**Documentatie**

Assigment 3

Nume: Neacă Radu-Sabin

Grupa: 30222

**Cuprins**

[1. Obiectivul temei 3](file:///C:\Downloads\FACULTATE\Tehnici%20de%20programare\PT2021-2022_Documentation_Template.doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///C:\Downloads\FACULTATE\Tehnici%20de%20programare\PT2021-2022_Documentation_Template.doc#_Toc95297886)

[3. Proiectare 5](file:///C:\Downloads\FACULTATE\Tehnici%20de%20programare\PT2021-2022_Documentation_Template.doc#_Toc95297887)

[4. Implementare 6](file:///C:\Downloads\FACULTATE\Tehnici%20de%20programare\PT2021-2022_Documentation_Template.doc#_Toc95297888)

[5. Rezultate 8](file:///C:\Downloads\FACULTATE\Tehnici%20de%20programare\PT2021-2022_Documentation_Template.doc#_Toc95297889)

[6. Concluzii 8](file:///C:\Downloads\FACULTATE\Tehnici%20de%20programare\PT2021-2022_Documentation_Template.doc#_Toc95297890)

[7. Bibliografie 9](file:///C:\Downloads\FACULTATE\Tehnici%20de%20programare\PT2021-2022_Documentation_Template.doc#_Toc95297891)

Obiectivul temei

Obiectiv principal:

Scopul lucrarii consta in crearea unei aplicatii ce efectueaza diferite operatii intr-o baza de date. Acestea operatii sunt de inserare, de stergere si de actulizare. In realizarea acetul proiect m-am folosit de o baza de date creata in MySql, interfata acesteia fiind prezentata cu ajutorul javaFx. Pe parcursul proiectului m-am folosit si de alta metode de scriere, precum reflexion tehnic.

Obiective secundare:

* Analiza problemei si identificarea cerintelor.
* Organizarea codului corespunzaor: Precum structura MVC (Model-View-Controller). Fiecare clasa creata nu trebuie sa depaseasca o dimensiune mai mare de 300 de linii, cu exceptia clasei Controller. Fiecare metoda trebuie sa cuprinda maxim 30 de linii. Clasele trebuie impartite in pachete astfel incat sa existe o vizibilitate si organizare a proiectului mult mai buna.
* Testarea aplicatiei, verificarea codului astfel incat metodele folosite pe parcursul programului sa fie corect implementate, fara a prezenta cazuri de eroare.
* Implementarea si simularea unei interfete ce sa dispuna si de o fereastra ce afiseaza rezultatul dorit.
* Folosirea tehnicilor de reflexivitate, dar si a cunostintelor in crearea unei baze de date (inserarea , stergerea si actualizarea datelor).

Analiza problemei, modelare, scenarii, cazuri de utilizare

Primul pas inainte de implementarea proiectului ar fi analiza problemei si modul cum ar trebui rezolvata aceasta. Un bun inceput il poate reprezenta alcatuirea bazei de date. Astfel, am decis sa mi construiesc baza de date cu ajutorul sistemului MySql, in care am indrodus urmatoarele tabele: clients, products, orders. Fiecare dintre acestea sunt alcatuite la randul lor din diverse attribute precum un ID unic, ce reprezinta cheia primara din baza de date. Un alt atribut il reprezitna ID-ul cu ajutorul caruia clientii/produsele/comenzile sunt identificate in java. Celelalte atribute au un rol semnificativ pentru fiecare tabela. Tot in MySql am realizat inserarea provizorie a unor clienti/produse/comenzi.



Scopul cerintei este de a afisa, de a insera, de a sterge si de a actualiza fiecare client/produs/comanda. Pentru realizarea interfetei acestor operatii, m-am folosit de aplicatia “SceneBuilder” si de conectarea fisierului cu javaFx.

