**Logo

Description automatically generated**

**Documentație pentru tema 3 la materia**

**Tehnici de programare fundamentale**

Student: Pop Adrian

grupa:30223

seria:A

An universitar:2020-2021

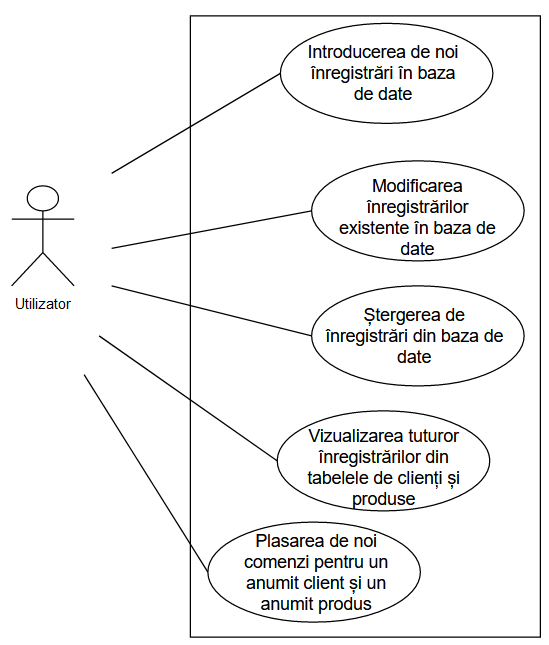
**1.Obiectivul temei:**

Obiectivul principal al acestei teme constă în elaborarea unui program ce permite modificarea și citirea unei baze de date ce conține informații legate de clienți, produse și comenzile depuse de clienți pentru anumite produse. Interacțiunea cu baza de date se realizează prin intermediul unei interfețe grafice.

Obiective secundare:

1. Analiza problemei și a algoritmilor necesari de implementat. Se analizează cerințele programului și se notează funcționalitățile acestuia si interacțiunea cu utilizatorul (2. Analiza problemei)
2. Crearea unei diagrame UML pentru elaborarea unei implementări pentru clase si a interacțiunii dintre acestea. Desenarea claselor împreună cu relațiile dintre acestea permit la o înțelegere mai bună a structurării programului și a logicii din acesta. (3. Proiectare)
3. Creearea unei baze de date simple ce va conține tabele cu date corespunzătoare clienților, produselor ce pot fi cumpărate de aceștia și comenzile clienților pentru un anumit produs.
4. Creearea unor clase ce vor fii legate printr-o conexiune de baza de date și care conțin diferite metode pentru preluarea și modificarea datelor din tabele. Se va face o clasă de acest gen pentru fiecare model de dată.(4. Implementare)
5. Creearea unor clase pentru reprezentarea modelelor de date. Acestea vor fi făcute în concordanță cu datele din tabelele pe care le reprezintă. (3. Proiectare, 4.Implementare)
6. Creearea unei clase cu metode folosite în verificarea corectitudinii datelor ce vor putea fi scrise în baza de date. (4. Implementare)
7. Creearea claselor și a interfeței pentru realizarea interogărilor de vizualizare, modificare și introducere de informații noi în baza de date. Alegerea căror date să fie adăugate se face prin intermediul unei clase MainPage ce va conșine butoane pentru alegerea ferestrei corespunzătoare uneia dintre modelele de date.(4.Implementare)

**2.Analiza problemei:**



Întreaga interacțiune cu programul se realizează prin GUI. Din prima fereastră utilizatorul va putea alege tipul de data pe care dorește să realizeze modificări/vizualizări: clienți, produse, comenzi. În urma apăsării unui buton se va deschide o nouă fereastră unde, în cazul clienților și a produselor, utilizatorul va putea introduce noi înregitrări în baza de date, va putea modifica sau chiar șterge înregistrări existente și va putea vizualiza conținutul întregii tabele. Pentru comenzi utilizatorul va putea alege un client existent, un produs existent, să introducă o cantitate valabilă din produsul respective și să plaseze comanda. Dacă comanda s-a înregistrat cu success, atunci se va creea un fișier text ce va reprezenta chitanța comenzii.

