Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

Центр программной инженерии

ОТЧЕТ

к лабораторной работе №6 по дисциплине:

|  |
| --- |
| «Объектно-ориентированное программирование» |
| «Организация работы с базой данных» |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | ИСТб-19-2 |  |  |  | Комогорцева Ю.В. |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил: | доцент |  |  |  | Маланова Т.В. |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск. 2020 г.

Содержание

[**1 Постановка задачи** 3](#_Toc70587827)

[**2 Описание БД 4**](#_Toc70587828)

[**3 Проектирование графического интерфейса 8**](#_Toc70587829)

[**4 Описание структуры классов 14**](#_Toc70587830)

[**5 Проектирование тестов 15**](#_Toc70587831)

[**6 Результаты тестирования 17**](#_Toc70587832)

[**7 Листинг программы 32**](#_Toc70587833)

[**7.1 класс Main 32**](#_Toc70587834)

[**7.2 класс MyFrame 33**](#_Toc70587835)

[**7.3 класс MyDialog 40**](#_Toc70587836)

[**7.4 класс MyTableModelProduct 47**](#_Toc70587837)

[**7.5 класс MyTableModelExistence 49**](#_Toc70587838)

[**7.6 класс MyTableModelSales 51**](#_Toc70587839)

[**7.7 класс MyTableModelDeliveries 53**](#_Toc70587840)

[**7.8 класс Product 55**](#_Toc70587841)

[**7.9 класс Sales 56**](#_Toc70587842)

[**7.10 класс Deliveries 57**](#_Toc70587843)

[**7.11 класс DBWorker 58**](#_Toc70587844)

[**Список использованных источников 66**](#_Toc70587845)

1 Постановка задачи

Цель работы: Применение объектно-ориентированного подхода к организации доступа к базе данных (БД).

Задачи:

1. Получение общего представления об объектном подходе к организации доступа к БД, выполнению запросов и обработке результатов.
2. Получение практического опыта организации работы с БД средствами платформы java.

Содержание задания

Для выполнения задания необходимо разработать java-приложение с графическим пользовательским интерфейсом позволяющее просматривать и редактировать информацию из базы данных. Работу предваряет организация и наполнение базы данных (БД) в соответствии с предметной областью из индивидуального задания. Графический пользовательский интерфейс должен предоставлять возможности работы с записями основных таблиц БД. Состав используемых компонентов определяется самостоятельно, но для их размещения должны быть предусмотрены соответствующие планировщики раскладки компонентов в основных контейнерах. Классы, реализующие графический пользовательский интерфейс, должны содержать только обращения к методам классов для выполнения заданий.

2 Описание БД

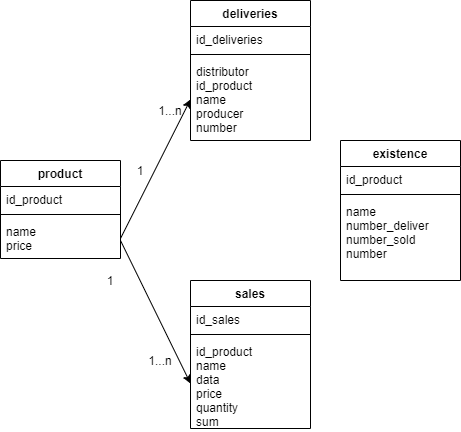


Рисунок 1 – Диаграмма классов БД

Таблица 1 – Описание таблиц БД

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя таблицы | Имя атрибута | Тип | Размер | Описание |
| product | id\_product | INTEGER | 8 байт | Идентификатор  продукта  Ключевое поле |
| name | TEXT | 256 байт | Название продукта |
| price | FLOAT | 4 байта | Стоимость за шт |
| sales | id\_sales | INTEGER | 8 байт | Идентификатор  продажи  Ключевое поле |
| id\_product | INTEGER | 8 байт | Идентификатор  продукта |
| name | TEXT | 256 байт | Название продукта |
| data | TEXT | 256 байт | Дата продажи |
| price | FLOAT |  | Стоимость за шт |
| quantity | INTEGER | 8 байт | Количество проданных |
| sum | FLOAT | 4 байта | Сумма за количество проданных |
| deliveries | id\_deliveries | INTEGER | 8 байт | Идентификатор  поставки  Ключевое поле |
| distributor | TEXT | 256 байт | Поставщик |
| id\_product | FLOAT | 4 байта | Идентификатор  продукта |
| name | TEXT | 256 байт | Название продукта |
| producer | TEXT | 256 байт | Производитель |
| number | INTEGER | 8 байт | Количество товара |
| existence | id\_product | INTEGER | 8 байт | Идентификатор  продукта  Ключевое поле |
| name | TEXT | 256 байт | Название продукта |
| number\_deliver | INTEGER | 8 байт | Количество поставленных |
| number\_sold | INTEGER | 8 байт | Количество проданных |
| number | INTEGER | 8 байт | Количество в наличии |

SQL-запросы к БД

CREATE TABLE if not exists 'product' ('id\_product' int PRIMARY KEY, 'name' text, 'price' float );

CREATE TABLE if not exists 'existence' ('id\_product' int PRIMARY KEY, 'name' text,'number\_deliver' int, 'number\_sold' int,'number' int );

CREATE TABLE if not exists 'sales' ( 'id\_sales' int PRIMARY KEY, 'id\_product' int, 'name' text, 'data' text, 'price' float, 'quantity' int , 'sum' float );

CREATE TABLE if not exists 'deliveries' ( 'id\_deliveries' int PRIMARY KEY, 'distributor' text , 'id\_product' int, 'name' text, 'producer' text , 'number' int );

INSERT INTO product ('id\_product','name', 'price') VALUES(?,?,?)

INSERT INTO existence ('id\_product','name', 'number\_deliver','number\_sold','number') VALUES(?,?,?,?,?)

INSERT INTO sales ('id\_sales','id\_product','name', 'data','price','quantity','sum') VALUES(?,?,?,?,?,?,?)

UPDATE existence SET 'number\_sold'=? WHERE id\_product =?

INSERT INTO deliveries ('id\_deliveries','distributor','id\_product','name','producer' ,'number') VALUES(?,?,?,?,?,?)

DELETE FROM product WHERE id\_product=?

DELETE FROM existence WHERE id\_product=?

DELETE FROM sales WHERE id\_product=?

DELETE FROM deliveries WHERE id\_product=?

DELETE FROM deliveries WHERE id\_deliveries=?

DELETE FROM sales WHERE id\_sales=?

DELETE From product

DELETE From existence

DELETE FROM sales

DELETE FROM deliveries

SELECT \* FROM product

SELECT \* FROM deliveries

SELECT \* FROM sales

SELECT \* FROM existence

SELECT id\_product,number FROM deliveries

SELECT id\_product,quantity FROM sales

SELECT id\_product,name FROM product

UPDATE existence SET number\_deliver=?,number\_sold=?,number=? WHERE id\_product =?

UPDATE product SET id\_product=? ,'name'=? ,'price'=? WHERE id\_product =?

UPDATE existence SET id\_product=? ,'name'=? WHERE id\_product =?

UPDATE deliveries SET id\_product=? ,'name'=? WHERE id\_product =?

UPDATE sales SET id\_product=? ,'name'=?,'sum'=? WHERE id\_product =?

UPDATE deliveries SET id\_deliveries=?,'distributor'=?,id\_product=? ,'name'=?,'producer'=? ,'number'=? WHERE id\_deliveries =?

UPDATE sales SET id\_sales=?,id\_product=? ,'name'=?,'data'=? ,'price'=?,'quantity'=?,'sum'=? WHERE id\_sales =?

**3 Проектирование графического интерфейса**

Описание главного окна

На главном окне с помощью компоновщика BorderLayout размещены : верхняя панель с картинкой (метод CreatePaneTitle()), центральная панель с элементом JTabbedPane (Data) и нижняя панель (метод CreatePaneLower()).

Центральная панель содержит элемент JTabbedPane, который состоит из 4 вкладок «Поставки», «Товары», «Продажи», «Наличие». Для каждой вкладки – своя панель с компоновщиком BorderLayout, где слева панель с кнопками ,а в центре таблица.

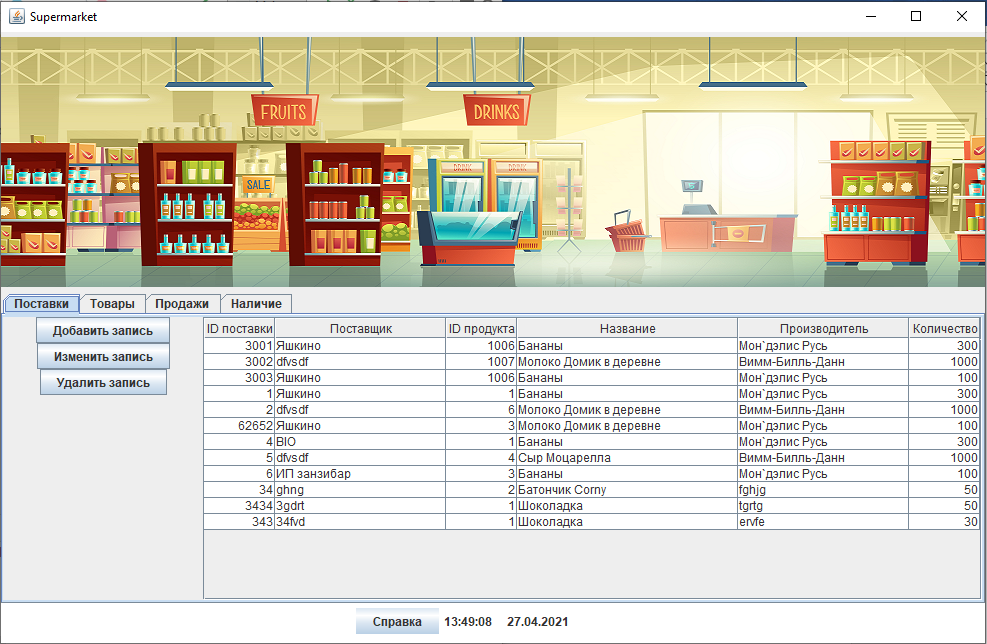


Рисунок 2 – Главное окно программы, вкладка «Поставки»