A screenshot of a computer

Description automatically generated with medium confidence

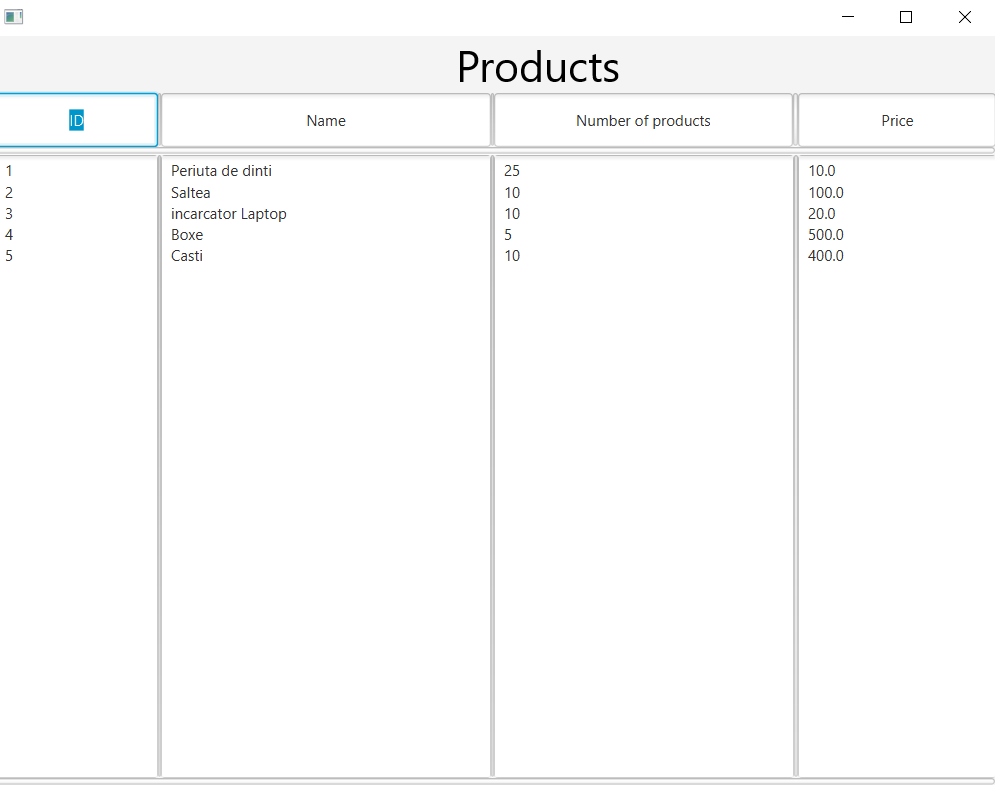
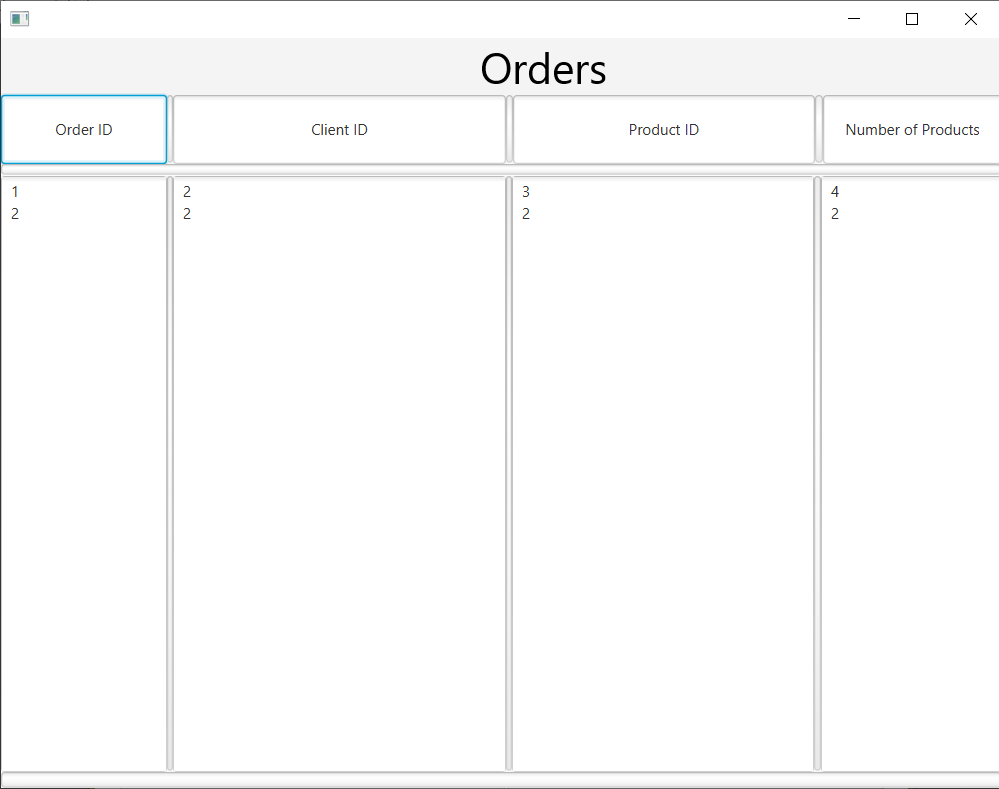
Aceasta este prima fereastra a interfetei. Aici am inserat fiecare operatie pe care dorim sa o facem cu ajutorul unor butoane. De fiecare data cand are loc apasarea unui buton, o sa se realizeze cate o metoda diferita. Pentru buton, metoda respective o sa reprezinta o “actiune”. Spre exemplu primul buton are urmatoarea actiune:

Graphical user interface

Description automatically generated with medium confidence

Pentru afisarea clientilor m-am folosit de mai multe tabele de tip TextArea. Primul TextArea este alcatuit din toate ID-urile clientilor, al doilea este alcatuit din numele lor, al treilea din email-urile lor iar ultimul din varstele lor. Similar cu clientii, afisarea produselor se realizeaza cu ajutorul a 4 TextArea. Unul pentru ID, al doilea pentru nume, al treilea pentru numarul de produse disponibile, iar al patrulea pentru pretul fiecaruia. Afisarea comenzilor continue doar id-ul clientilor, al produselor si numarul de produse cumparate de catre clienti.

Table

Description automatically generated  

Proiectare

Pentru proiectarea codului, am urmat structura MVC (Model-View-Controller), astfel incat fiecare pachet adaugat sa aibe un rol specific. In pachetul “connection” are loc conectarea la baza de date MySQL. In pachetul “dataAccess” are loc salvarea informatiilor din baza de date in liste, dar si actualizarea acesteia dupa inserarea, actualizarea sau stergerea unui client, produs sau comanda. In pachetul “model” am plasat clasele “Client”, “Order”, “Product”. Iar in pachetul “controller” se realizeaza pornirea aplicatiei si editarea interfetei. Cum fiecare proiect in java se incepe cu o diagrama UML, a mea este urmatoarea:

A screenshot of a computer

Description automatically generated with medium confidence

Se poate observa faptul ca nu am realizat vreo legatura intre clase, neavand vreo instanta a unei alte clase intr-o alta clasa. Pe langa pachetele de mai sus, sunt prezente si resursele primite datorita aplicatiei SceneBuilder. Aici sunt prezente toate ferestrele aplicatiei, cum ar fi urmatoarele:

Text

Description automatically generated with medium confidence

Implementare

In implementarea aplicatiei am urmat structura MVC (Model-View-Controller) care sa respecte conditiile unei bune organizari a codului, astfel incat sa nu am clase ce depasesc de 300 de linii sau metode ce au mai mult de 30 de linii. Dupa cele prezentate in capitolul anterior, acum putem discuta si despre clasele si metodele apelate. Astfel, incepem cu Clasa unde se lanseaza aplicatia. Primul lucru care trebuie respectat cand o aplicatie se leaga cu o baza de date, este conectarea codului la baza de date. Astfel, prima data are loc instantierea obiectului “dataAccess” urmand sa fie respectate instructiunile din constructorul acesteia. In “DataAccess” se realizeaza instantierea obiectului “mySql” reprezentand mijlocul de conectare al bazei de date. Astfel ajungem in punctul de conectare al codului.

In clasa “MySql” avem urmatoarele variabile private “USERNAME”, “PASSWORD”, “CONNECTION LINK”, “SINGLE\_INSTANCE”. Toate sunt de tip static, urmand sa fie utilizate doar in metode statice, dar si de tip “final” urmand sa iti pastreze aceeasi valoare pe parcursul compilarii. Variabila “USERNAME” primeste ca initializarea, numele host-ului din baza de date. Similar cu aceasta, variabila “PASSWORD” primeste parola host-lui . Variabila “CONNECTION LINK” se initializeaza cu “jdbc:mysql://localhost:3306/virtualShop”, reprezantand o cale de access catre baza de date. Numele schemei cu baza de date este present intotdeauna la finalul link-ului, acesta fiind “virtualShop”. Dupa initializarea acestora, se afla metodele cu conecaterea proiectului la baza de date. In “createConnection” are loc conectarea la server cu ajutorul package-ului “connection” prin comanda DriverManager.getConnection(CONNECTION\_LINK, USERNAME, PASSWORD) . Din aceasta se apeleaza metoda “getConnection” care retine linkul de conectare la baza de date. Metoda “close” realizeaza inchiderea conexiunii la baza de date.

Ne intoarcem in clasa “DataAccess” unde avem acces deja la baza de date. Memorarea clientilor, a produselor si a comenzilor se realizeaza prin comanda statement.executeQuery(), urmand ca in paranteze sa fie puse tabelele pe care dorim sa le modificam. Valorile primate le salvam intr-un obiect de tip “ResultSet” dupa crearea unui “Statement” dar si a conexiunii la baza de date.