Use Case: Introducerea de noi înregistrări în baza de date

Actor principal: Utilizatorul

Scenariu in caz de success:

1. Utilizatorul introduce datele corect, corespunzând cu tipul de dată din tabela coresnpunzătoare înregistrării
2. Utilizatorul apasă butonul “Add …”
3. Este introdusă o nouă înregistrare în tabela corespunzătoare
4. Se golesc casetele text.

Scenariu in caz de eșec:

1. Utilizatorul introduce în casetele text date necorespunzătoare
2. Casetele text nu se golesc
3. Va apărea o fereastră Popup cu un mesaj corespunzător

Use Case: Modificarea înregistrărilor existente în baza de date

Actor principal: Utilizatorul

Scenariu în caz de success:

1. Utilizatorul introduce datele dorite corect și alege una dintre înregistrările din tabela corespunzătoare
2. Utilizatorul apasă butonul “Edit…”
3. Se realizează modificările înregistrării alese
4. Casetele text în care au fost introduse datele pentru modificare sunt golite.

Scenariu in caz de eșec:

1. Utilizatorul introduce în casetele text date necorespunzătoare sau nu a ales o înregistrare în care să se realizeze modificările
2. Casetele text cu date greșite nu se golesc și se realizează doar modificările corecte

Use Case: Ștergerea de înregistrări din baza de date

Actor principal: Utilizatorul

Scenariu în caz de success:

1. S-a ales o înregistrare din tabelă.
2. Se va șterge din tabelă înregistrarea respective.

Scenariu în caz de eșec:

1. Nu s-a ales o înregistrare corespunzătoare
2. Va apărea o fereastră Popup cu un mesaj corespunzător.

Use Case: Plasarea de noi comenzi pentru un anumit client și un anumit produs

Actor principal: Utilizatorul

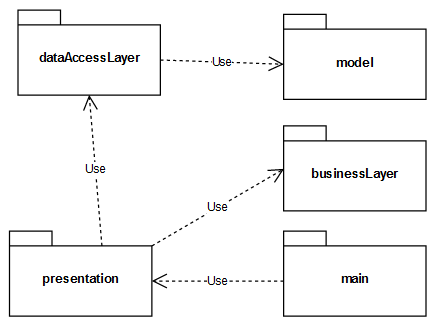
Scenariu în caz de success:

1. S-a ales un client din baza de date
2. S-a ales un produs din baza de date
3. S-a introdus o cantitate mai mica sau egală cu cantitatea produsului ales
4. Se va introduce o nouă înregistrare în tabela pentru comenzi și se va creea un fișier text pentru chitanța comenzii.

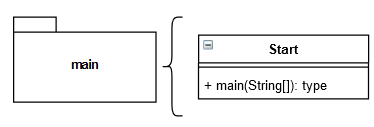
Scenariu în caz de eșec:

1. Nu s-a ales un client sau un produs din tabelă sau s-a introdus o cantitate necorespunzătoare
2. Va apărea o fereastră Popup cu un mesaj corespunzător.

