



**TECHNICAL
UNIVERSITY**
OF CLUJ-NAPOCA
ROMANIA

FACULTATEA: Automatică și Calculatoare
SPECIALIZAREA: Calculatoare și Tehnologia Informației
DISCIPLINA: Tehnici de programare
Grupa 30226 | An 2 semestrul 2

Student:

Galiș George-Laurențiu



Cuprins

1. Obiectivul temei.....	3
2. Analiza problemei.....	3
3. Proiectare.....	4
4. Implementare.....	6
5. Rezultate.....	13
6. Concluzii.....	15
7. Webografie.....	15



1. Obiectivul temei

Obiectivul acestei teme pentru laborator este acela de a implementa un program care ajută la gestionarea produselor unui depozit sau magazin folosind o baza de date. Pe langa programarea orientata pe obiecte invatata pana acum, mai apare si notiunea de baza de date. Aceasta are la baza manipularea unor tabele in care sunt introduse diferite date.

2. Analiza problemei

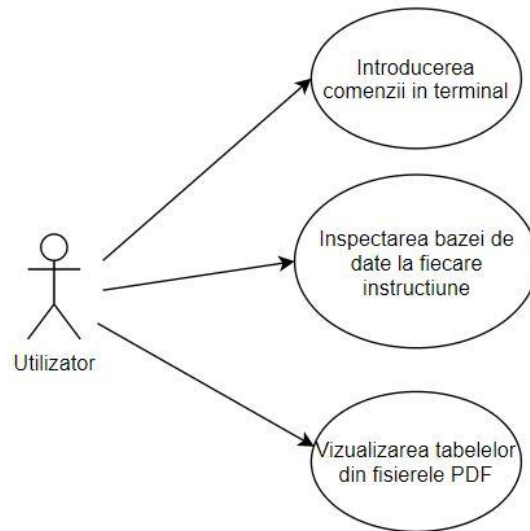
Aceasta tema poate fi folosita foarte bine si in lumea reala deoarece usureaza o munca suplimentara in ceea ce priveste administrarea unor produse. Cand vine vorba de o institutie, tot ce tine cont de aceasta(angajati, clienti, produse, accesorii) pot fi introduse si memorate intr-o baza de date deoarece acestea sunt foarte accesibile si usor de folosit.

Crearea unui astfel de aplicatie nu e foarte dificila dar, necesita niste cunostinte minime despre o baza de date deoarece aceasta notiune e la baza acestei probleme

Mai jos este reprezentată diagramă use-case care ajută la înțelegerea folosirii programului și opțiunile pe care le are de ales utilizatorul. Mai întâi utilizatorul trebuie să introducă comanda pentru rularea fisierului .jar in terminal, mai exact: `java -jar` dupa care numele fisierului executabil si fisierul text(in-test-1.txt). Comanda ar arata in felul urmator:

```
java -jar PT2020_30226_Galis_Laurentiu_Assignment_3.jar commands.txt
```

Dupa care, programul se va executa. In acest timp, utilizatorul poate inspecta baza de date si sa observe modificarile facute pe aceasta(INSERT, DELETE, UPDATE)



3. Proiectare

Cand vine vorba de implementarea propriu-zisa a acestui program, trebuie sa se tina cont de mai multi factori si sa punem urmatoarele intrebari: Ce tabele trebuiesc create pentru baza de date?, Ce tipuri de date se vor introduce in tabele?, Ce interogari vor avea loc asupra bazei de date?, Cum poate fi implementat in Programarea Orientata pe Obiect?.

Implementarea acestei teme are la baza folosirea thread-urilor si a notiuni de baza de date O bază de date este o colecție organizată de informații sau de date structurate, stocate electronic într-un computer. O bază de date este controlată, de regulă, de un sistem de management al bazelor de date (DBSM). Cumulat, datele, DBMS și aplicațiile asociate reprezintă un sistem de baze de date, denumit prescurtat bază de date. Datele din cele mai obișnuite tipuri de baze de date sunt distribuite de regulă pe linii și coloane, în diferite tabele, pentru eficientizarea procesării și interogării datelor. Datele pot fi accesate, gestionate, modificate, actualizate, controlate și organizate cu ușurință. Majoritatea bazelor de date utilizează un limbaj structurat de interogare (SQL) pentru scrierea și interogarea datelor.



