Documentatie Tema 3

-ORDER MANAGEMENT-

Nume: Hilbert Dennis

Seria: B, Grupa: 30226 An 2, CTI

Cuprins

Cuprins. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .2

Obiectivul temei . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

Proiectare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

-Descrierea generala a proiectului . . . . . . . . . . . . . . . . . . . 3

-Descrierea claselor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

-Pachetul connection. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4

-Pachetul presentation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4

-Pachetul model. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

-Pachetul main. . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .6

Implementare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .8

Rezultate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

Concluzii . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

Bibliografie . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .10

1. Obiectivul temei

Obiectivul temei consta in implementarea unei aplicatii care realizeaza conexiunea cu o baza de date prin intermediul unui program java.

Aplicatia preia comenzi dintr-un fisier primit ca argument, realizeaza operatiile cerute, salveaza date in baza de date si genereaza rapoarte in format pdf.

Obiectivele secundare sunt reprezentate de crearea si structurarea intr-un mod avantajos a pachetelor, claselor si functiilor care le compun. Acestea ar fi, in ordine, urmatoarele:

-crearea functiilor care transforma datele de tip String in interogari care sunt rulate in MySql

-prelucrarea datelor prin intermediul functiilor care determina carei clase ii este destinata comanda la un anumite moment(client, product, order)

-crearea functiilor care determina daca este vorba de o opratie de insert, report sau delete(+comenzile care creeaza comenzi)

-utilizarea javaDoc

-crearea unui fisier .jar cu care aplicatia va putea fi rulata

-generarea fisierelor de tip SQL dump file

Acestea urmeaza sa fie detaliate in capitolul 3 care descrie in detaliu proiectarea aplicatiei.

2.Proiectare

2.1Descrierea generala a proiectului:

Din punct de vedere structural, proiectul contine patru pachete si anume main, connection, model si presentation.

In pachetul model se afla clasele Model, Client, Product si Order. Acestea contin metode care se ocupa in mare parte de executia programului, mai exact de operatiile pe date. Comenzile citite in presentation sunt preluate prin intermediul claselor acestui pachet si sunt prelucrate pentru a furniza rezultatele necesare.

In pachetul presentation se afla clasa Presentation care se ocupa de citirea comenzilor din fisierul extern comenzi.txt si le transmite mai departe in program pentru a fi executate.

In pachetul connection se afla clasa Connection care contine metode necesare realizarii conexiunii cu baza de date(deschiderii si inchiderii acesteia)

In pachetul Main se afla clasa Main care contine un atribut ce retine numarul de ordine al PDF-ului ce urmeaza sa fie generat si metoda Main care controleaza actiunile programului.

2.2 Descrierea claselor:

**Pachetul connection:**

-Clasa Connection

Atribute:

public static final Logger *LOGGER* = Logger.*getLogger*(ConnectionControl.class.getName());  
public static final String *DRIVER* = "com.mysql.cj.jdbc.Driver";  
public static final String *DBURL* = "jdbc:mysql://localhost:3306/warehouse";  
public static final String *USER* = "root";  
public static final String *PASS* = "rootpass";

* Constructori:
  + public Connection()
* Metode:

public Connection getConnection()   
 public static void closeStatement(Statement statement) throws SQLException

**Pachetul presentation:**

-Clasa Presentation

* Metode

public static void executeFromInputFile(Connection connection, String path)   
 private void addTableHeader(PdfPTable table, int nr, String s)   
   
private void addRows(PdfPTable table, ResultSet rs, String s) throws SQLException   
public void createDocument(ResultSet rs, int nr, String s) throws FileNotFoundException, DocumentException, SQLException  
public void generateBill(Connection con, String[] str) throws FileNotFoundException, DocumentException  
   
 public void generateUnderStockPDF(String[] str) throws FileNotFoundException, DocumentException

* Constructori:
* public Presentation()

**Pachetul model:**

-Clasa Model

* Constructori:
* public Model()
* Metode:

public void executeOperation(Connection con, String s) throws SQLException, FileNotFoundException, DocumentException

-Clasa Client

* Constructori:
* public Client()

public static void detOperation(Connection con, String[] str) throws SQLException   
public static void viewTable(Connection con) throws SQLException

public static void insertClient(Connection con, String q) throws SQLException

public static void deleteClient(Connection con, String q) throws SQLException

-Clasa Order

* Constructori:
* public Order()
* Metode:

public static void detOperation(Connection con, String[] str) throws SQLException, FileNotFoundException, DocumentException

public static void viewTable(Connection con) throws SQLException

public static void insertOrder(Connection con, String q) throws SQLException

private static int checkValidity(Connection con, String[] str) throws SQLException