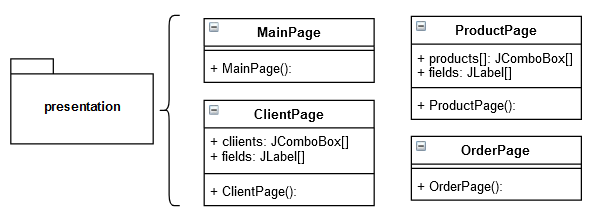
**3.Proiectare:**



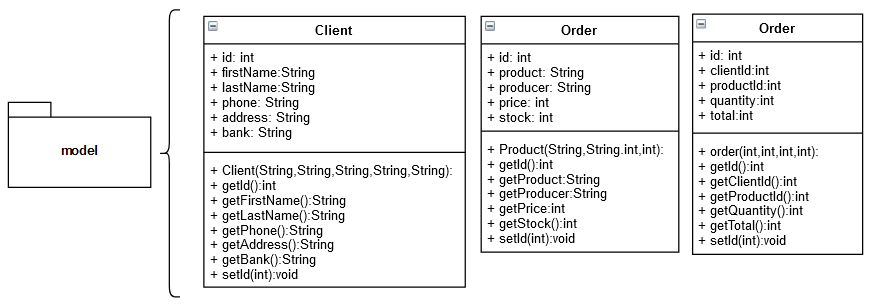
Programul va conține 5 pachete: main de un se va începe rularea programului, presentation ce conține clasele pentru interfața grafică, dataAccesLayer corespunzător claselor pentru accesarea datelor din baza de date, businessLayer ce conține o clasă cu metode statice de verificarea a corectitudinii informațiilor introduse și model ce conține clase corespunzătoare modelelor de date folosite în program.



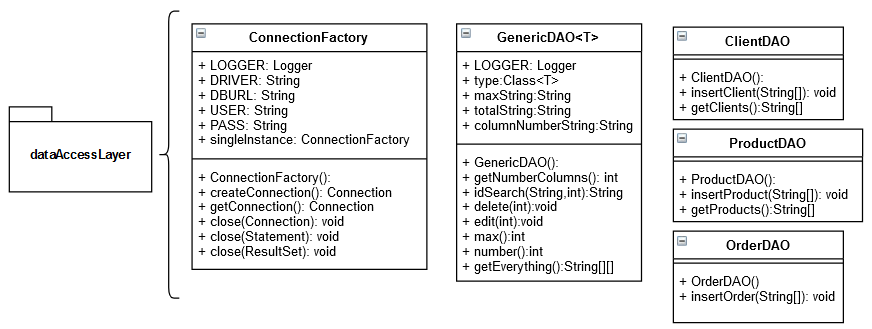
Pachetul main conține o singură clasă Start ce are o metodă de tip main prin care se va începe rularea programului, adică deschiderea unei ferestre MainPage.



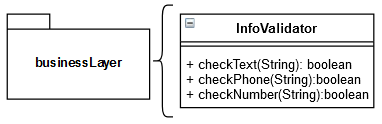
Pachetul presentation conține 4 clase ce extind Jframe si care vor reprezenta GUI-ul pentru introducerea datelor și apelarea metodelor de accesare a bazei de date. ClientPage si ProductPage sunt aproape identice la înfățișare, dar apelează metode diferite.



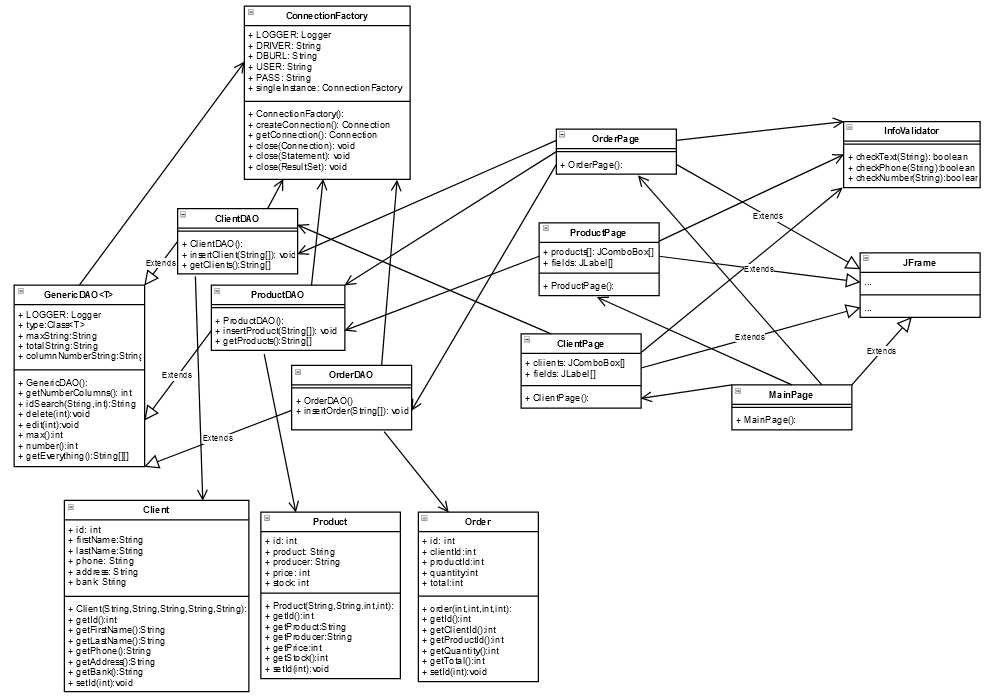
Pachetul model conține toate clasele folosite pentru reprezentarea modelelor de date. Toate aceste clase vor atribute corespunzătoare coloanelor din tabela reprezentantă și vor folosii doar metode de tip getter și setter.



