezinta si coloanele tabelelor corespunzatoare. Asadar corespondenta este de „1 la 1” intre tipurile de obiecte si tabelele lor. Pe deasupra, tabelele contin toate instantele create in interiorul aplicatiei.

Scenariile principale care se vor realiza de catre aplicatie sunt: Adaugarea unui client sau a unui produs in baza de date, Editarea unui client sau a unui produs existent in baza de date, Stergerea unui client sau a unui produs existent deja in baza de date si in final vizualizarea intr-un tabel a tuturor clientilor sau produselor din baza de date. De asemenea, aplicatia se mai poate utiliza si pentru a se crea o comanda(care este introdusa in baza de date). Aceasta comanda este ceruta de catre un client care doreste sa cumpere un produs si sa ia din acesta o anumita cantitate. Daca datele sunt valide, adica clientul si produsul exista, iar cantitatea produsului poate fi acoperita, aceasta comanda respecta normele si i se poate crea o factura cu datele clientului si ale produsului. Aceasta factura este sub forma de fisier de tip text.

Toate aceste scenarii sunt descrise in diagrama UML use case de mai jos.



Proiectare

Asa cum am precizat deja, modelele de baza ale acestei aplicatii sunt clientul, produsul si comanda. Pentru fiecare dintre acestea exista un tabel, iar aceste tabele din baza de date contin chei primare reprezentate de id-ul obiectului modelat si chei secundare(pentru tabelul de comenzi) dependente de cheile primare din celelalte tabele(de client si produs).

Pornind de sus in jos, aplicatia incepe cu interfata grafica. Aceasta interfata contine optiunile de adaugare, editare, stergere si vizualizare pentru client sau produs, iar la final se poate crea o comanda noua. Astfel, aceasta interfata este realizata de o clasa denumita View care este dependenta de alte doua clase ce implementeaza ActionListener si sunt folosite pentru cele doua butoane ale interfetei.

Dupa ce s-a selectat o optiune din interfata si s-au adaugat datele de intrare dupa caz, aplicatia isi va duce operatiile inspre centru, adica in pachetul de logica al business-ului. Aici se va verifica daca clientul(sau produsul) de introdus este valid, adica exista cateva clase denumite validatori care implementeaza o interfata generica cu o metoda de validare. Aceasta metoda va arunca anumite exceptii in cazul in care obiectul dat ca paramentru nu este valid. Daca obiectul este valid, in business logic acest obiect va fi adaugat la baza de date prin intermediul unui obiect care acceseaza date(data access object), adica se va cobora si mai adanc in structura aplicatiei. Aceste obiecte de accesare a datelor sunt trei(pentru fiecare obiect in parte) si sunt de fapt mostenitoare ale unui clase generice care implementeaza operatiile amintite mai sus pentru baza de date. Tot in acest obiect se construiesc dinamic(in functie de obiectul folosit) interogarile necesare operatiilor cu baza de date. Din acest nivel se cere conectarea la baza de date, deci se va cobora si mai mult in ierarhia aplicatiei ajungandu-se la nivelul de conectare la baza de date. Dupa cum am mai spus, aici exista o singura clasa denumita DBConnection care foloseste desing pattern-ul singleton, adica se foloseste o singura instanta. Aceasta clasa este dependenta de clasele deja existente denumite Connection, Statement si ResultSet folosite pentru a extrage rezultatul in data access object. De asemenea, aceasta clasa foloseste mai multe obiecte de tip String care contin denumirea serverului si a bazei de date, precum si denumimrea utilizatorului si a parolei necesare pentru conectare.

In final, in urma conectarii la baza de date se ajunge la ultimul nivel sau miezul aplicatiei, adica baza de date in sine, care este de fapt continuta in serverul localhost. Toate aceste nivele sunt dependente individual de catre modelul de baza, adica client, produs si comanda.

Detaliile de mai sunt sunt descrise sub forma unor diagrame UML de clase si pachete, mai jos.