In clasa “InsertClient” m-am folosit de salvarea fiecarui client intr-o lista de clienti. Metodele din aceasta clasa sunt “showClientsFromDataBase”, “showClientsList” , “insertOneClient” , “insertOneClientInDataBase” , “deleteOneClient” , “deleteOneClientFromDataBase” , “editOneClient” , “editOneClientFromDataBase” reprezentand metode de afisare de inserare de stergere si de editare a clientilor. In metodele “showClientsFromDataBase” si “showClientsList” au loc 2 afisari diferite. In a doua are loc afisarea in consola a clientilor salvati in lista la momentul actual, iar in cea de a 2-a metoda are loc afisarea clientilor ce au fost inserati si/sau memorati in baza de date. In metodele “insertOneClient” , “insertOneClientInDataBase”, se realizeaza inserarea unui singur client, Prima metoda reprezinta inserarea clientului in lista de clienti, list ace urmeaza a fi trimisa catre afisarea in consola. A doua metoda reprezinta inserarea in baza de date, pentru a memora clientul permanent in aceasta. In metodele “deleteOneClient” , “deleteOneClientFromDataBase” are lor stergerea unui singur client din baza de date. Prima metoda sterge clientul din lista si il trimite catre a 2 a metoda pentru a-l sterge si din baza de date. In metodele “editOneClient” , “editOneClientFromDataBase” are loc actualizarea unui singur client, al ID-ului, numelui, email-ului sau al varstei. Prima realizeaza acest process, urmand sa trimita catre a 2-a metoda lista cu clienti actualizata, pentru a o memora in baza de date.

In clasa “InsertProduct” m-am folosit de salvarea fiecarui produs intr-o lista de produse. Metodele din clasa sunt “showProductsFromDataBase”, “showProductsList” , “insertOneProduct”, “insertOneProductInDataBase” , “deleteOneProduct” , “deleteOneProductFromDataBase” , “editOneProduct” , “editOneProductFromDataBase”, “decrementProduct” reprezentand metode de afisare de inserare de sterger si de editare a clientilor. In metodele “showProductsFromDataBase” si “showProductsList” au loc 2 afisari diferite. In a doua are loc afisarea in consola a produselor salvate in lista la momentul actual, iar in cea de a 2-a metoda are loc afisarea produselor ce au fost inserate si/sau memorate in baza de date. In metodele “insertOneProduct” , “insertOneProductInDataBase”, se realizeaza inserarea unui singur produs, Prima metoda reprezinta inserarea produsului in lista de produse, lista ce urmeaza a fi trimisa catre afisarea in consola. A doua metoda reprezinta inserarea in baza de date, pentru a memora produsul permanent in aceasta. In metodele “deleteOneProduct” , “deleteOneProductFromDataBase” are lor stergerea unui singur produs din baza de date. Prima metoda sterge produsul din lista si il trimite catre a 2 a metoda pentru a-l sterge si din baza de date. In metodele “editOneProduct” , “editOneProductFromDataBase” are loc actualizarea unui singur produs, al ID-ului, numelui, numarului de exemplare sau al pretului. Prima realizeaza acest process, urmand sa trimita catre a 2-a metoda lista cu produse actualizata, pentru a o memora in baza de date. In metoda “decrementProduct” are loc decrementarea numarului de produse in cazul in care o comanda continue ID-ul acestuia.

In clasa “InsertOrder” m-am folosit de o lista de comenzi pentur a le memora in baza de date dar si pentru a realiza decrementarea produselor in cazul in care o comanda continue ID-ul unui anumit produs. Metodele de care m-am folosit sunt “insertOneOrder”, “insertOneOrderInDataBase”, “showOrdersFromDataBase”, “showOrdersList”. Metodele “insertOneOrder”, “insertOneOrderInDataBase” au rolul de a insera in baza de date o comanda. Prima metoda adauga comanda in lista de comenzi, urmand sa fie trimisa catre a doua metoda pentru o insera in baza de date. Tot in prima metoda are loc decrementarea produsui in cazul in care ID-ul inserat in comanda este egal cu ID ul unui produs din baza de date. Metodele “showOrdersFromDataBase”, “showOrdersList” au rolul de a afisa comenzile. Prima metoda afiseaza comenzile salvate din baza de date, iar a doua metoda afiseaza in consola lista de comenzi.