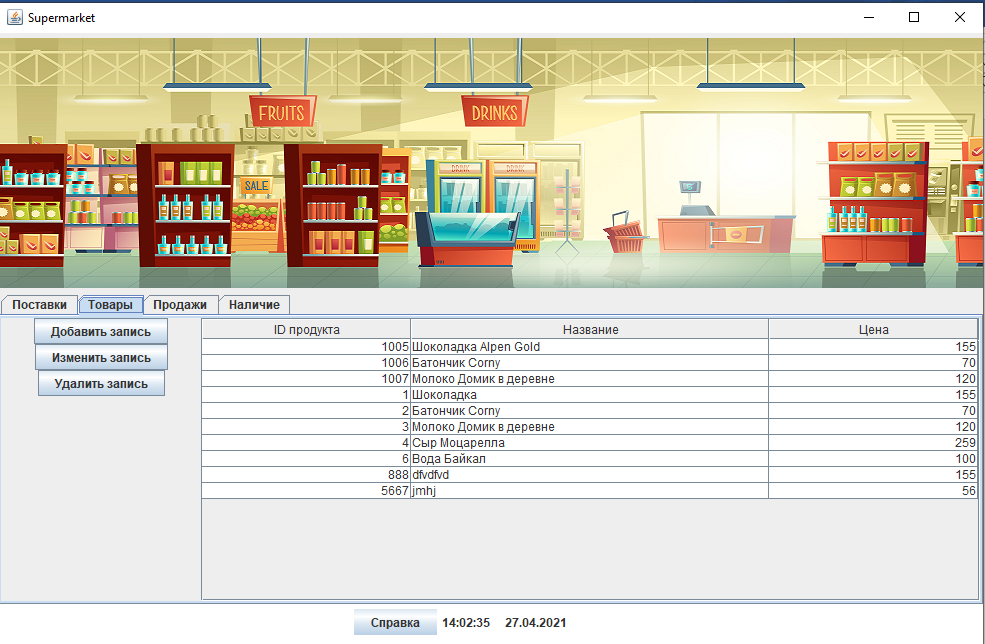


Рисунок 3 – Главное окно программы, вкладка «Товары»

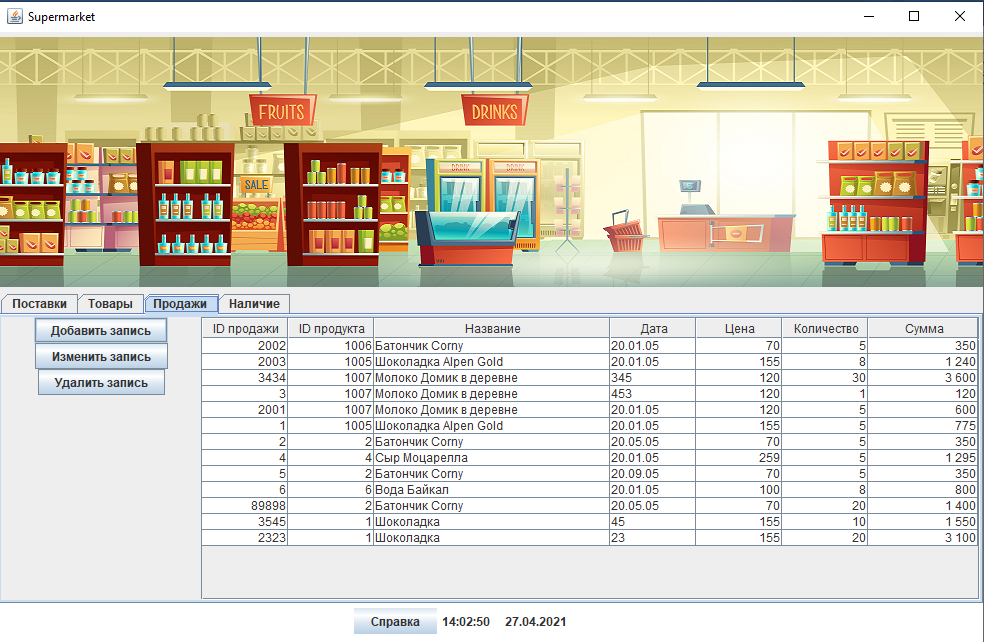


Рисунок 4 – Главное окно программы, вкладка «Продажи»

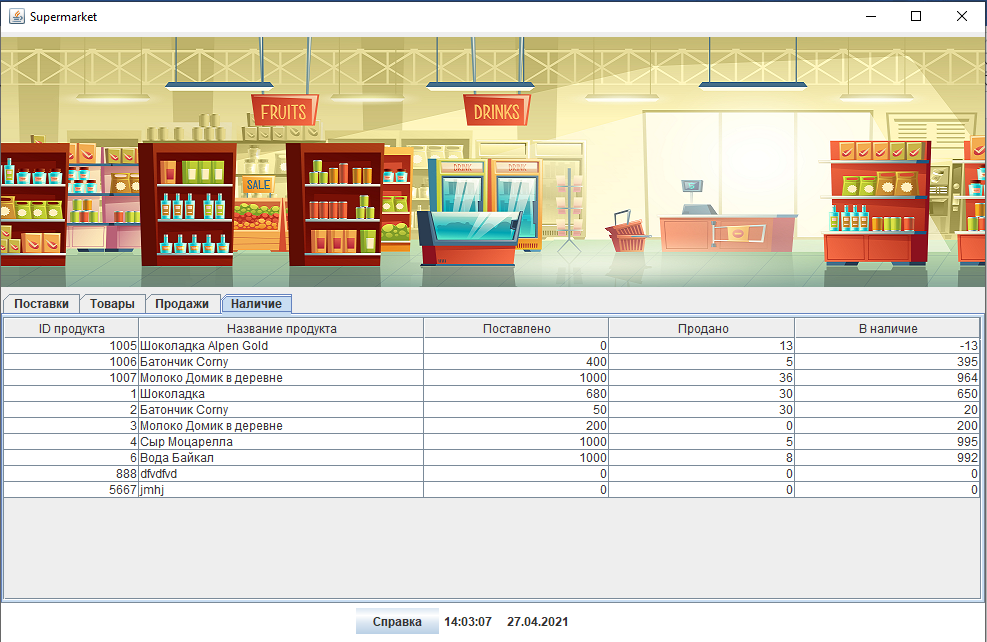


Рисунок 5 – Главное окно программы, вкладка «Наличие»

Диалоговое окно для добавления поставки

Рисунок 6 – Окно добавления поставки

Диалоговое окно для изменения поставки, данные в него автоматически переносятся из выбранной для изменения записи



Рисунок 7 – Окно изменения поставки

Диалоговое окно для добавления товара

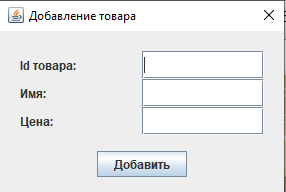


Рисунок 8 – Окно добавления товара

Диалоговое окно для изменения поставки, данные в него автоматически переносятся из выбранной для изменения записи

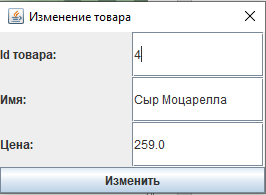


Рисунок 9 – Окно изменения товара

Диалоговое окно для добавления продажи

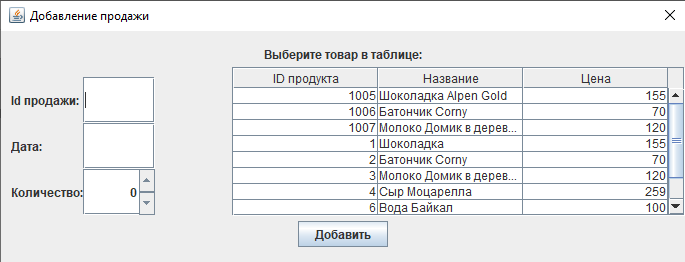


Рисунок 10 – Окно добавления продажи

Диалоговое окно для изменения продажи, данные в него автоматически переносятся из выбранной для изменения записи

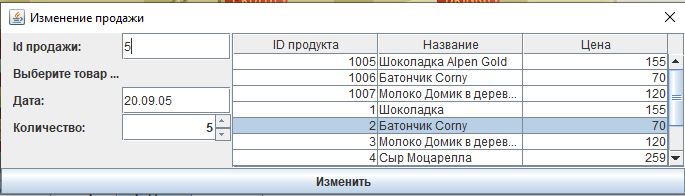


Рисунок 11 – Окно изменения продажи

Диалоговое окно справка

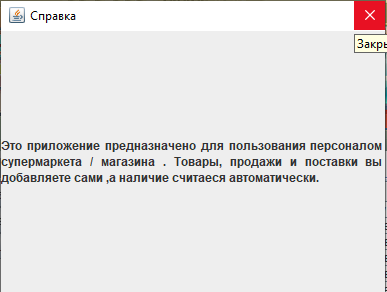


Рисунок 12 – Окно справка

Диалоговое окно для отображения неверного ввода

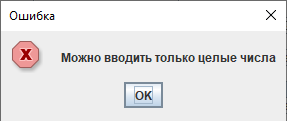


Рисунок 13 – Окно ошибки

Диалоговое окно для отображения отсутствия выбранной записи

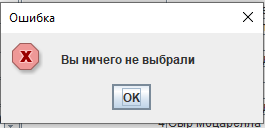


Рисунок 14 – Окно ошибки

4 Описание структуры классов

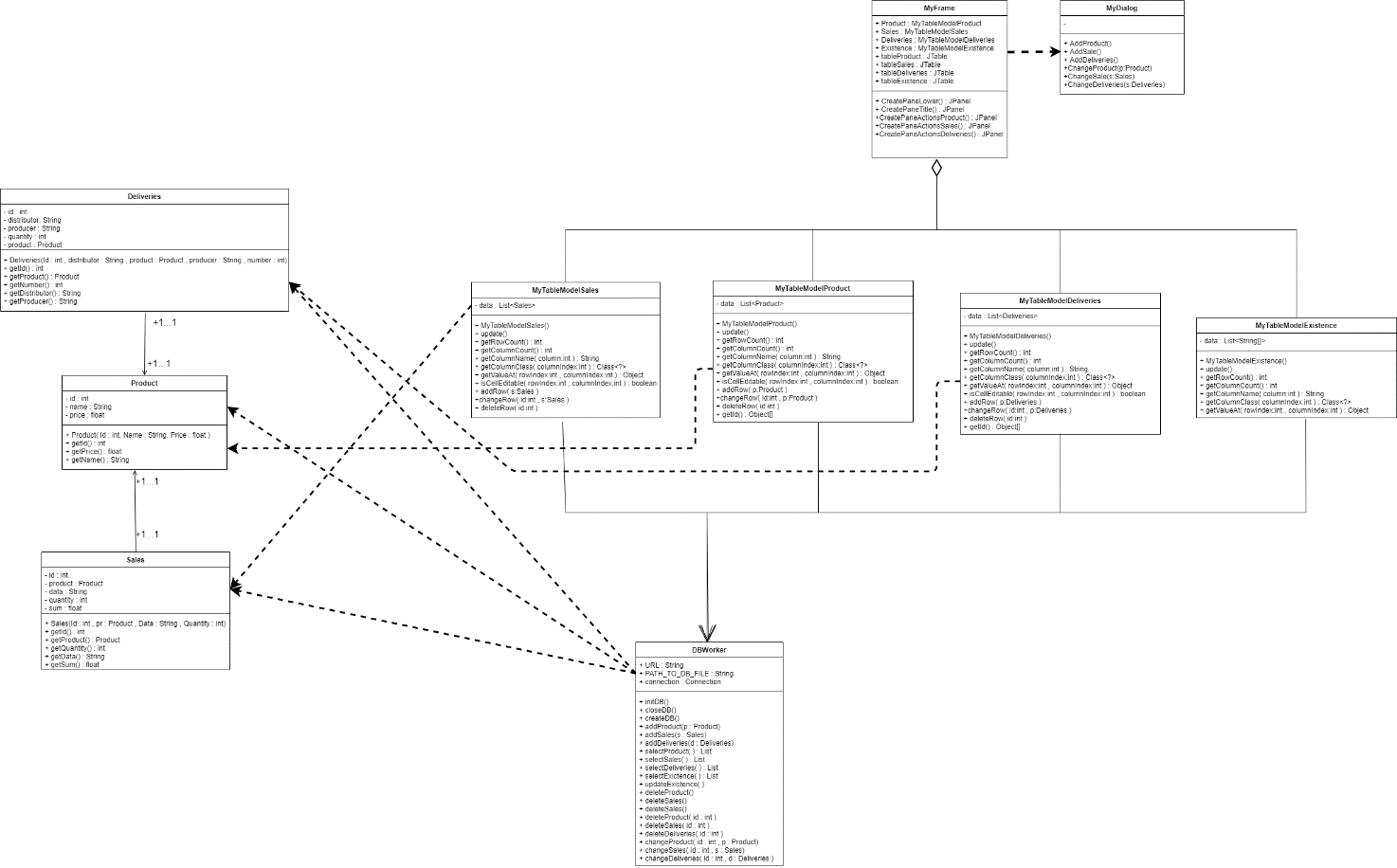


Рисунок 15 – UML-диаграмма классов проекта

5 Проектирование тестов

В ходе разработки было проведено тестирование программы, результаты которого представлены в табл. 1.