La proiectarea acestei probleme m-am folosit de tehnica „Layered Architecture”. Aceasta face mult mai usor de inteles si proiectat pasii pentru aplicatia in sine deoarece desparte problema in mai multe „straturi”. Am proiectat aceste „straturi” sub forma de pachete .Eu m-am folosit de urmatoarele:

- businessLayer in care am pus clasa MainClass in care are loc rulara aplicatiei si Operations care se foloseste de baza de date si executa operatiile citite asupra bazei de date

- dataAccessLayer in care am pus clasa ConnectionFactory care face conexiunea aplicatiei la baza de date. In aceasta clasa am avut nevoie de pachetul java.sql deoarece acolo se gasesc clasele si metodele care ma ajuta sa folosesc notiunile bazelor de date

- modelLayer in care am pus clasele corespunzatoare fiecarui tabel din baza de date. Acest pachet ma ajuta sa pot manipula tabelele si datele din fiecare in programul Java.

- presentationLayer in care am pus doua clase: PDFgenerator care genereaza fisiere PDF pentru o instructiuni citita, de ex: Report Client sau Report Product. Aceasta clasa se foloseste foarte mult de libraria Itext care genereaza fisiere pdf sau documente in lumbajul Java cu metodele specifice. Pe langa aceasta clasa se afla si clasa ReadFromFile care face citirea din fisierul .txt. in constructorul acestei clase se verifica fiecare rand din fisierul text si face apel la metoda din clasa Operations specifica instructiunii citite.

Citirea din fisier si face cuajutorul clasei File si Scanner si a metodel din aceste clase cum ar fi getAbsolutePath() sau hasNextLine().

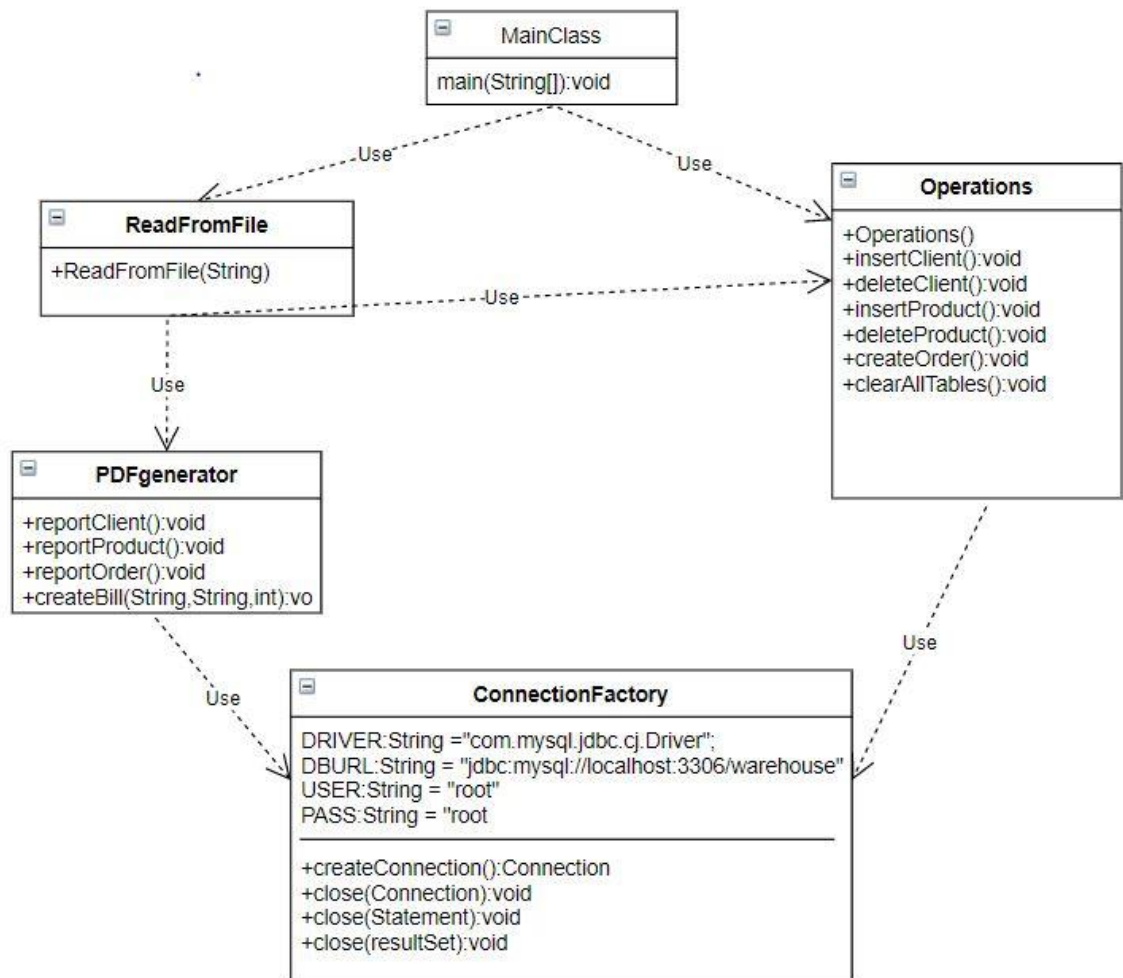
Datele din fierul .txt sunt extrase cu ajutorul clasei Scanner si metodele din acestea returneaza date de tip String. Asa ca am folosit metodele clasei Integer pentru a transforma sirurile de caractere extrase in numere intregi, mai exact pretul si cantitatea: Integer.parseInt(a), unde ,a’ este o linie din fisierul .txt.