Pachetul dataAccesLayer are clase al căror rol este de-a accesa baza de date. ConnectionFactory are rolul de-a stabilii și oprii conecxiunea cu baza de date, iar clasele GenericDAO<T> și subclasele acesteia vor avea metode ce vor apela interogări și operații de update, inserti și delete în fiecare tabelă din baza de date.



Pachetul businessLayer conține o singură clasă InfoValidator ce conține diferite metode statice folosite pentru verificarea corectitudinii unor obiecte de tip String transmise ca și parametrii. Toate acestea vor returna o valoare de tip boolean dacă String-ul transmis a fost corect redactat sau nu.



Din diagrama UML se observă mai multe relații de descendență între anumite clase. Clasa GenericDAO<T> este extinsă în clasele ClientDAO, ProductDAO și OrderDAO, fiecare dintre acestea prezentând metode specializatte pentru modelul de dat reprezentat. Toate clasele din pachetul presentation extind clasa JFrame, ceea ce permite scrierea mai ușoară a datelor din acestea. Restul legăturilor sunt asocieri simple între clase. În metoda main a clasei Start este declarată un obiect din clasa MainPage din care se vor putea accesa toate funcționalitățile programului.

**4.Implementare:**

În MySQL este creată o nouă bază de date numită OrderManagement care va conține 3 tabele: Clients în care sunt stocați clienții, Products în care sunt stocate produsele și Orders unde sunt stocate comenzile clienților.

Clasele Client, Product și Ordeer au rolul de reprezentanre pentru modelele de date folosite în program, ele având atribute corespunzătoare coloanelor din tabelele reprezentate. Clasa Client reprezintă un element din tabela ”Clients” din baza de date, aceasta având ca li atribute id, care este identificatorul individului respectiv, firstName, adică prenumele clientului, lastName, care este numele de familie, phone, care este numărul de telefon, address, care este adresa clientului, și bank, ce reprezintă banca unde își are cont clientul. Atributele bank, phone și address n-au funcționalitate pentru program, acestea având rol de informație suplimentară pentru chitanță. Această clasă prezintă metode de tip getter pentru fiecare atribut și o metodă de tip setter pentru atributul id.

Clasa Product reprezintă un element din tabela ”Products”. Acesta are ca și atribute id, cu rol de identificare, product, care este numele produsului, producer, adică numele producătorului, stock, care reprezintă cantitatea totală disponibilă pentru vânzare din acel produs, și price, adică prețul unui singur produs. Această clasă prezintă metode de tio getter pentru fiecare atribut și o metodă de tip setter pentru atributul id.