Таблица 1 – Тестирование программы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Действие | | Результат действия |
| 1 | Первоначальный запуск программы | | Открытие главного окна, подключение к БД |
| 2 | Добавление новой записи | Товары | Открытие диалогового окна добавления товара с необходимыми полями, добавление записи в БД, добавление товара в таблицу «Наличие» |
| 3 | Поставки | Открытие диалогового окна добавления поставки с необходимыми полями, добавление записи в БД, перерасчет «Наличие» |
| 4 | Продажи | Открытие диалогового окна добавления продажи с необходимыми полями, добавление записи в БД, перерасчет «Наличие» |
| 5 | Удаление записи | Товары | Необходимо выбрать строку в таблице и нажать кнопку удалить, произойдет удаление в БД и удаление товара в «Наличие» |
| 6 | Продажи | Необходимо выбрать строку в таблице Продажи и нажать кнопку удалить, произойдет удаление в БД и перерасчет «Наличие» |
| 7 | Поставки | Необходимо выбрать строку в таблице Поставки и нажать кнопку удалить, произойдет удаление в БД и перерасчет «Наличие» |
| 8 | Изменение записи | Товары | Необходимо выбрать строку в таблице и нажать кнопку изменить, откроется диалоговое окно изменения, с заполненными полями в, внести изменения и сохранить, БД обновляется и изменяются все упоминания этого товара |
| 9 | Продажи | Необходимо выбрать строку в таблице и нажать кнопку изменить, откроется диалоговое окно изменения, с заполненными полями в, внести изменения и сохранить, БД обновляется и перерасчет «Наличие» |
| 10 | Поставки | Необходимо выбрать строку в таблице и нажать кнопку изменить, откроется диалоговое окно изменения, с заполненными полями в, внести изменения и сохранить, БД обновляется и перерасчет «Наличие» |
| 11 | Неверный ввод (вместо целых чисел буквы, символы или пустая строка) | | Открытие диалогового окна с сообщением " Можно вводить только целые числа" |
| 12 | Не выбрана запись | | Открытие диалогового окна с сообщением " Вы ничего не выбрали" |
| 13 | Нажатие кнопки справка | | Открытие окна с информацией о приложение |