• Diagrama UML



Tema 1

Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora (numite și obiecte). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT.





4. Implementare

- **Pachetele:**

Pachetele din aceasta clasa sunt: buisnessLayer, dataAccessLayer, modelLayer, presentationLayer. Scopul fiecarui pachet l-am explicat mai sus. Acestea sunt formate pentru a impartii aplicatia in mai multe ramuri, fiecare cu un scop bine definit

- **Clasele:**

Clasa MainClass:

Este clasa în care se implementează apeleaza clasa Controller in care are loc executia efectiva a programului. Functia ,public static void main' are ca argument un vectori de tipul String(String[] args) unde args sunt argumentele ce vor fi citite de exemplu din terminal args[1] va fi numele fisierului care va fi executat si args[2] va fi numele fisierului din care se vor extrage datele(in-test-1.txt)

Clasa Operations:

Aceasta clasa este formata dn metode care fac conectarea la baza de date si executa operatii asupra tabelelor din acestea.

Clasa ConnectionFactory:

Aceasta clasa are metode ce fac posibila conectarea la baza de date, inchiderea ei, a unei instructiuni in SQL si a unui set de rezultate din baza de date

Clasa Client, Product si Order:



Aceste clase reprezinta fiecare tabela din baza de date cu coloanele specifice fiecaruia: Client cu nume si adresa; Product cu nume, cantitate si pret; Order cu nume client, nume produs, cantitate. In aceasta clasa se afla getters si setters pentru variabile

Clasa PDFgenerator:

Aceasta clasa contine metode care genereaza fisiere PDF pentru fiecare instructiune citita din fisier: Raport Client, Raport Product, Raport Order si createBill. Fiecare metoda genereaza cate un fisier sPDF eparat pentru fiecare instructiune in care se afla un tabel ce contine elementele bazei de date.