Diagrama UML de clase

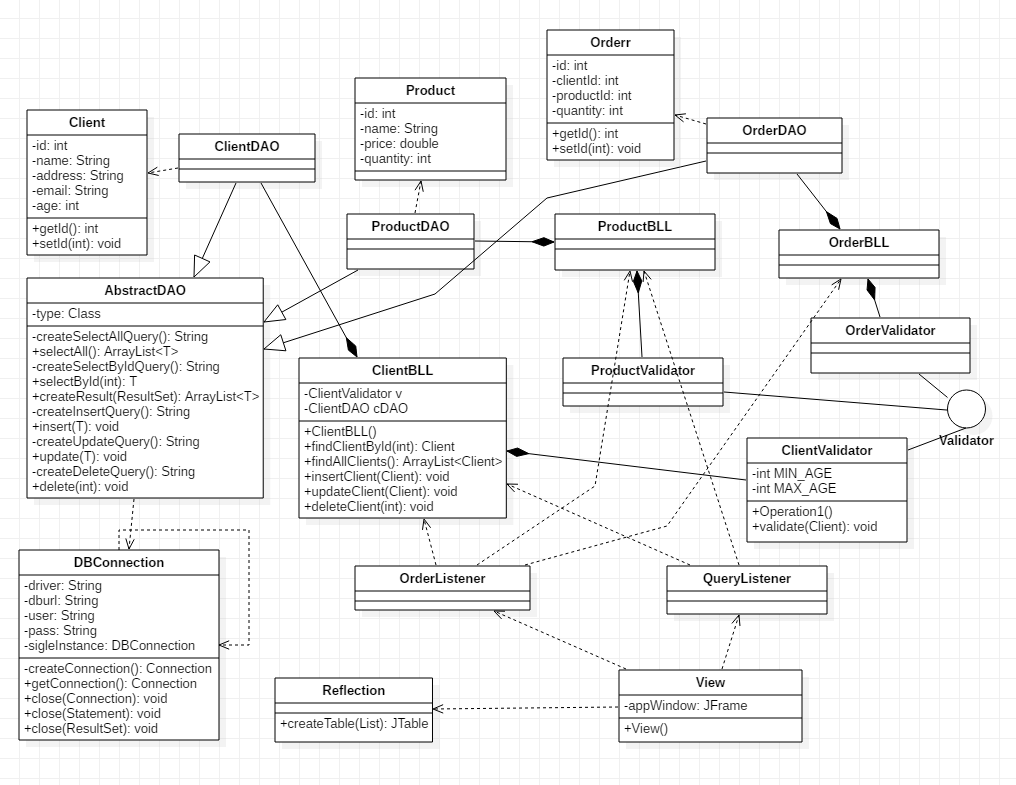
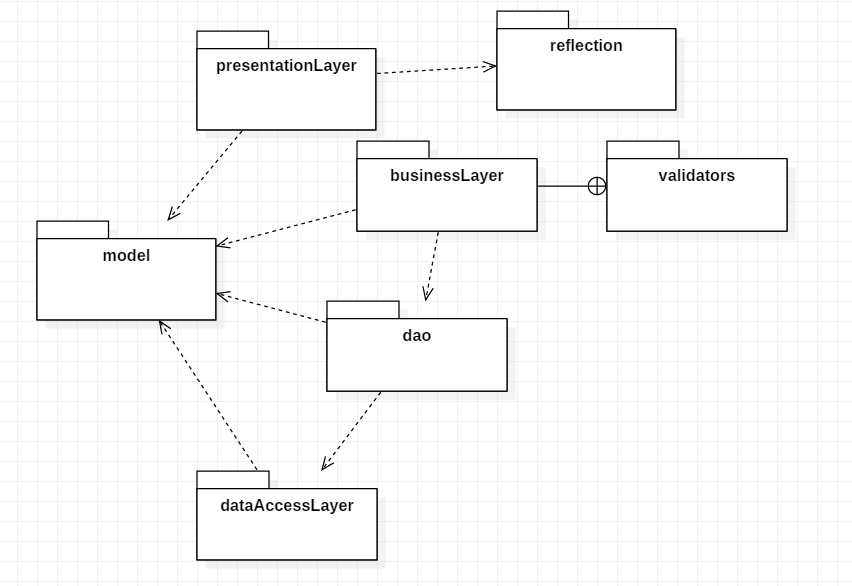


Diagrama UML de pachete de mai jos, surprinde organizarea „sferica”(adica de la suprafata la miez) a aplicatiei despre care am tot vorbit mai sus. PresentationLayer este la suprafata, dependent de catre businessLayer, iar businessLayer continua dependenta pana la dataAccessLayer care este in centru.