6 Результаты тестирования

Действие № 1-открытие главного окна ,подключение к БД

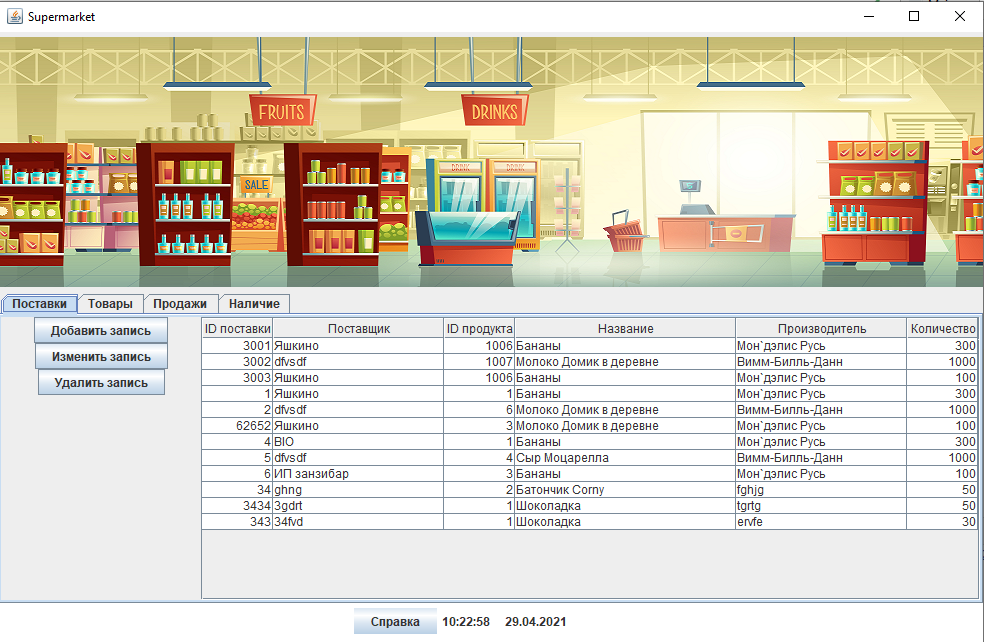


Рисунок 16 – открытие главного окна , подключение БД

Действие № 2 – Добавления записи о товаре



Рисунок 17 – Таблица до добавления

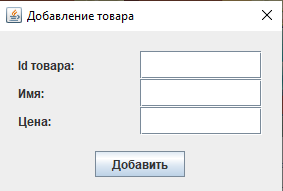


Рисунок 18 – Открытие окна добавления

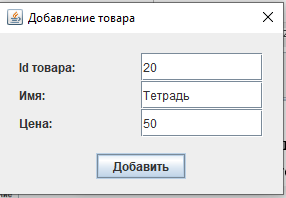


Рисунок 19 – Заполнение полей окна добавления



Рисунок 20 – Добавленная запись



Рисунок 21 – В таблице наличия так же появляется новая запись

Действие № 3 – Добавления записи о поставке



Рисунок 22 – Таблица до добавления



Рисунок 23 – Открытие окна добавления



Рисунок 24 – Заполнение полей окна добавления

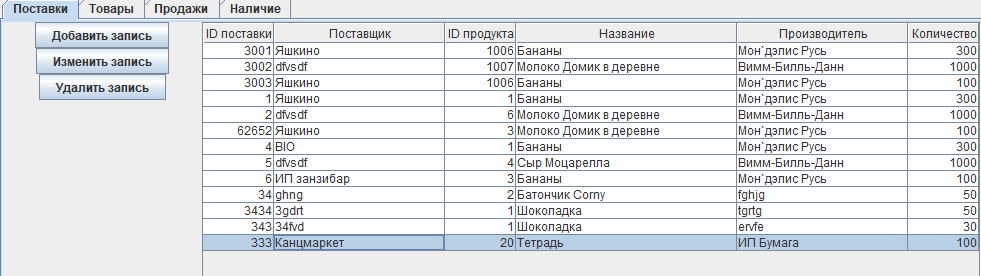


Рисунок 25 – Добавленная запись

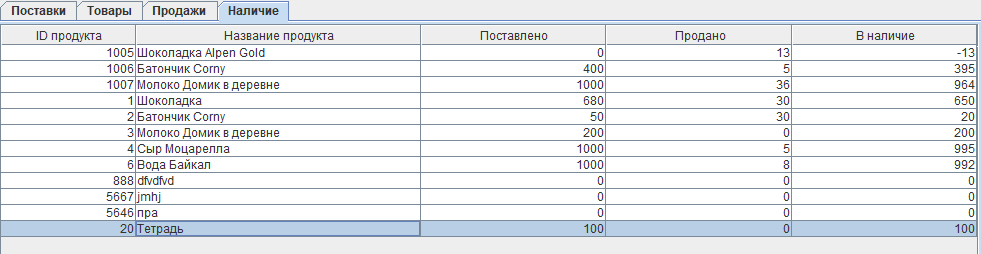


Рисунок 26 – В таблице наличия происходит перерасчет

Действие № 4 – Добавления записи о продаже



Рисунок 27 – Таблица до добавления

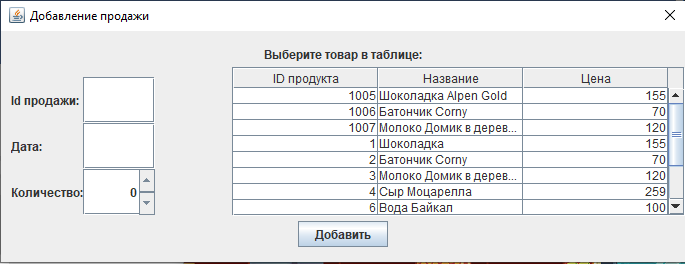


Рисунок 28 – Открытие окна добавления

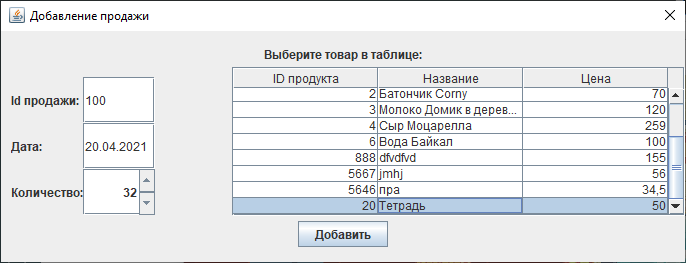


Рисунок 29 – Заполнение полей окна добавления

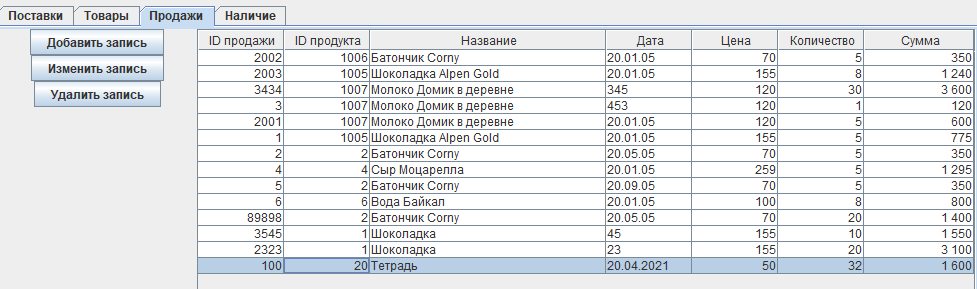


Рисунок 30 – Добавленная запись

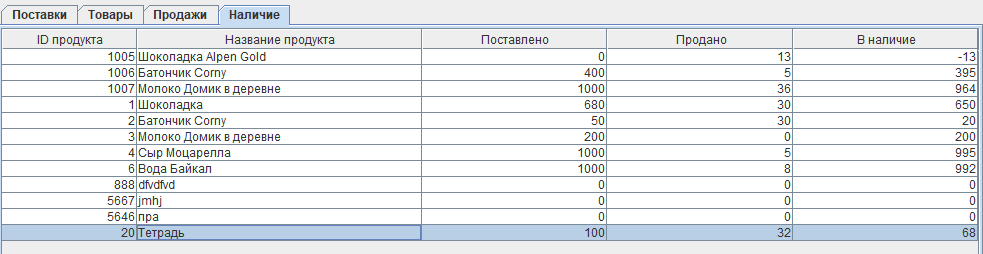


Рисунок 31 – В таблице наличия происходит перерасчет

Действие № 5 – Удаление записи о товаре

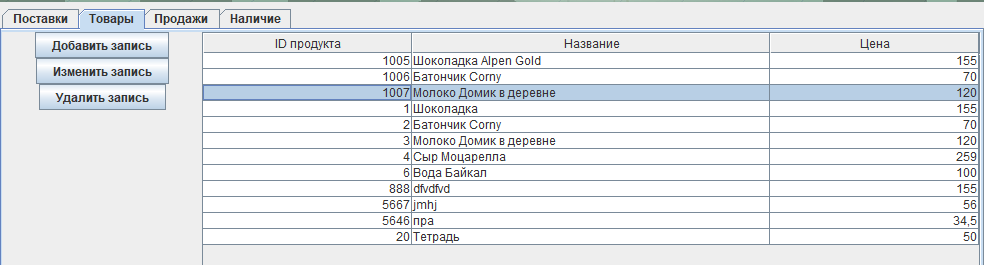


Рисунок 32 – Таблица товаров до удаления



Рисунок 33 – Таблица продаж до удаления



Рисунок 34 – Таблица поставок до удаления

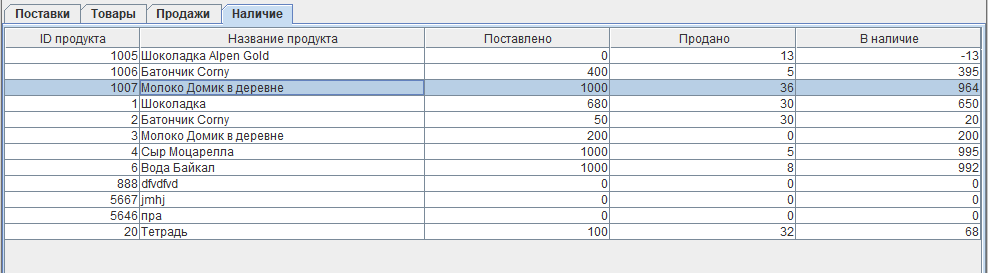


Рисунок 35 – Таблица наличия до удаления



Рисунок 36 – Таблица товаров после удаления



Рисунок 37 – Таблица продаж после удаления



Рисунок 38 – Таблица поставок после удаления

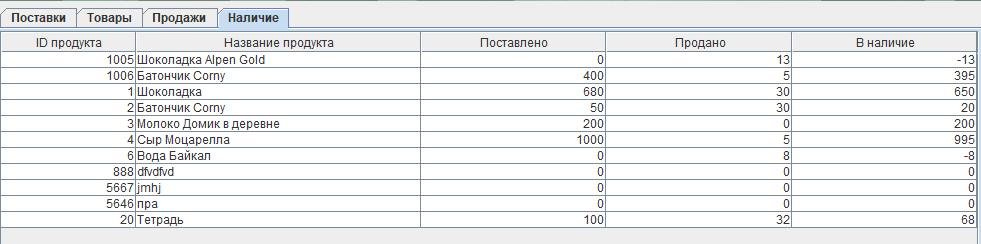


Рисунок 39 – Таблица наличия после удаления

Действие № 6 – Удаление записи о продаже

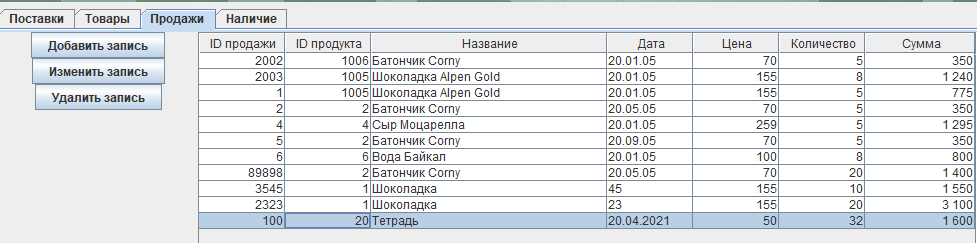


Рисунок 40 – Таблица продаж до удаления

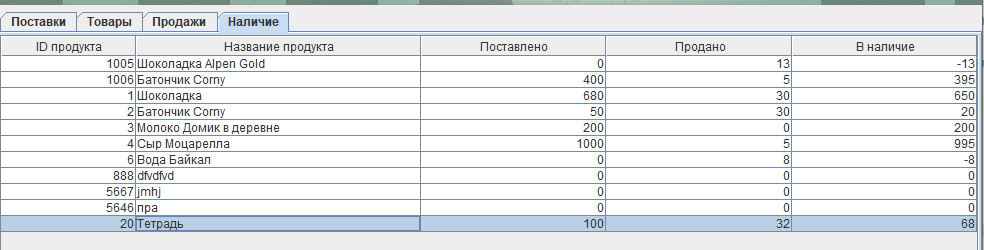


Рисунок 41 – Таблица наличия до удаления

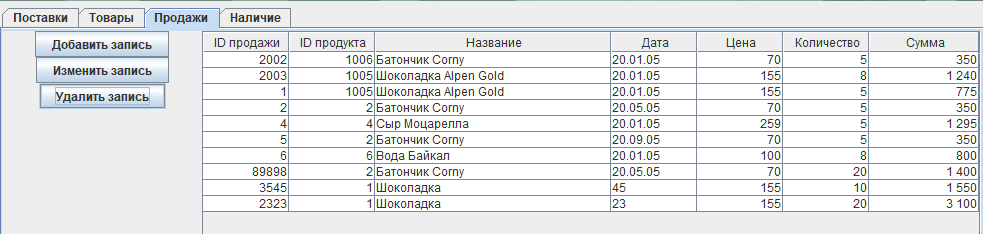


Рисунок 42 – Таблица продаж после удаления

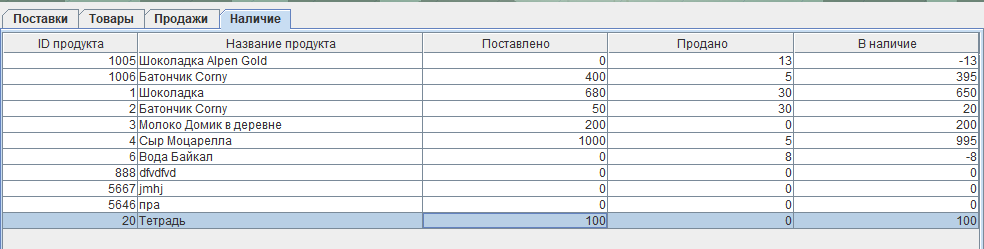


Рисунок 43 – Таблица наличия после удаления

Действие № 7 – Удаление записи о поставке

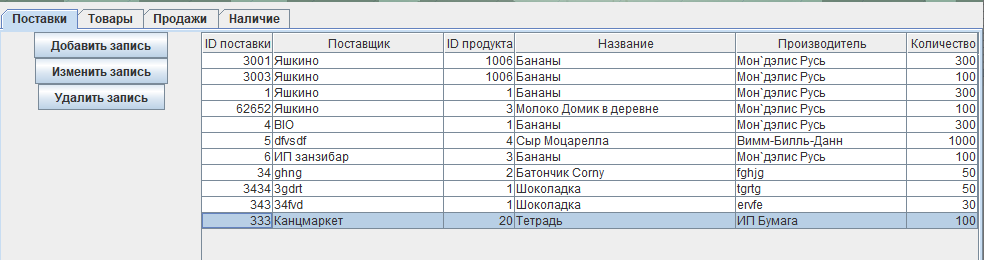


Рисунок 44 – Таблица поставок до удаления

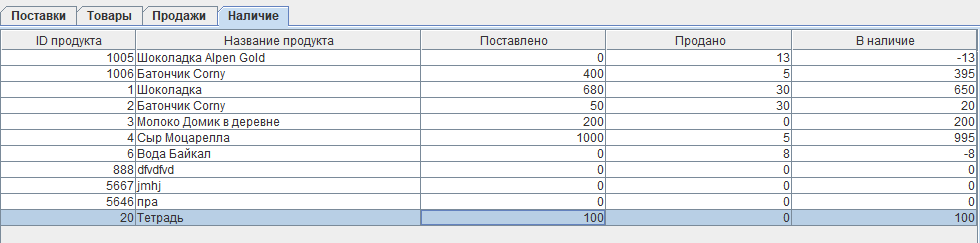


Рисунок 45 – Таблица наличия до удаления

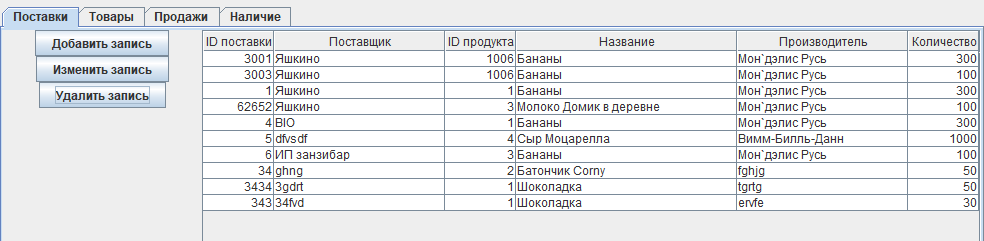


Рисунок 46 – Таблица поставок после удаления

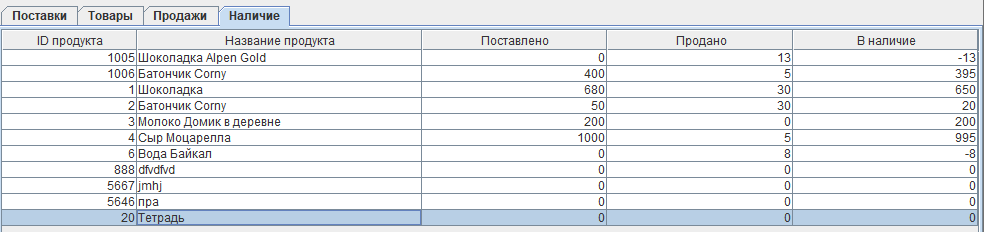


Рисунок 47 – Таблица наличия после удаления

Действие № 8 – Изменение записи о товаре

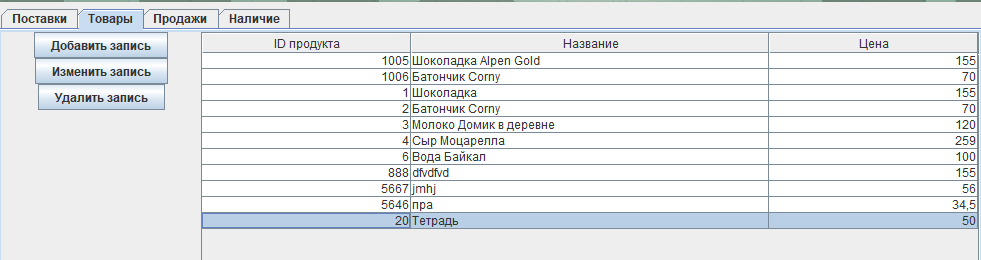


Рисунок 48 – Таблица товаров до изменения



Рисунок 49 – Таблица продаж до изменения

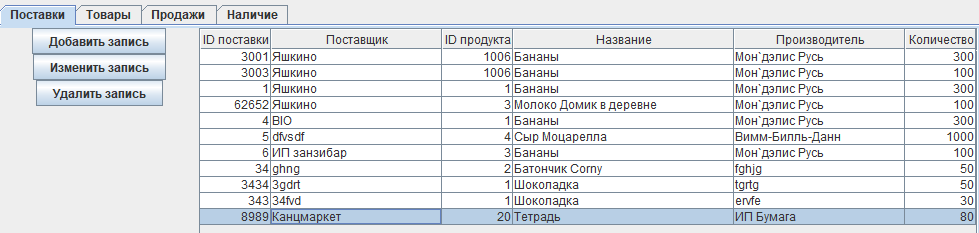


Рисунок 50 – Таблица поставок до изменения

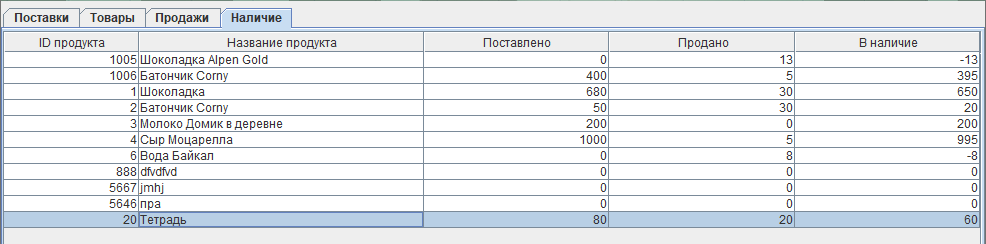


Рисунок 51 – Таблица наличия до изменения

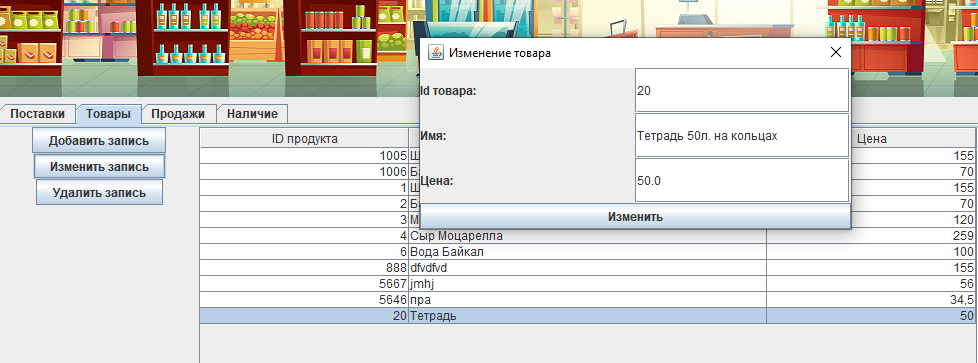


Рисунок 52 – Окно изменения записи



Рисунок 53 – Таблица товаров после изменения



Рисунок 54 – Таблица продаж после изменения

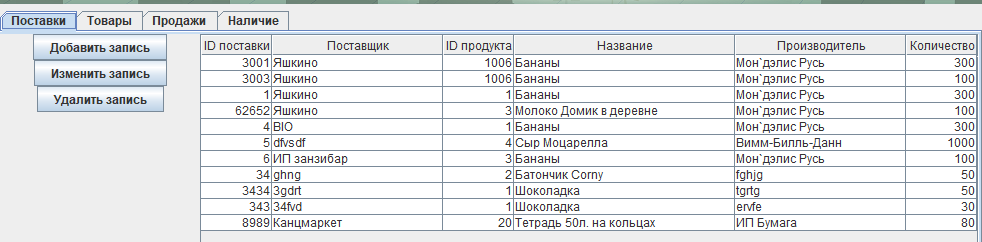


Рисунок 55 – Таблица поставок после изменения

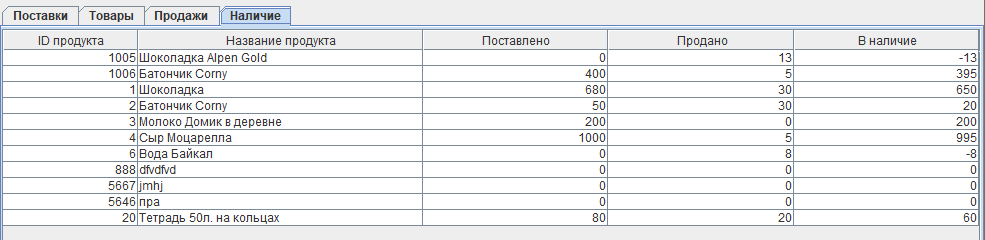


Рисунок 56 – Таблица наличия после изменения

Действие № 9 – Изменение записи о продаже

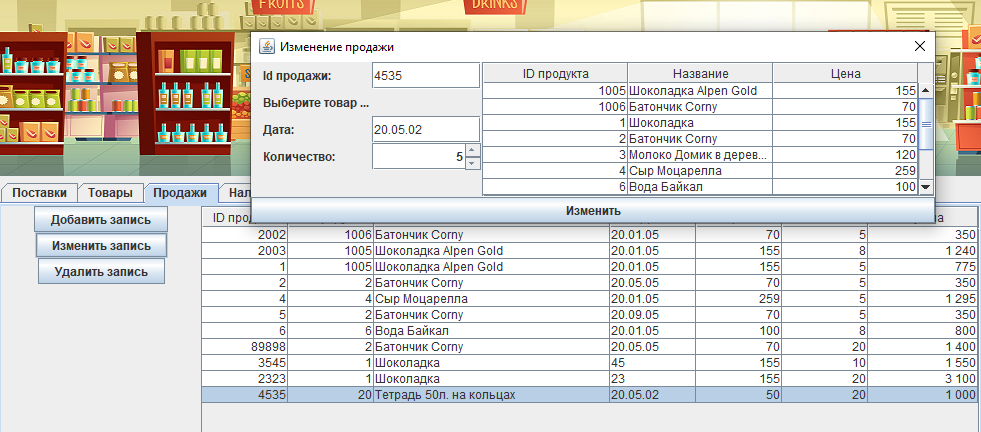