Clasa Order are rolul de-a reprezenta un element din tabela ”Orders”. Atributele acestei clase sunt id, cu rol de identificare, clientId, adică id-ul clientului pentru care s-a depus comanda, productId, adică id-ul produsului comandat, quantity, cantiatea de produs, și total care este prețul total pentru comanda respectivă. La fel ca și la Client și Product, această clasă are ca numai metode getter pentru fiecare atribut și o metodă setter pentru id.

Clasa ConnectionFactory are rolul de-a stabilii conexiunea dintre program și baza de date. Ca și atribute aceasta are un obiect din clasa Logger, folosit în excepții de tip SQLException, și mai multe String-uri, DRIVER, DBURL, USER, PASS, folosite în realizarea conexiunii. Clasa mai are deasemenea o instanță declarată în aceasta. Construcul clasei apelează metoda Class.forName pentru încărcarea clasei de tip Driver. Această clasă conține metoda createConnection pentru crearea unei conexiuni, getConnection pentru returnarea unei conexiuni create, și 3 metode supraîncărcate close care pot să închidă o conexiune, un obiect din clasa Statement sau un obiect din clasa ResultSet.

Pentru realizarea unor interogări și operații MySQL pe caz general se folosește clasa GenericDAO<T>, unde T poate fi oricare dintre clasele Client, Product sau Order. Aceasă clasă are ca și atribute un obiect din clasa Logger, cu același rol ca și cel din clasa ConnectionFactory, type ce are rolul de reprezentant al clasei desemnate de T și 3 String-uri maxString, totalString și columnNumberString ce reprezintă interogări din MySQL. Constructorul clasei va stabilii în type clasa T. Această clasă va conține metode care se pot aplica pentru oricare dintre modelele de date. Metoda getNumberColumns va apela o interogare prin care se va returna numărul total de coloane al tabelei reprezentate de clasa T. Metoda idSearch primește ca și parametrii un String, ce reprezintă numele unei coloane, și un int numit id care va reprezenta id-ul unei tuple, și va apela o interogare ce va returna ceea ce este stocat pe coloana column pe tupla id-ului. Metoda max va apela o interogare prin care se află valoarea id-ului maxim din tabelă. Metoda numbers() va apela o interogare prin care este returnat numărul total de tuple din tabelă, număr pe care îl returnează. Metoda getEverything este folosită pentru a apela o interogare ce returnează toată informația stocată în tabelă și o returnează sub forma unei matrici de String-uri.Metoda edit primește ca și parametrii o valoarea întreagă care este id-ul tuplei ce va fi modificate, un String column ce reprezintă coloană ce va fi modificată și un String newValue ce reprezintă noua informație ce va fi scrisă. Este apelată o operație de tip update ce va seta pe coloana column a tuplei corespunzatoare id-ului informația desemnata de newValue. Metoda delete are un singur parametru id, aceasta apelând operatia delete pentru a șterge tupla desemnată de id din tabelă.

Pentru interogări și operații mai specifice se folosesc clasele ClientDAO, ProductDAO și OrderDAO. Fiecare dintre acestea conțin câte o metodă add cu nume reprezentative pentru fiecare model de dată, precum addClient, ce vor primii un array de String-uri ce reprezintă atributele noului obiect, creează o instanță a modelului de dată corespunzătoare clasei și introduce prin intermediul claselor getter informația necesară pentru operația de add. ClientDAO mai prezintă deasemenea o metodă getClients ce va apela o interogare pentru numele și prenumele fiecărui client și returnează un array de String-uri cu numele complete ale clienților. Clasa ProductDAO mai prezintă o metodă getProducts care, similar cu getClients de la ClientDAO, returnează un array de String-uri cu numele fiecărui produs.

Pentru realizarea GUI-ului programului se folosesc clasele MainPage,ClientPage,ProductPage și OrderPage, acestea îndeplinind și rolul de preluare a datelor de intrare pentru anumite operații pe baza de date. Clasa MainPage este foarte simplu realizată, aceasta prezentând o fereastră cu 3 butoane: Clients, Product și Order. Fiecare buton este folosit pentru deschidera uneia dintre celelalte ferestre.