Implementare

Acest capitol este dedicat implementarii aplicatiei. Deoarece aceasta aplicatie este dependenta de o baza de date implementarea a fost mai costisitoare decat in general avand o structura ierarhica despre care am vorbit mai sus.

Obiectele principale ale acestei aplicatii sunt Client, Product si Orderr. In jurul acestora se „joaca” celelalte obiecte care au rolul de modelare a acestor obiecte. De asemenea, aceste obiecte sunt singurele reale, adica palpabile in lumea reala(fiind si singurele obiecte din baza de date). Voi incepe cu descrierea implementarii acestor obiecte.

Obiectul Client contine atributele: id, name, address, email si age. Acest obiect, la fel ca si restul contine trei constructori: unul simplu fara niciun parametru, unul cu toate parametrele necesare fiecarui atribut, iar cel din urma contine valori pentru toate atributele mai putin pentru id. Acest constructor din urma este folosit cel mai des, deoarece in implementarea bazei de date, cheia primara adica id-ul este auto-incrementat, asadar nu trebuie specificat in adaugarea clientului in baza de date. Acest aspect este valabil si pentru celelalte clase. Client mai contine si get-ere si set-ere pentru fiecare atribut, care este privat, si se suprascrie metoda toString pentru afisarea prietenoasa a acestui obiect. Aceste aspecte sunt valabile si pentru Product si Orderr. Spre deosebire de aceasta, Product contine atributele: id, name, price si quantity, iar Orderr contine atributele: id, clientId, productId si quantity. Prin urmare, Orderr va trebui sa aiba clientId existent intr-un id al lui Client, idem pentru productId, iar quantity din Orderr, va trebui sa aiba o valoare mai mica sau egala cu cea a produsului.

Lasand modelul la o parte, putem incepe in descrierea ierarhiei de sus in jos. Asa cum am mai spus deja prezentarea contine trei clase. Cea principala este denumita View. Aceasta contine doua component de tip JComboBox folosite pentru pentru a selecta operatia si operandul. In centrul ferestrei pe partea stanga se pot introduce datele valabile pentru tabelul de Client, iar in partea dreapta se pot introduce datele valabile pentru tabelul Product. La finalul paginii se pot introduce id-ul clientului si al produsului impreuna cu cantitatea acestuia, pentru a se crea o comanda. Din cele descrise mai sus rezulta ca este nevoie de doua butoane. Primul care foloseste QueryListener este folosit pentru interogare, iar al doilea folosit pentru crearea de comenzi foloseste OrderListener. Dupa cum le spune si numele acesti ascultatori apeleaza o metoda ori de cate ori butoanele corespunzatoare sunt apasate. Pentru metoda asculatatorului QueryListener se vor verifica selectiile din fereastra principala de la cele doua componente JComboBox. In functie de selectia aleasa se vor realiza anumite operatii, dar nu inainte de a se extrage datele din JTextField-uri necesare. Doar in cazul operatiei de vizualizare se va crea o noua fereastra care va contine datele din tabelul corespunzator din baza de date. In cazul in care se semnaleaza erori legate de operatiile existente, exista doua componente de tip JLabel folosite pentru a arata utilizatorului aceste erori(cate unul pentru fiecare buton).

Coborand mai jos in ierarhie se ajunge la blocul de business. Aici sunt implementate pentru fiecare tip trei clase de business logic. In fiecare clasa sunt implementate metodele de adaugare a unui obiect, selectare a unui obiect dupa id, selectarea tuturor obiectelor dintr-un tabel, editarea unui obiect, si stergerea unui obiect. Aproape fiecare metoda foloseste un validator care verifica daca obiectul dat ca parametru respecta conditiile necesare pentru baza de date. Fiecare validator este o clasa care suprascrie o metoda de validare corespunzatoare fiecarui obiect. Aceasta metoda poate arunca o exceptie numita IllegalArgumentException, in cazul in care atributele obiectului nu sunt bune. Astfel in acest caz obiectul respectiv nu va fi adaugat in baza da date, iar utilizatorul va fi semnalat de acest lucru.