Рисунок 57 – Таблица продаж до изменения

Рисунок 58 – Таблица наличия до изменения



Рисунок 59 – Таблица продаж после изменения

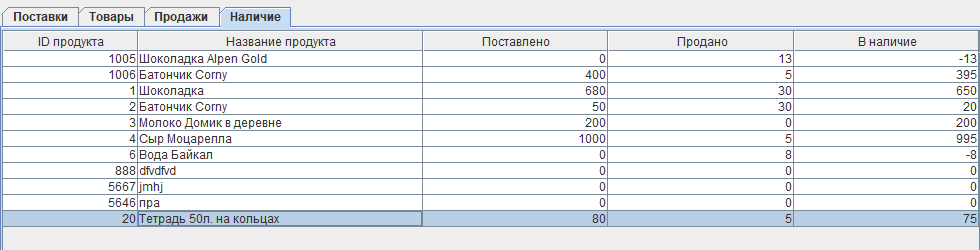


Рисунок 60 – Таблица наличия после изменения

Действие № 10 – Изменение записи о поставке

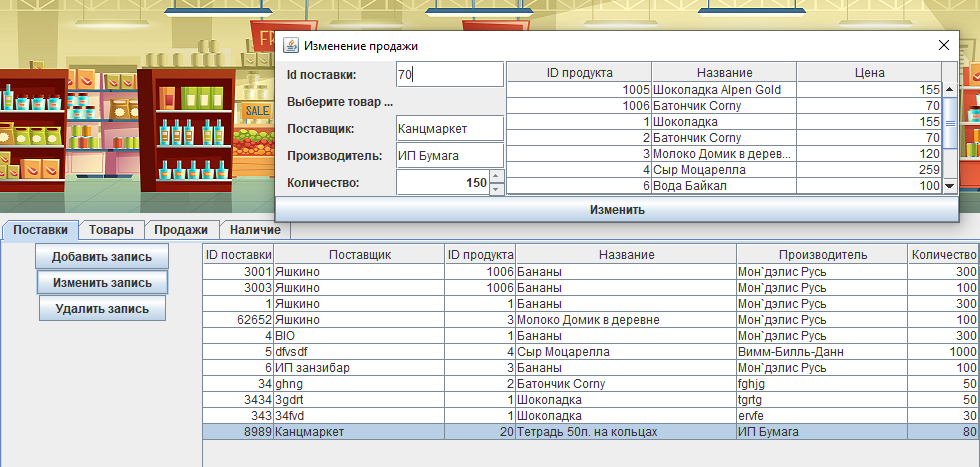


Рисунок 61 – Таблица поставок до изменения

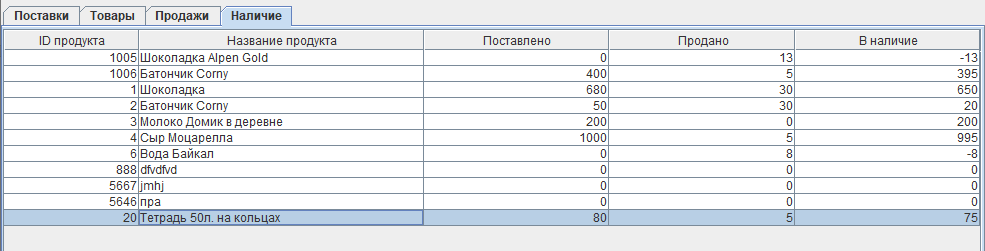


Рисунок 62 – Таблица наличия до изменения

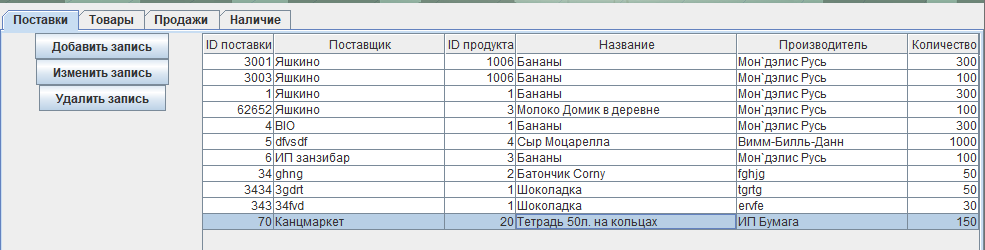


Рисунок 63 – Таблица поставок после изменения

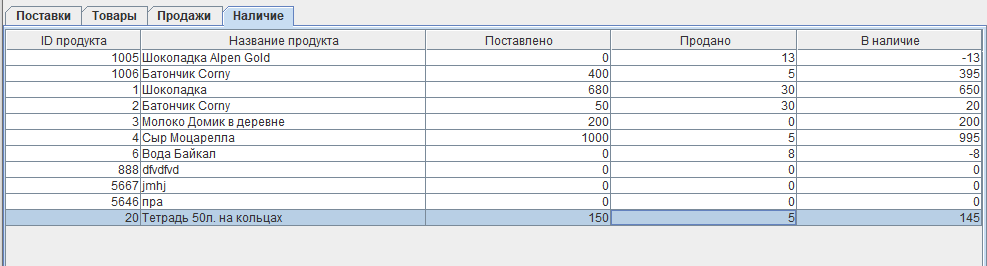


Рисунок 64 – Таблица наличия после изменения

Действие № 11 – Неверный ввод

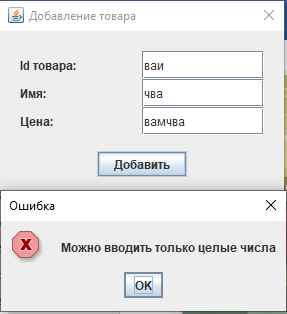


Рисунок 65 – Открытие окна с сообщением об ошибке

Действие № 12 – Не выбрана запись

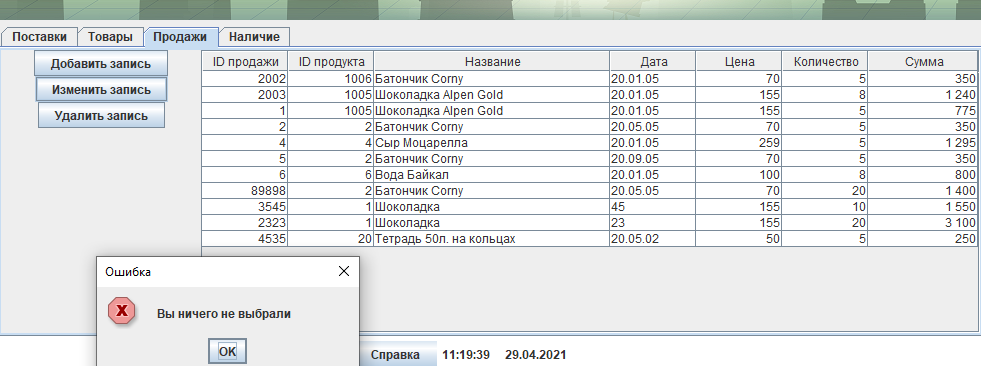


Рисунок 66 – Открытие окна с сообщением об ошибке

Действие № 13 – Нажатие на кнопку справка

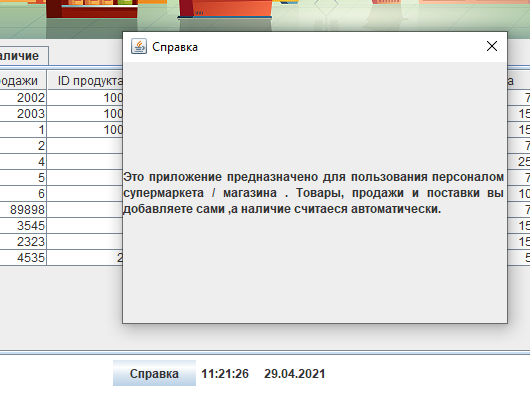


Рисунок 67 – Открытие окна с информацией о приложение

7 Листинг программы

## **7.1 класс Main**

**public class** Main {  
  
 **public static void** main(String[] args) {  
 MyFrame frame=**new** MyFrame();  
 frame.setVisible(**true**);  
 }  
}

## **7.2 класс MyFrame**

**import** javax.imageio.ImageIO;  
**import** javax.swing.\*;  
**import** javax.swing.event.AncestorEvent;  
**import** javax.swing.event.AncestorListener;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.awt.event.MouseAdapter;  
**import** java.awt.event.MouseEvent;  
**import** java.awt.image.BufferedImage;  
**import** java.io.File;  
**import** java.io.IOException;  
**import** java.text.SimpleDateFormat;  
**import** java.util.Date;  
**import** java.util.Locale;  
  
**public class** MyFrame **extends** JFrame {  
  
 **public static** MyTableModelDeliveries *Deliveries* =**new** MyTableModelDeliveries();  
 **public static** JTable *tableDeliveries* =**new** JTable(*Deliveries*);  
 **public static** MyTableModelProduct *Products*=**new** MyTableModelProduct();  
 **public static** JTable *tableProducts*=**new** JTable(*Products*);  
 **public static** MyTableModelSales *Sales*=**new** MyTableModelSales();  
 **public static** JTable *tableSales*=**new** JTable(*Sales*);  
 **public static** MyTableModelExistence *Existence*=**new** MyTableModelExistence();  
 **public static** JTable *tableExistence*=**new** JTable(*Existence*);  
  
 **public** MyFrame(){  
 **super**(**"Supermarket"**);  
 */\*DBWorker.initDB();  
 Products.addRow(new Product(1,"Шоколадка ",155));  
 Products.addRow(new Product(2,"Батончик Corny",70));  
 Products.addRow(new Product(3,"Молоко Домик в деревне",120));  
 Products.addRow(new Product(4,"Сыр Моцарелла",259));  
 Products.addRow(new Product(6,"Вода Байкал",100));  
 Sales.addRow(new Sales(1,new Product(1005,"Шоколадка Alpen Gold",155),"20.01.05",5));  
 Sales.addRow(new Sales(2,new Product(2,"Батончик Corny",70),"20.05.05",5));  
 Sales.addRow(new Sales(3,new Product(6,"Вода Байкал",100),"20.01.05",8));  
 Sales.addRow(new Sales(4,new Product(4,"Сыр Моцарелла",259),"20.01.05",5));  
 Sales.addRow(new Sales(5,new Product(2,"Батончик Corny",70),"20.09.05",5));  
 Sales.addRow(new Sales(6,new Product(6,"Вода Байкал",100),"20.01.05",8));  
 Deliveries.addRow(new Deliveries(1,"Яшкино",new Product(1,"Бананы",188),"Мон`дэлис Русь",300));  