Clasa InfoValidator conține 3 metode statice folosite pentru verificarea corectitudinii datelor de intrare. Fiecare dintre acestea vor returna true dacă String-ul trimis respectă condițiile scrise în metodă. Metoda checkNumbers va returna false dacă String-ul are cel puțin un caracter care să nu fie cifră. Metoda checkText va returna false dacă String-ul conține cel puțin un caracter care să nu fie literă, spațiu sau cratimă. Metoda checkPhone se folosește de biblioteca regex și returnează false dacă String-ul nu respectă regula unui număr de telefon.

Clasa ClientPage și ProductPage sunt foarte similare în structură și funcționalitate. Fiecare conține mai multe casete text, prin care sunt scrise datele pentru o nouă tuplă sau pentru modificare a uneia, un JComboBox cu numele fiecărui produs sau client, și 5 butoane: un buton pentru adăugare, buton pentru editare, buton pentru ștergere, buton pentru vizualizare și un buton Back. Butonul Back este folosit pentru revenirea la fereatra MainPage. Butoanele pentru adăugare vor prelua datele din casetele text, le va verifica cu metodele din clasa InfoValidator și, dacă toate sunt corecte, va adăuga datele în tabelele corespunzătoare și casetele text vor fi golite. Butoanele de editare vor face ca datele din casetele text să fie preluate, verificate cu metodele din InfoValidator și, pentru datele care sunt corect formatate, se vor realiza modificări pentru clientul sau produsul selectați în JComboBox și se vor golii casetele text. Pentru butonul de ștergere se va verifica dacă este selectat în JComboBox un client sau un produs și acestea vor fi șterse din baza de date. Butonul pentru vizualizare va apela pentru fiecare client sau produs metoda getEverything din GenericDAO<T> pentru a a obține toate infiormațiile din baza de date și ale afișa într-o fereastră nouă într-un JTable.

Clasa OrderPage este făcută într-un mod mai simplu, aceasta fiind formată din două JComboBox-uri, o casetă text, un buton ”Back” și un buton ”Order”. Pentru punerea unei comenzi se alege un client din primul JComboBox, se alge un produs din al doilea JComboBox, se scrie o cantitate din produsul ales și se apasă butonul. Atunci când se selectează un produs, dacă stock-ul acestuia nu este 0 atunci se va afișa sub acesta mesajul ”in stock:” urmat de numărul total de produse, altfel se va scrie ”out of stock”. Dacă nu s-au selectat un client sau produs sau cantitatea scrisă în casetă este prea mare, atunci se va afișa o fereastră PopUp cu un mesaj corespunzător. Altfel se vor reseta JComboBox-urile, se va scrie în tabela Orders o nouă tuplă și se va creea un fișier text ce va reprezenta chitanța comenzii.

**5.Concluzii:**

Pentru realizarea acestei teme erau necesare cunoștiințe de bază legate de realizarea bazelor de date și de modul în care aceasta poate interacționa cu un program Java.

Această temă s-a dovedit a fii folositoare în reamintirea cunoștiințelor legate de creearea și interogarea bazelor de date și modul de accesare a acestora folosind un program extern. Acest lucru a fost practicat și în smestrul trecut pentru materia Baze de Date.

Îmbunătățiri ce ar putea fi aduse la program ar fii simplificarea metodelor de adăugare în baza de date, precum creearea unei metode add în clasa GenericDAO<T> ce poate fi accesată de oricare dintre clasele descendente. Altă îmbunătățire ar fi combinarea claselor ClientPage și ProductPage datorită similarității acestora.

**6.Bibliografie:**

Connect to MySql from a Java application:

<http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/>

Layered architectures:

<https://dzone.com/articles/layers-standard-enterprise>

Reflection:

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