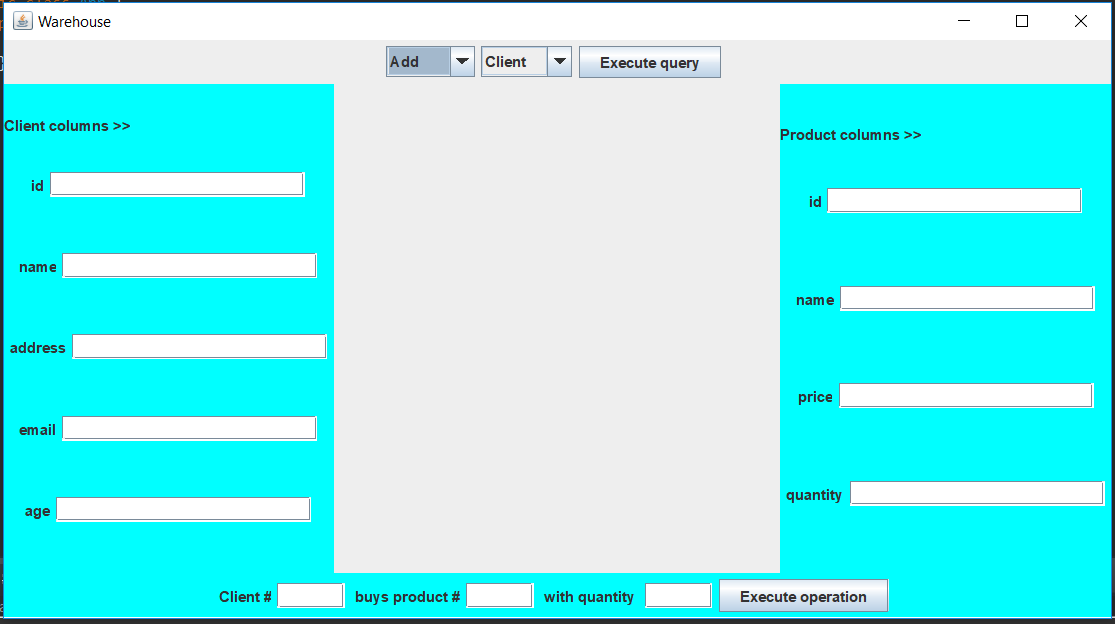
In cazul in care obiectul se poate adauga, se trece la urmatorul nivel numit dao(data access object) care va realiza operatia in sine. La acest nivel sunt existente patru clase. Prima denumita AbstractDAO<T> este generica, iar celelalte ClientDAO, ProductDAO si OrderDAO sunt mostenitoare ale acestia folosind tipul corespunzator. In clasa generica sunt implementate metodele care realizeaza practic operatiile descrise mai sus in interfata grafica. Pentru fiecare metoda de operatie exista o alta metoda care concateneaza dinamic la un sir de caractere ce este necesar pentru instructiunea respectiva si astfel va returna aceasta instructiune sub forma unui String. In metodele de operatii, acel string va capata valorile, unde este cazul ale obiectelor date ca parametru. De exemplu, pentru metoda de insert, se vor accesa atributele necesare fara id(motivul este descris mai sus), iar valorile vor fi adugate pe loc inainte de executia interogarii extragandu-se valorile atributelor obiectului de inserat.

Din data access object se cere o conectare la baza de date realizata de catre o singura instanta a clasei DBConnection(motivul este descris mai sus). Conectarea se realizeaza folosind serverul, utilizatorul si parola ca obiecte de tip String.