 Deliveries.addRow(new Deliveries(2,"dfvsdf",new Product(6,"Молоко Домик в деревне",120),"Вимм-Билль-Данн",1000));  
 Deliveries.addRow(new Deliveries(3,"Яшкино",new Product(3,"Бананы",188),"Мон`дэлис Русь",100));  
 Deliveries.addRow(new Deliveries(4,"BIO",new Product(1,"Бананы",188),"Мон`дэлис Русь",300));  
 Deliveries.addRow(new Deliveries(5,"dfvsdf",new Product(4,"Сыр Моцарелла",259),"Вимм-Билль-Данн",1000));  
 Deliveries.addRow(new Deliveries(6,"ИП занзибар",new Product(3,"Бананы",188),"Мон`дэлис Русь",100));  
 DBWorker.closeDB();\*/* JTabbedPane Data = **new** JTabbedPane(JTabbedPane.***TOP***);  
 {  
 *tableDeliveries*.getColumnModel().getColumn(0).setPreferredWidth(50);  
 *tableDeliveries*.getColumnModel().getColumn(1).setPreferredWidth(150);  
 *tableDeliveries*.getColumnModel().getColumn(2).setPreferredWidth(50);  
 *tableDeliveries*.getColumnModel().getColumn(3).setPreferredWidth(200);  
 *tableDeliveries*.getColumnModel().getColumn(4).setPreferredWidth(150);  
 *tableDeliveries*.getColumnModel().getColumn(5).setPreferredWidth(50);  
  
 *tableExistence*.getColumnModel().getColumn(0).setPreferredWidth(50);  
 *tableExistence*.getColumnModel().getColumn(1).setPreferredWidth(200);  
 *tableExistence*.getColumnModel().getColumn(2).setPreferredWidth(100);  
 *tableExistence*.getColumnModel().getColumn(3).setPreferredWidth(100);  
 *tableExistence*.getColumnModel().getColumn(4).setPreferredWidth(100);  
  
 *tableProducts*.getColumnModel().getColumn(0).setPreferredWidth(50);  
 *tableProducts*.getColumnModel().getColumn(1).setPreferredWidth(200);  
 *tableProducts*.getColumnModel().getColumn(2).setPreferredWidth(50);  
  
 *tableSales*.getColumnModel().getColumn(0).setPreferredWidth(50);  
 *tableSales*.getColumnModel().getColumn(1).setPreferredWidth(50);  
 *tableSales*.getColumnModel().getColumn(2).setPreferredWidth(200);  
 *tableSales*.getColumnModel().getColumn(3).setPreferredWidth(50);  
 *tableSales*.getColumnModel().getColumn(4).setPreferredWidth(50);  
 *tableSales*.getColumnModel().getColumn(5).setPreferredWidth(50);  
 *tableSales*.getColumnModel().getColumn(5).setPreferredWidth(50);  
  
 JPanel PaneDeliveries = **new** JPanel();  
 JPanel PaneProduct = **new** JPanel();  
 JPanel PaneSales = **new** JPanel();  
 JPanel PaneExistence = **new** JPanel();  
  
 PaneDeliveries.setLayout(**new** BorderLayout());  
 PaneDeliveries.add(CreatePaneActionsDeliveries(),BorderLayout.WEST);  
 PaneDeliveries.add(**new** JScrollPane(tableDeliveries),BorderLayout.CENTER);  
 Data.addTab(**"Поставки"**, PaneDeliveries);  
  
 PaneProduct.setLayout(**new** BorderLayout());  
 PaneProduct.add(CreatePaneActionsProduct(),BorderLayout.WEST);  
 PaneProduct.add(**new** JScrollPane(tableProducts),BorderLayout.CENTER);  
 Data.addTab(**"Товары"**, PaneProduct);  
  
 PaneSales.setLayout(**new** BorderLayout());  
 PaneSales.add(CreatePaneActionsSales(),BorderLayout.WEST);  
 PaneSales.add(**new** JScrollPane(tableSales),BorderLayout.CENTER);  
 Data.addTab(**"Продажи"**, PaneSales);  
  
 PaneExistence.setLayout(**new** BorderLayout());  
 PaneExistence.add(**new** JScrollPane(tableExistence),BorderLayout.CENTER);  
 Data.addTab(**"Наличие"**, PaneExistence);  
 }  
  
 add(CreatePaneTitle(), BorderLayout.NORTH);  
 add(CreatePaneLower(),BorderLayout.SOUTH);  
 add(Data,BorderLayout.CENTER);  
 setMinimumSize(**new** Dimension(1000,650));  
 setLocationRelativeTo(**null**);  
 }  
  