Pentru a detalia fiecare client, podus sau comanda, in pachetul “model” se afla 3 clase de baza.  
 In clasa “Client” se afla toate atributele unui client din baza de date precum : ID-ul ce reprezinta, un identificatory unic,“name” ce reprezinta numele clientului, “email” ce reprezinta email-ul fiecarui client si “age” ce reprezinta varsta fiecarui client. In constructor retinem atributele lor in momentul in care se instantieaza un client nou. Sunt prezente si setter-e si getter-e pentur folosirea mai usoara a codului.

In clasa “Product” se afla toate atributele unui produs din baza de date precum : ID-ul ce reprezinta, un identificatory unic,“name” ce reprezinta numele produsului, “number” ce reprezinta numarul de produse din stoc si “price” ce reprezinta valoarea fiecarui produs. In constructor retinem atributele lor in momentul in care se instantieaza un produs nou. Sunt prezente si setter-e si getter-e pentur folosirea mai usoara a codului.

In clasa “Order” se afla toate atributele unei comenzi din baza de date precum : orderID-ul ce reprezinta, un identificatory unic pentru comenzi, “clientID” ce reprezinta, un identificatory unic pentru clienti, “productID” ce reprezinta, un identificatory unic pentru produse, “numberOfProducts” numarul de produse ce urmeaza a fi sterse. In constructor retinem atributele lor in momentul in care se instantieaza un client nou. Sunt prezente si setter-e si getter-e pentur folosirea mai usoara a codului.

In clasa “Controller” se afla tot ce tine de interfata aplciatiei. Fiecare metoda reprezinta o actiune diferita ce are loc pentru afisarea rezultatului dorit. Metoda “initialize” are rolul de a se efectua mereu inaintea altor metode cu extensia “@FXML”. Metoda “clickShowClients” are rolul de a afisa intr-o fereastra noua clientii cu atributele lor, cu ajutorul unel variabile de tip TextArea. Metoda “clickShowProducts” are rolul de a afisa intr-o fereastra noua produsele cu atributele lor, cu ajutorul unei variabile de tip TextArea. Metoda “clickShowOrders” are rolul de a afisa intr-o fereastra noua comenzile cu atributele lor, cu ajutorul unel variabile de tip TextArea. Metoda “clickInsertClientWindow” are rolul de a deschide fereastra de inserare a clientilor. Metoda “clickInsertProductWindow” are rolul de a deschide fereastra de inserare a produselor. Metoda “clickInsertOrderWindow” are rolul de a deschide fereastra de inserare a comenzilor. Metoda “ClickInsertClientButton” are rolul de a deschide fereastra de inserare a clientilor in momentul apasarii butonului respective. Metoda “ClickInsertProductButton” are rolul de a deschide fereastra de inserare a produselor in momentul apasarii butonului respectiv. Metoda “ClickInsertOrderButton” are rolul de a deschide fereastra de inserare a comenzilor in momentul apasarii butonului respectiv. Similar cu acestea sunt si metodele “clickDeleteOneClient”, “clickDeleteOneProduct”. Metoda “clickDeleteButton” are rolul de inchidere a ferestrei in momentul in care se apasa butonul. Metoda “clickInsertButton” are rolul de inchidere a ferestrei in momentul in care se apasa butonul. Metoda “clickEditButton” are rolul de inchidere a ferestrei in momentul in care se apasa butonul. Metodele “clickEditClientButton”, “clickEditClient”, “clickEditProduct” au rolul de a edita atat clienti cat si produse.

Cateva metode ajutatoare folosite in clasa “Controller” sunt: “writeClients”, “writeProducts”, “writeOrders”, “showClientsWindow”, “showProductsWindow”, “showOrdersWindow”, ce au rol de a transforma informatiile primate in string-uri si de a le trimite catre metodele de afisare din interfata.

Rezultate

Ca rezultate avem corectitudinea implementarii codului, astfel incat, de fiecare data cand se insereaza un nou client, produs, sau comanda, acesta este salvat in baza de date, pana la modificarea acesteia din MySQL. Astfel, am respectat cerinta si am implementat corespunzator operatiile de inserare, stergere si editare a clientilor, produselor si a comenzilor.

Concluzii

Cateva concluzii ale acestui assignment ar fi date atat de faptul ca m-am familiarizat mai mult cu modelarea unei baze de date, conectarea prin java a acesteia, dar si implementarea codului folosind reflexion tehnic.

Bibliografie

1. <https://dsrl.eu/courses/pt/materials/PT2021-2022_Assignment_3.pdf>
2. <https://dsrl.eu/courses/pt/materials/A3_Support_Presentation.pdf>
3. <http://elf.cs.pub.ro/poo/laboratoare/reflection>
4. <https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html>
5. <https://www.geeksforgeeks.org/java-swing-jtable/>