Clasa ReadFromFile:

Aceasta clasa ajuta la citirea si transformarea datelor din fisierul .txt. transformarea se face cu ajutorul metodelor parseInt din clasa Integer. Pe langa acest lucru, aceasta clasa foloseste clasa Operations pentru a face operatii pe baza de date conform instructiunii citite din fisierul.txt

- **Metodele utilizate în clasa ReadFromFile:**



Tema 1

Aceasta metoda este formata din getters and setters pentru variabilele care se vor citi din fisier. Citirea are loc in constructorul clasei:

```
public ReadFromFile(String inputFile) {  
    try {  
        File f = new File(inputFile);  
        String path = new String();  
        if (f.exists()) {  
            path = f.getAbsolutePath();  
            System.out.println("File path: " + path + "\n");  
        }  
        File file = new File(path);  
        Scanner s = new Scanner(file);  
        while (s.hasNextLine()) {  
            String a = s.nextLine();  
            if (a.equals("Report client")) {  
                PDFgenerator.reportClient();  
                continue;  
            }  
            if (a.equals("Report product")) {  
                PDFgenerator.reportProduct();  
                continue;  
            }  
        }  
    }  
}
```

... (cate un if pentru fiecare instructiune citita din
fisiere)



- **Metodele utilizate în clasa Client, Product si Order:**

Aceasta clasa este foemata doar din getters and setters pentru tabelele Client, Product si Orders. De ex pentru Client:

```
public String getNameC() {  
    return nameC;  
}
```

```
public void setNameC(String nameC) {  
    this.nameC = nameC;  
}
```

```
public String getAddress() {  
    return address;  
}
```

```
public void setAddress(String address) {  
    this.address = address;  
}
```



- **Metodele utilizate în clasa Operations:**

Mai jos se poate observa una din operatiile ce au loc asupra bazei de date”

```
public static void insertClient(String name, String address) {  
    System.out.println("Inserting Client: " + name);  
    Connection db = ConnectionFactory.createConnection();  
    final String statementString = "INSERT INTO client  
VALUES('" + name + "', '" + address + "')";  
    try {  
        Statement st = db.createStatement();  
        st.executeUpdate(statementString);  
        db.close();  
    } catch (SQLException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
    ...  
}
```



- **Metodele utilizate în clasa ConnectionFactory:**

Pe langa metodele ce incid conexiunea la baza de date se afla si metoda ce face legatura la aceasta

```
public static Connection createConnection() {  
    Connection conn = null;  
    try {  
        conn = DriverManager.getConnection(DBURL, USER,  
PASS);  
        if (conn == null) {  
            System.out.println("Connect ERROR!!!");  
        }  
    } catch (SQLException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
    return conn;  
}
```