 **public static** JPanel CreatePaneLower(){  
 JButton Reference=**new** JButton(**"Справка"**);  
 Reference.setBorderPainted( **false** );  
 Reference.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 JDialog dialog=**new** JDialog();  
 dialog.setTitle(**"Справка"**);  
 JLabel l =**new** JLabel(**""**);  
 l.setText(**"<html><p align=\"justify\">Это приложение предназначено для пользования персоналом "** +  
 **" супермаркета / магазина . "** +  
 **"Товары, продажи и поставки вы добавляете сами "** +  
 **" ,а наличие считаеся автоматически."** +  
 **"</p></html>"**);  
 l.setBackground(Color.white);  
 dialog.add(l);  
 dialog.setSize(400,300);  
 dialog.setModal(**true**);  
 dialog.setVisible(**true**);  
 dialog.pack();  
 dialog.setLocationRelativeTo(**null**);  
 }  
 });  
 JLabel j=**new** JLabel(**""**);  
 Timer timer = **new** Timer(1000,**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent arg0) {  
 Date current = **new** Date();  
 SimpleDateFormat formatter = **new** SimpleDateFormat(**"HH:mm:ss dd.MM.yyyy"**);  
 String message = formatter.format(current);  
 j.setText(message+**" "**);  
 j.repaint();  
 j.revalidate();  
 }  
 });  
 timer.start();  
 JPanel PaneTimer=**new** JPanel();  
 PaneTimer.add(j);  
 j.setHorizontalAlignment(SwingConstants.CENTER);  
 JPanel PaneLower = **new** JPanel(**new** FlowLayout());  
 PaneLower.add(Reference);  
 PaneLower.setPreferredSize(**new** Dimension(1000,40));  
 PaneLower.add(j);  
 PaneLower.setBackground(Color.white);  
 **return** PaneLower;  
 }  
  
 **public static** JPanel CreatePaneTitle(){  
 JPanel PaneTitle = **new** JPanel();  
 BufferedImage myPicture = **null**;  
 **try** {  
 myPicture = ImageIO.read(**new** File(**"C:\\Юлия\\Институт\\2 курс\\ООП\\Лабы\\Лаба 6.1\\out\\production\\Лаба 6.1\\com\\company\\2.png"**));  
 JLabel picLabel = **new** JLabel(**new** ImageIcon(myPicture));  
 PaneTitle.add(picLabel);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 PaneTitle.setMinimumSize(**new** Dimension(1000,200));  
 **return** PaneTitle;  
 }  
  
 **public static** JPanel CreatePaneActionsDeliveries(){  
 JPanel PaneActionsDeliveries=**new** JPanel();  
 PaneActionsDeliveries.setLayout(**new** BoxLayout(PaneActionsDeliveries, BoxLayout.Y\_AXIS));  
 JButton add=**new** JButton(**"Добавить запись"**);  
 add.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 MyDialog.AddDeliveries();  
 }  
 });  
 JButton change=**new** JButton(**"Изменить запись"**);  
 change.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **int** row= tableDeliveries.getSelectedRow();  
 **if** (row==-1){  
 JOptionPane.showMessageDialog(PaneActionsDeliveries,  
 **" Вы ничего не выбрали"**,  
 **"Ошибка"**,  
 JOptionPane.ERROR\_MESSAGE);  
 **return**;  
 }  
 Deliveries d=**new** Deliveries(  
 Integer.parseInt(String.valueOf(Deliveries.getValueAt(row,0))),  
 String.valueOf(Deliveries.getValueAt(row,1)),  
 **new** Product(  
 Integer.parseInt(String.valueOf(Deliveries.getValueAt(row,2))),  
 String.valueOf(Deliveries.getValueAt(row,3)),  
 0  
 ),  
 String.valueOf(Deliveries.getValueAt(row,4)) ,  
 Integer.parseInt(String.valueOf(Deliveries.getValueAt(row,5)))  
 );  
 MyDialog.ChangeDeliveries(d);  
 }  
 });  
 JButton delete=**new** JButton(**"Удалить запись"**);  
 delete.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **int** row= tableDeliveries.getSelectedRow();  
 **if** (row==-1){  
 JOptionPane.showMessageDialog(PaneActionsDeliveries,  
 **" Вы ничего не выбрали"**,  
 **"Ошибка"**,  
 JOptionPane.ERROR\_MESSAGE);  
 **return**;  
 }  
 **int** id= (**int**) Deliveries.getValueAt(row,0);  
 Deliveries.deleteRow(id);  
 Existence.update();  
 }  
 });  
 PaneActionsDeliveries.add(add);  
 PaneActionsDeliveries.add(change);  
 PaneActionsDeliveries.add(delete);  
 add.setAlignmentX(CENTER\_ALIGNMENT);  
 change.setAlignmentX(CENTER\_ALIGNMENT);  
 delete.setAlignmentX(CENTER\_ALIGNMENT);  
 PaneActionsDeliveries.setPreferredSize(**new** Dimension(200,500));  
 **return** PaneActionsDeliveries;  
 }  
 **public static** JPanel CreatePaneActionsProduct(){  
 JPanel PaneActionsProduct=**new** JPanel();  
 PaneActionsProduct.setLayout(**new** BoxLayout(PaneActionsProduct, BoxLayout.Y\_AXIS));  
 JButton add=**new** JButton(**"Добавить запись"**);  
 add.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 MyDialog.AddProduct();  
 }  
 });  
 JButton change=**new** JButton(**"Изменить запись"**);  
 change.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **int** row= tableProducts.getSelectedRow();  
 **if** (row==-1){  
 JOptionPane.showMessageDialog(PaneActionsProduct,  
 **" Вы ничего не выбрали"**,  
 **"Ошибка"**,  
 JOptionPane.ERROR\_MESSAGE);  
 **return**;  
 }  
 Product p= **new** Product(  
 Integer.parseInt(String.valueOf( Products.getValueAt(row,0))),  
 String.valueOf(Products.getValueAt(row,1)),  
 Float.valueOf(String.valueOf( Products.getValueAt(row,2)))  
 );  
 MyDialog.ChangeProduct(p);  
 }  
 });  
 JButton delete=**new** JButton(**"Удалить запись"**);  
 delete.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **int** row=tableProducts.getSelectedRow();  
 **if** (row==-1){  
 JOptionPane.showMessageDialog(PaneActionsProduct,  
 **" Вы ничего не выбрали"**,  
 **"Ошибка"**,  
 JOptionPane.ERROR\_MESSAGE);  
 **return**;  
 }  
 **int** id= (**int**) Products.getValueAt(row,0);  
 Products.deleteRow(id);  
 Deliveries.update();  
 Sales.update();  
 Existence.update();  
 }  
 });  
 add.setAlignmentX(CENTER\_ALIGNMENT);  
 change.setAlignmentX(CENTER\_ALIGNMENT);  
 delete.setAlignmentX(CENTER\_ALIGNMENT);  
 PaneActionsProduct.add(add);  
 PaneActionsProduct.add(change);  
 PaneActionsProduct.add(delete);  
 PaneActionsProduct.setPreferredSize(**new** Dimension(200,500));  
 **return** PaneActionsProduct;  
 }  
 **public static** JPanel CreatePaneActionsSales(){  
 JPanel PaneActionsSales=**new** JPanel();  
 PaneActionsSales.setLayout(**new** BoxLayout(PaneActionsSales, BoxLayout.Y\_AXIS));  
  
 JButton add=**new** JButton(**"Добавить запись"**);  
 add.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 MyDialog.AddSale();  
 }  
 });  
 add.setAlignmentX(CENTER\_ALIGNMENT);  
 JButton change=**new** JButton(**"Изменить запись"**);  
 change.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **int** row= tableSales.getSelectedRow();  
 **if** (row==-1){  
 JOptionPane.showMessageDialog(PaneActionsSales,  
 **" Вы ничего не выбрали"**,  
 **"Ошибка"**,  
 JOptionPane.ERROR\_MESSAGE);  
 **return**;  
 }  
 Sales p= **new** Sales(  
 Integer.parseInt(String.valueOf(Sales.getValueAt(row,0))),  
 **new** Product( Integer.parseInt(String.valueOf( Sales.getValueAt(row,1))),  
 String.valueOf(Sales.getValueAt(row,2)),  
 Float.valueOf(String.valueOf(Sales.getValueAt(row,5)))),  
 String.valueOf(Sales.getValueAt(row,3)),  
 Integer.valueOf(String.valueOf(Sales.getValueAt(row,5)))  
 );  
 MyDialog.ChangeSale(p);  
 }  
 });  
 change.setAlignmentX(CENTER\_ALIGNMENT);  
 JButton delete=**new** JButton(**"Удалить запись"**);  
 delete.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **int** row= tableSales.getSelectedRow();  
 **if** (row==-1){  
 JOptionPane.showMessageDialog(PaneActionsSales,  
 **" Вы ничего не выбрали"**,  
 **"Ошибка"**,  
 JOptionPane.ERROR\_MESSAGE);  
 **return**;  
 }  
 **int** id= (**int**) Sales.getValueAt(row,0);  
 Sales.deleteRow(id);  
 Existence.update();  
 }  
 });  
 delete.setAlignmentX(CENTER\_ALIGNMENT);  
 PaneActionsSales.add(add);  
 PaneActionsSales.add(change);  
 PaneActionsSales.add(delete);  
 PaneActionsSales.setPreferredSize(**new** Dimension(200,500));  
 **return** PaneActionsSales;  
 }  
}

## **7.3 класс MyDialog**

**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
  
**public class** MyDialog {  
  
 **public static void** AddProduct() {  
 JDialog dialog=**new** JDialog();  
 dialog.setTitle(**"Добавление товара"**);  
 JPanel Pane1=**new** JPanel(**new** GridLayout(3,2));  
 JTextField id=**new** JTextField();  
 JTextField name=**new** JTextField();  
 JTextField price= **new** JTextField();  
  