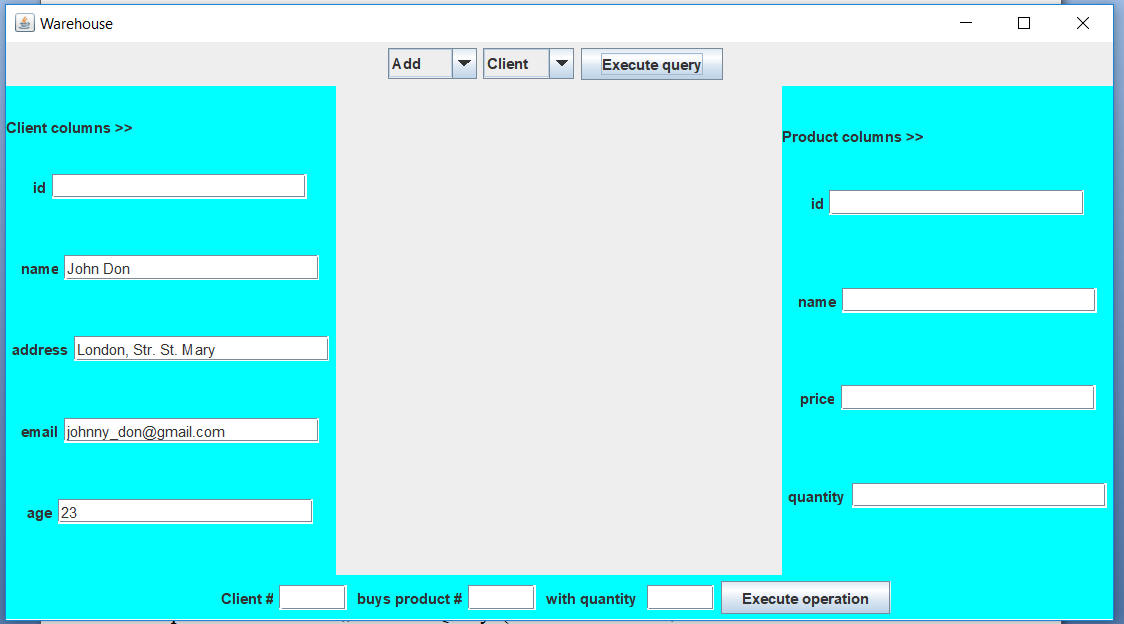
Un aspect particular despre care nu am vorbit reprezinta metoda createTable din clasa Reflection care foloseste tehnici de reflection pentru a extrage campurile(Field) pentru o lista de obiecte, creand coloanele unui tabele cu numele acestori campuri, iar valorile existente in atributele fiecarui camp vor fi inserate in tabel pe coloana corespunzatoare. Campurile se acceseaza astfel: obiectul „o” apeleaza metoda getClass(), iar apoi se apeleaza metoda getDeclaredFields() care va returna campurile corespunzatoare clasei obiectlui. Valoarea unui camp al unui obiect se acceseza astfel: campul „field” apeleaza metoda get() asupra obiectului de la care se doreste valoarea respectiva. Astfel, cu un tablou unidimensional cu numele coloanelor si inca un tablou bidimensional cu valorile pentru fiecare coloana si fiecare obiect din lista data ca parametru pentru metoda se va crea un JTable care va fi si returnat. Aceasta metoda va fi apelata in interfata grafica pentru operatia de vizualizare.

Rezultate

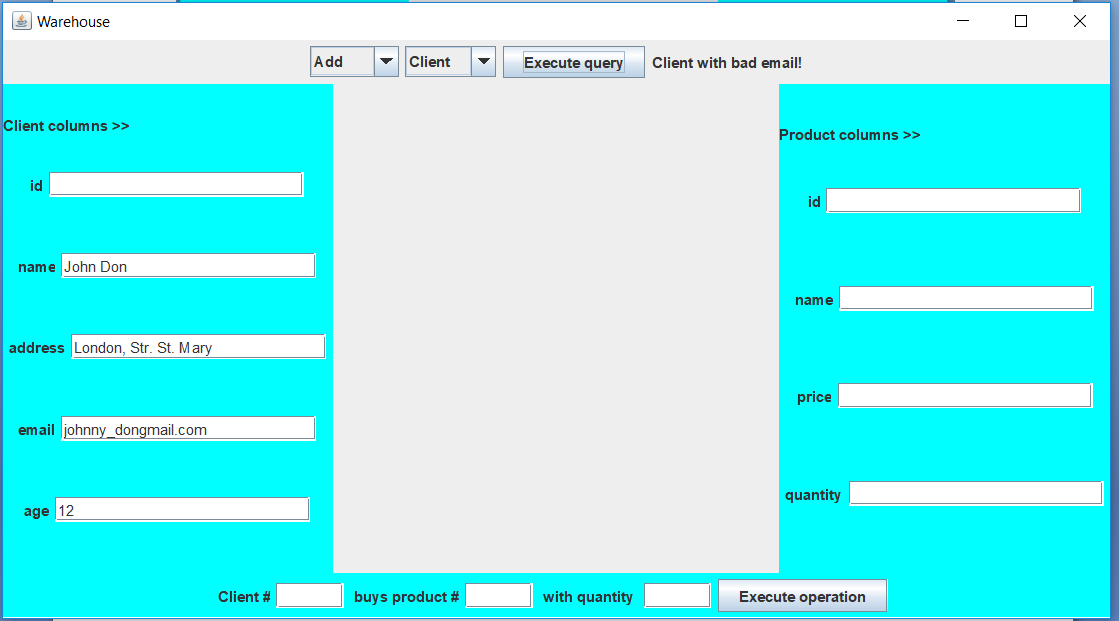
La pornire aplicatia arata in felul urmator.