- **Metodele utilizate în clasa PDFgenerator:**

Mai jos am luat exemplul de metoda in care se creeaza un fisier PDF care afiseaza o chitanta pentru o anumita comanda. In acest fiser se poate vedea numele clientului, produsul cumparat si pretul total in functie de cantitatea cumparata a produsului sespectiv

```
public static void createBill(String nameC, String nameP, int quantity) {  
    Document document = new Document();  
    double totalPrice = 0;  
    Connection db = ConnectionFactory.createConnection();  
    final String statement = "SELECT * FROM product WHERE  
nameP='" + nameP + "';";  
    ResultSet rs = null;  
    Statement st;  
    try {  
        st = db.prepareStatement(statement);  
        rs = st.executeQuery(statement);  
        rs.next();  
        double price = Double.parseDouble(rs.getString("price"));  
        totalPrice = price * (double) quantity;  
    } catch (SQLException e1) {  
        // TODO Auto-generated catch block  
        e1.printStackTrace();  
    }  
  
    try {  
        PdfWriter.getInstance(document, new FileOutputStream("bill"  
+ billCount + ".pdf"));
```



```
        document.open();

        Font font = FontFactory.getFont(FontFactory.COURIER, 16,
BaseColor.BLACK);

        Chunk chunk = new Chunk("Client: " + nameC + "\n" +
"Produs: " + nameP + "\nCantitate: " + quantity, font);

        chunk.setLineHeight(50);

        document.add(chunk);

        font = FontFactory.getFont(FontFactory.COURIER, 24,
BaseColor.BLACK);

        Chunk chunk1 = new Chunk("\n\n\nPret total: " + totalPrice,
font);

        chunk1.setLineHeight(50);

        document.add(chunk1);

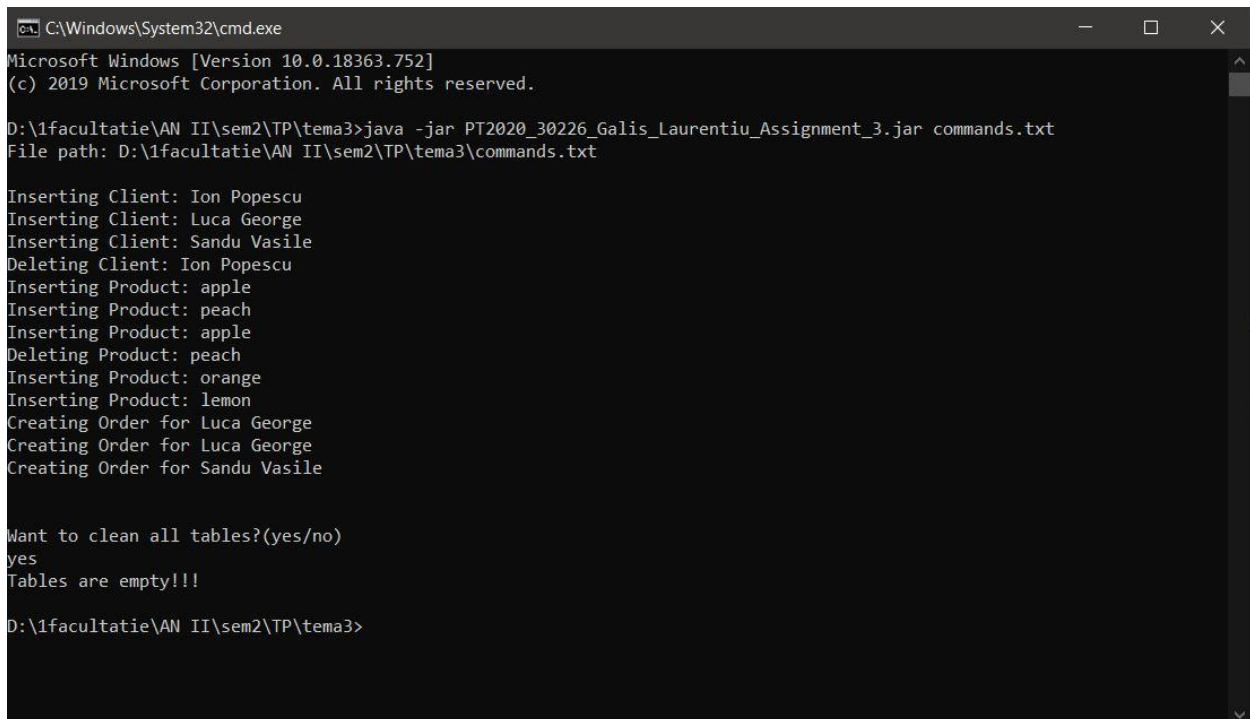
        document.close();

    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (DocumentException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```



5. Rezultate

Afisarea este facuta in consola sau in terminal cu ajutorul metodei din clasa `System.out.println()`. Dupa ce se introduce comanda de rulare, se va afisa fiecare comanda ce va avea loc. La finalul executiei comenzilor se va afisa mesajul „Want to clean all tables?” si in functie de alegerea utilizatorului programul va ssterge sau nu toate datele din tabele.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.752]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\1facultatie\AN II\sem2\TP\tema3>java -jar PT2020_30226_Galis_Laurentiu_Assignment_3.jar commands.txt
File path: D:\1facultatie\AN II\sem2\TP\tema3\commands.txt

Inserting Client: Ion Popescu
Inserting Client: Luca George
Inserting Client: Sandu Vasile
Deleting Client: Ion Popescu
Inserting Product: apple
Inserting Product: peach
Inserting Product: apple
Deleting Product: peach
Inserting Product: orange
Inserting Product: lemon
Creating Order for Luca George
Creating Order for Luca George
Creating Order for Sandu Vasile

Want to clean all tables?(yes/no)
yes
Tables are empty!!!

D:\1facultatie\AN II\sem2\TP\tema3>
```