-Clasa Product

* Constructori:
* public Product()
* Metode:

public static void detOperation(Connection con, String[] str) throws SQLException

public static void viewTable(Connection con) throws SQLException

public static void insertProduct(Connection con, String q) throws SQLException   
  
 public static void deleteProduct(Connection con, String q) throws SQLException   
  
 public static void updateStock(Connection con, String[] str, int newQuantity) throws SQLException

**Pachetul main:**

-Clasa Main

* Metode:

public static void main(String[] args) throws SQLException, FileNotFoundException, DocumentException

3.Implementare

-Clasa Presentation

Clasa presentation se ocupa de functionalitatile de input/output ale programului, mai exact, citeste datele din fisierul de comanda si dupa ce acestea sunt prelucrate de program, genereaza PDF-uri cu rezultatele.

In secventa de mai jos se citesc comenzile:

public static void executeFromInputFile(Connection connection, String path) {

try {  
 File myObj = new File(path);  
 Scanner myReader = new Scanner(myObj);  
 Model m = new Model();  
 while (myReader.hasNextLine()) {  
 String data = myReader.nextLine();  
 System.*out*.println(data);  
 m.executeOperation(connection, data);  
 }  
 myReader.close();  
 } catch (  
 FileNotFoundException | SQLException | DocumentException e) {  
 System.*out*.println("An error occurred.");  
 e.printStackTrace();  
 }  
}

Iar un exemplu de generare PDF este:

public void generateUnderStockPDF(String[] str) throws FileNotFoundException, DocumentException {  
 String message = "Nu sunt destule " + str[3] + " in stoc pentru comanda lui " + str[1] + " " + str[2];  
 Document document = new Document();  
 Paragraph paragraph = new Paragraph();  
 PdfWriter.*getInstance*(document, new FileOutputStream("PDF" + Main.*pdf\_nr* + ".pdf"));  
 Main.*pdf\_nr*++;  
 document.open();  
  
 Font font = FontFactory.*getFont*(FontFactory.*COURIER*, 16, BaseColor.*BLACK*);  
 Chunk chunk = new Chunk(message, font);  
 paragraph.add(chunk);  
 document.add(paragraph);  
 document.close();  
}

-Clasa Model

Clasa model este responsabila cu identificarea comenzilor. Aici se stabileste daca instructiunile ce urmeaza sa se execute tin de partea de client, product sau order, iar dupa ce acest lucru este stabilit se apeleaza metodele corespunzatoare ce continua executia in functie de domeniu.

public void executeOperation(Connection con, String s) throws SQLException, FileNotFoundException, DocumentException {  
 boolean client = false, product = false, order = false;  
 String[] str = s.split("(:)?(,)? ");  
  
 String q = "";  
 for(String val : str)  
 if(val.equals("model.Client") || val.equals("client"))  
 client = true;  
 else if(val.equals("model.Product") || val.equals("product"))  
 product = true;  
 else if(val.equals("model.Order") || val.equals("order"))  
 order = true;  
 if(client)  
 {  
 Client.*detOperation*(con, str);  
 }  
 else if(product)  
 {  
 Product.*detOperation*(con, str);  
 }  
 else if(order)  
 {  
 Order.*detOperation*(con, str);  
 }  
  
}

-Clasa Client

Clasa Client se ocupa de compunerea de query-uri pentru tabela clients din baza de date si efectueaza operatiile corespunzatoare(insert, report, delete).