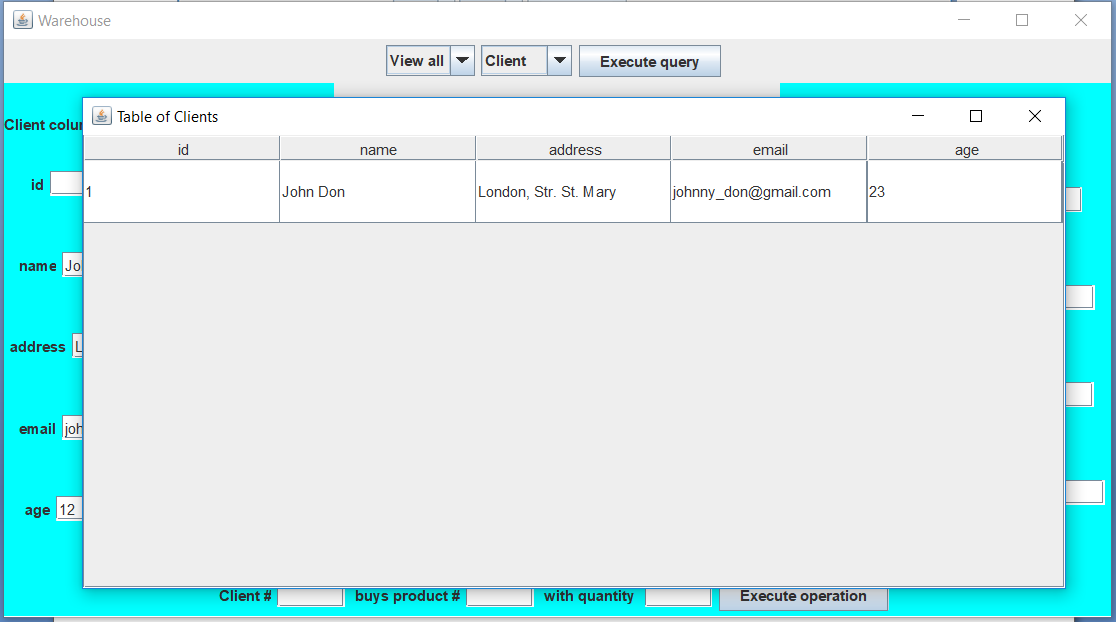
Dupa inserarea unor valori pentru client se poate adauga acest nou client prin apasarea butonului „Execute Query”(nu conteaza id-ul, acesta se auto-incrementeaza in baza de date).



In cazul in care clientul este invalid se va semnala o eroare. In cazul de fata eroarea este: „Client with bad email”, adica nu contine ‚@’ si, de asemenea, varsta nu este potrivita.



In cazul vizualizarii clientilor(optiunea „View all”) apare o noua fereastra cu un JTable.



Concluzii

In concluzie, pot sa spun ca am dobandit cunostinte noi legate de posibilitatea conectarii la o baza de date, posibilitatea accesarii campurilor unui obiect, fara a le cunoaste, prin tehnici de reflection si posibilitatea executarii unor interogari SQL, folosind limbajul Java.

Imbunatatirile pe care as vrea sa le aduc sunt legate de baza de date(adaugarea unui nou tabel) si felul in care se desfasoara interfata grafica.

Bibliografie

* World Wide Web
* <https://stackoverflow.com>
* <https://www.oracle.com>
* <https://www.w3schools.com/sql>
* <https://www.tutorialspoint.com>
* <https://en.wikipedia.org/wiki/Singleton_pattern>