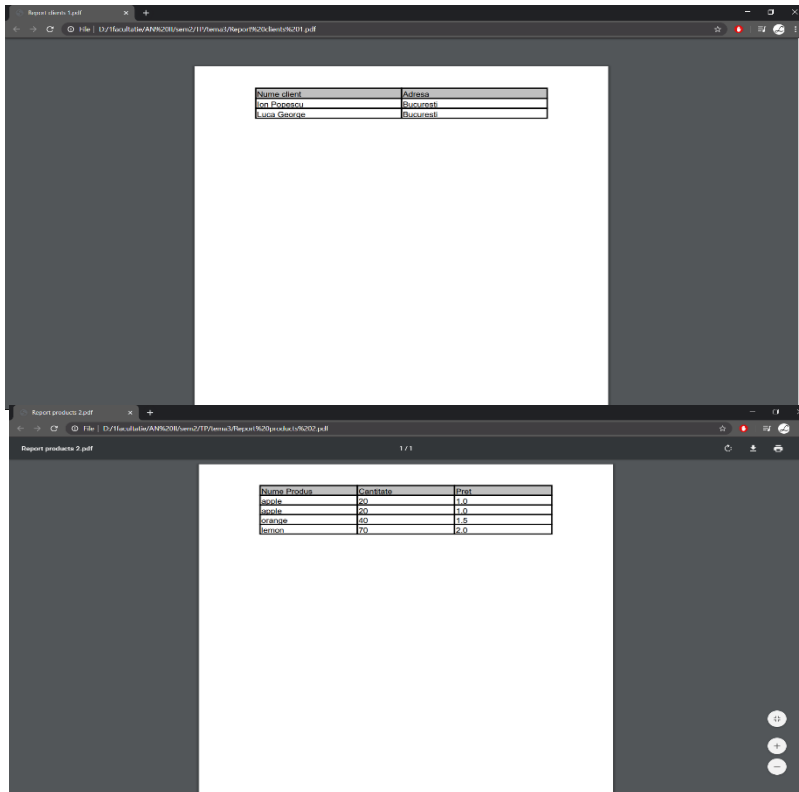


Tema 1

Dupa rulare programului, in directorul programului vor fi genera te fisierele PDFnumerate in functie de ordinea executiei:

bill1	4/16/2020 11:49 PM	Chrome HTML Do...	1 KB
bill2	4/16/2020 11:49 PM	Chrome HTML Do...	1 KB
bill3	4/16/2020 11:49 PM	Chrome HTML Do...	1 KB
Report clients 1	4/16/2020 11:49 PM	Chrome HTML Do...	2 KB
Report clients 2	4/16/2020 11:49 PM	Chrome HTML Do...	2 KB
Report clients 3	4/16/2020 11:49 PM	Chrome HTML Do...	2 KB
Report clients 4	4/16/2020 11:49 PM	Chrome HTML Do...	2 KB
Report orders 3	4/16/2020 11:49 PM	Chrome HTML Do...	2 KB
Report products 1	4/16/2020 11:49 PM	Chrome HTML Do...	2 KB
Report products 2	4/16/2020 11:49 PM	Chrome HTML Do...	2 KB
Report products 3	4/16/2020 11:49 PM	Chrome HTML Do...	2 KB

Fisierele PDF generate au urmatoarea forma:





6. Concluzie

În concluzie, acest proiect a avut scopul de a aprofunda cunoștințele în tot ce înseamnă limbajul Java, implementarea paradigmatelor OOP, folosirea și înțelegerea bazelor de date și citirea din fișier. Mai mult de atât cred că a avut și scopul de a reaminti tehnicile de programare învățate semestrul trecut despre tot ce înseamnă sintaxa și aranjarea unui cod ca să poată fi înțeles foarte ușor și de un alt programator.

7. Webografie

1. <http://youtube.com>
2. <https://www.wikipedia.org/>
3. <https://www.w3schools.com/>
4. <https://www.geeksforgeeks.org/>