O metoda importanta din clasa Client:

public static void detOperation(Connection con, String[] str) throws SQLException {  
 String q = "";  
 for (String val : str)  
 if (val.equals("Insert")) {  
 q += "insert into clients values(\"" + str[2] + " " + str[3] + "\", \"" + str[4] + "\")";  
 Client.*insertClient*(con, q);  
 }  
 else if(val.equals("Report"))  
 Client.*viewTable*(con);  
 else if(val.equals("Delete"))  
 {  
 q += "delete from clients where name = \"" + str[2] + " " + str[3] + "\" and address = \"" + str[4] + "\"";  
 Client.*deleteClient*(con, q);  
 }  
}

-Clasa Order

Clasa Order se ocupa de compunerea de query-uri pentru tabela orders din baza de dadte si efectueaza operatiile corespunzatoare(insert, report). In cazul in care comanda nu poate fi creata din cauza lipsei de produse, se va genera un PDF cu un mesaj de under stock. In cazul in care comanda poate fi stabilita, se va face un update in tabela de produse care are rolul de a modifica stock-ul disponibil in urma efectuarii comenzii. In cazul in care exista destule produse in stock si sunt indeplinite toate conditiile necesare, in final, comanda este intrudusa in baza de date si este creat un bon pentru client in format PDF.

O metoda importanta din clasa Order:

private static int checkValidity(Connection con, String[] str) throws SQLException {  
 int request = Integer.*parseInt*(str[4]);  
 for (String val : str) {  
 System.*out*.println(val);  
 }  
 int newQuantity = -1;  
 int quantity = 0;  
 Statement stmt = null;  
 String query = "select \* from products where name = \"" + str[3] + "\"";  
  
 try {  
 stmt = con.createStatement();  
 ResultSet rs = stmt.executeQuery(query);  
 while (rs.next()) {  
 quantity = rs.getInt("quantity");  
 }  
 } catch (SQLException e) {  
 System.*out*.println(e);  
 } finally {  
 if (stmt != null) {  
 ConnectionControl.*closeStatement*(stmt);  
 }  
 }  
 int stock = quantity;  
  
 if (request <= stock)  
 newQuantity = stock - request;  
 return newQuantity;  
}

-Clasa Product

Clasa Product se ocupa de compunerea de query-uri pentru tabela products din baza de date si efectueaza operatiile corespunzatoare(insert, report, delete).

O metoda importanta din clasa Product:

public static void updateStock(Connection con, String[] str, int newQuantity) throws SQLException {  
 Statement stmt = null;  
 String q = "update products set quantity = " + newQuantity + " where name = \"" + str[2] + "\"";  
 try {  
 stmt = con.createStatement();  
 stmt.executeUpdate(q);  
 } catch (SQLException e) {  
 System.*out*.println(e);  
 } finally {  
 if (stmt != null) {  
 ConnectionControl.*closeStatement*(stmt);  
 }  
 }  
  
}

-Clasa ConnectionControl

Clasa ConnectionControl este responsabila de stabilirea conexiunii cu baza de date(deschidere si inchidere).

O metoda importanta din clasa ConnectionControl:

public Connection getConnection() {  
 Connection c = null;  
 try{  
 Class.*forName*(*DRIVER*);  
 c = DriverManager.*getConnection*(*DBURL*, *USER*, *PASS*);  
 }  
 catch(Exception e){  
 System.*out*.println(e);  
 }  
  
 return c;  
}

-Clasa Main

In clasa Main este coordonata executia intregului program prin utilizarea tuturor functionalitatilor ale acestuia.

ConnectionControl c = new ConnectionControl();  
Connection connection = c.getConnection();  
  
String path = "commands.txt";  
Presentation.*executeFromInputFile*(connection, path);

4. Rezultate:

Rezultatele obtinute sunt ilustrate cel mai bine in fisierele de tip PDF generate in urma executiei programului.

5. Concluzii:

In concluzie, lucrul cu bazele de date prin intermediul unui program este foarte util si totodata accesibil utilizatorului. Exista multe functii cu diferite functionalitati care fac aceasta munca mai usoara uneori si se poate lucra cu acestea in diferite scopuri.

Astfel un program Java poate interactiona cu o baza de date fara probleme, stabilind o conexiune cu aceasta iar mai apoi executand comenzi de tip query pentru prelucrarea datelor.

Implementarea ar putea fi dezvoltata de exemplu pentru lucrul cu mai multe baze de date in acelasi timp.

6. Bibliografie:

• Connect to MySql from a Java application

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

o http://theopentutorials.com/tutorials/java/jdbc/jdbc-mysql-create-database-example/

• Layered architectures

o https://dzone.com/articles/layers-standard-enterprise

• Reflection in Java

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

• Creating PDF files in Java

o https://www.baeldung.com/java-pdf-creation

• JAVADOC

o https://www.baeldung.com/javadoc

• SQL dump file generation

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

o https://dev.mysql.com/doc/refman/5.7/en/using-mysqldump.html