 Pane1.add(**new** JLabel(**"Id товара:"**));Pane1.add(id);  
 Pane1.add(**new** JLabel(**"Имя:"**));Pane1.add(name);  
 Pane1.add(**new** JLabel(**"Цена:"**));Pane1.add(price);  
 JButton addProduct=**new** JButton(**"Добавить"**);  
 addProduct.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 **boolean** exist=**false**;  
 **for** (**int** i=0;i<MyFrame.*Products*.getRowCount();i++) {  
 **if** (Integer.*parseInt*(id.getText()) == Integer.*parseInt*(String.*valueOf*(MyFrame.*Products*.getValueAt(i, 0)))){  
 exist=**true**;  
 }  
 }  
 **if**(exist){  
 JOptionPane.*showMessageDialog*(dialog,  
 **" Уже существует продукт с таким ID!"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }**else**{  
 MyFrame.*Products*.addRow(**new** Product(  
 Integer.*parseInt*(String.*valueOf*(id.getText())),  
 String.*valueOf*(name.getText()),  
 Float.*valueOf*(price.getText())  
 ));  
 dialog.dispose();  
 MyFrame.*Existence*.update();  
 }  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(dialog,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 id.setText(**""**);  
 name.setText(**""**);  
 price.setText(**""**);  
 }  
 });  
 JPanel pane=**new** JPanel();  
 pane.add(addProduct);  
 dialog.add(Pane1,BorderLayout.***CENTER***);  
 dialog.add(pane,BorderLayout.***SOUTH***);  
 dialog.setModal(**true**);  
 dialog.setSize(**new** Dimension(300,200));  
 Pane1.setBorder(BorderFactory.*createEmptyBorder*(20, 20, 10, 20));  
 pane.setBorder(BorderFactory.*createEmptyBorder*(0,10,10,10));  
 dialog.setVisible(**true**);  
 dialog.pack();  
 dialog.setLocationRelativeTo(**null**);  
 }  
 **public static void** AddSale() {  
 JDialog dialog=**new** JDialog();  
 dialog.setTitle(**"Добавление продажи"**);  
 JPanel Pane1=**new** JPanel(**new** GridLayout(3,2));  
 JTextField id=**new** JTextField();  
 JTextField data=**new** JTextField();  
 SpinnerModel n = **new** SpinnerNumberModel(0, 0, 500, 10);  
 JSpinner number = **new** JSpinner(n);  
 Pane1.add(**new** JLabel(**"Id продажи:"**));Pane1.add(id);  
 Pane1.add(**new** JLabel(**"Дата:"**));Pane1.add(data);  
 Pane1.add(**new** JLabel(**"Количество:"**));Pane1.add(number);  
 JButton addSale=**new** JButton(**"Добавить"**);  
 MyTableModelProduct Products=**new** MyTableModelProduct();  
 JTable tableProducts=**new** JTable(Products);  
 addSale.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 **int** row= tableProducts.getSelectedRow();  
 **if** (row==-1){  
 JOptionPane.*showMessageDialog*(dialog,  
 **" Вы ничего не выбрали"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 **return**;  
 }  
 Product p= **new** Product(  
 Integer.*parseInt*(String.*valueOf*( Products.getValueAt(row,0))),  
 String.*valueOf*(Products.getValueAt(row,1)),  
 Float.*valueOf*(String.*valueOf*( Products.getValueAt(row,2)))  
 );  
 MyFrame.*Sales*.addRow(**new** Sales(  
 Integer.*parseInt*(id.getText()),  
 p,  
 String.*valueOf*(data.getText()),  
 Integer.*parseInt*(String.*valueOf*( number.getValue()))  
 ));  
 dialog.dispose();  
 MyFrame.*Existence*.update();  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(dialog,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 id.setText(**""**);  
 data.setText(**""**);  
 number.setValue(0);  
 }  
 });  
  
 Pane1.setBorder(BorderFactory.*createEmptyBorder*(10, 10, 0, 0));  
 dialog.add(Pane1,BorderLayout.***WEST***);  
  
 JPanel pane=**new** JPanel();  
 pane.add(addSale);  
 pane.setBorder(BorderFactory.*createEmptyBorder*(0,10,10,10));  
 dialog.add(pane,BorderLayout.***SOUTH***);  
  
 JPanel t=**new** JPanel();  
 t.add(**new** JLabel(**"Выберите товар в таблице:"**));  
 t.setAlignmentX(Component.***CENTER\_ALIGNMENT***);  
 t.setBorder(BorderFactory.*createEmptyBorder*(10,10,0,10));  
 dialog.add(t,BorderLayout.***NORTH***);  
 JScrollPane p=**new** JScrollPane(tableProducts);  
 dialog.add(p,BorderLayout.***EAST***);  
  
 dialog.setSize(700,270);  
 dialog.setModal(**true**);  
 dialog.setVisible(**true**);  
 dialog.pack();  
 dialog.setLocationRelativeTo(**null**);  
 }  
 **public static void** AddDeliveries() {  
 JDialog dialog=**new** JDialog();  
 dialog.setTitle(**"Добавление поставки"**);  
 JPanel Pane1=**new** JPanel(**new** GridLayout(5,2));  
 JTextField id=**new** JTextField();  
 JTextField distributor=**new** JTextField();  
 JTextField producer=**new** JTextField();  
 SpinnerModel n = **new** SpinnerNumberModel(0, 0, 500, 10);  
 JSpinner number = **new** JSpinner(n);  
 Pane1.add(**new** JLabel(**"Id поставки:"**));Pane1.add(id);  
 Pane1.add(**new** JLabel(**"Выберите товар справа в таблице:"**));Pane1.add(**new** JLabel(**" "**));  
 Pane1.add(**new** JLabel(**"Поставщик:"**));Pane1.add(distributor);  
 Pane1.add(**new** JLabel(**"Производитель:"**));Pane1.add(producer);  
 Pane1.add(**new** JLabel(**"Количество:"**));Pane1.add(number);  
 JButton addDeliveries=**new** JButton(**"Добавить"**);  
 MyTableModelProduct Products=**new** MyTableModelProduct();  
 JTable tableProducts=**new** JTable(Products);  
 addDeliveries.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 **int** row= tableProducts.getSelectedRow();  
 **if** (row==-1){  
 JOptionPane.*showMessageDialog*(dialog,  
 **" Вы ничего не выбрали"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 **return**;  
 }  
 Product p= **new** Product(  
 Integer.*parseInt*(String.*valueOf*( Products.getValueAt(row,0))),  
 String.*valueOf*(Products.getValueAt(row,1)),  
 Float.*valueOf*(String.*valueOf*( Products.getValueAt(row,2)))  
 );  
 MyFrame.*Deliveries*.addRow(**new** Deliveries(  
 Integer.*parseInt*(id.getText()),  
 String.*valueOf*(distributor.getText()),  
 p,  
 String.*valueOf*(producer.getText()),  
 Integer.*parseInt*(String.*valueOf*( number.getValue()))  
 ));  
 dialog.dispose();  
 MyFrame.*Existence*.update();  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(dialog,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 id.setText(**""**);  
 distributor.setText(**""**);  
 number.setValue(0);  
 producer.setText(**""**);  
 }  
 });  
 dialog.add(Pane1,BorderLayout.***CENTER***);  
 dialog.add(addDeliveries,BorderLayout.***SOUTH***);  
 dialog.add(**new** JScrollPane(tableProducts),BorderLayout.***EAST***);  
 dialog.add(**new** JLabel(**" "**),BorderLayout.***WEST***);  
 dialog.setSize(700,200);  
 dialog.setModal(**true**);  
 dialog.setVisible(**true**);  
 dialog.pack();  
 dialog.setLocationRelativeTo(**null**);  
 }  
  
 **public static void** ChangeProduct(Product p) {  
 JDialog dialog=**new** JDialog();  
 dialog.setTitle(**"Изменение товара"**);  
 JPanel Pane1=**new** JPanel(**new** GridLayout(3,2));  
 JTextField id=**new** JTextField();id.setText(String.*valueOf*(p.getId()));  
 JTextField name=**new** JTextField();name.setText(p.getName());  
 JTextField price= **new** JTextField();price.setText(String.*valueOf*(p.getPrice()));  
 Pane1.add(**new** JLabel(**"Id товара:"**));Pane1.add(id);  
 Pane1.add(**new** JLabel(**"Имя:"**));Pane1.add(name);  
 Pane1.add(**new** JLabel(**"Цена:"**));Pane1.add(price);  
 JButton changeProduct=**new** JButton(**"Изменить"**);  
 changeProduct.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 **boolean** exist=**false**;  
  
 MyFrame.*Products*.changeRow(p.getId(),**new** Product(  
 Integer.*parseInt*(id.getText()),  
 String.*valueOf*(name.getText()),  
 Float.*valueOf*(price.getText())  
 ));  
 dialog.dispose();  
 MyFrame.*Existence*.update();  
 MyFrame.*Deliveries*.update();  
 MyFrame.*Sales*.update();  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(dialog,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 id.setText(**""**);  
 name.setText(**""**);  
 price.setText(**""**);  
 }  
 });  
 dialog.add(Pane1,BorderLayout.***CENTER***);  
 dialog.add(changeProduct,BorderLayout.***SOUTH***);  
 dialog.setSize(200,200);  
 dialog.setModal(**true**);  
 dialog.setVisible(**true**);  
 dialog.setLocationRelativeTo(**null**);  
 dialog.pack();  
 }  
 **public static void** ChangeDeliveries(Deliveries s){  
 JDialog dialog=**new** JDialog();  
 dialog.setTitle(**"Изменение продажи"**);  
 JPanel Pane1=**new** JPanel(**new** GridLayout(5,2));  
 JTextField id=**new** JTextField();id.setText(String.*valueOf*(s.getId()));  
 JTextField distributor=**new** JTextField();distributor.setText(s.getDistributor());  
 JTextField producer=**new** JTextField();producer.setText(s.getProducer());  
 SpinnerModel n = **new** SpinnerNumberModel(0, 0, 500, 10);  
 JSpinner number = **new** JSpinner(n);number.setValue(s.getNumber());  
 Pane1.add(**new** JLabel(**"Id поставки:"**));Pane1.add(id);  
 Pane1.add(**new** JLabel(**"Выберите товар справа в таблице:"**));Pane1.add(**new** JLabel(**" "**));  
 Pane1.add(**new** JLabel(**"Поставщик:"**));Pane1.add(distributor);  
 Pane1.add(**new** JLabel(**"Производитель:"**));Pane1.add(producer);  
 Pane1.add(**new** JLabel(**"Количество:"**));Pane1.add(number);  
 JButton changeSale=**new** JButton(**"Изменить"**);  
 MyTableModelProduct Product=**new** MyTableModelProduct();  
 JTable tableProduct=**new** JTable(Product);  
 **int** e=Product.getRow(s.getProduct().getId());  
 tableProduct.setRowSelectionInterval(e,e);  
 changeSale.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 **int** row= tableProduct.getSelectedRow();  
 **if** (row==-1){  
 JOptionPane.*showMessageDialog*(dialog,  
 **" Вы ничего не выбрали"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 **return**;  
 }  
 Product p= **new** Product(  
 Integer.*parseInt*(String.*valueOf*( Product.getValueAt(tableProduct.getSelectedRow(),0))),  
 String.*valueOf*(Product.getValueAt(tableProduct.getSelectedRow(),1)),  
 Float.*valueOf*(String.*valueOf*( Product.getValueAt(tableProduct.getSelectedRow(),2)))  
 );  
 MyFrame.*Deliveries*.changeRow(s.getId(),**new** Deliveries(  
 Integer.*parseInt*(id.getText()),  
 String.*valueOf*(distributor.getText()),  
 p,  
 String.*valueOf*(producer.getText()),  
 Integer.*parseInt*(String.*valueOf*( number.getValue()))  
 ));  
 dialog.dispose();  
 MyFrame.*Deliveries*.update();  
 MyFrame.*Existence*.update();  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(dialog,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 id.setText(**""**);  
 distributor.setText(**""**);  
 number.setValue(0);  
 }  
 });  
 dialog.add(Pane1,BorderLayout.***CENTER***);  
 dialog.add(changeSale,BorderLayout.***SOUTH***);  
 dialog.add(**new** JScrollPane(tableProduct),BorderLayout.***EAST***);  
 dialog.add(**new** JLabel(**" "**),BorderLayout.***WEST***);  
 dialog.setSize(700,200);  
 dialog.setModal(**true**);  
 dialog.setVisible(**true**);  
 dialog.setLocationRelativeTo(**null**);  
 dialog.pack();  
 }  
 **public static void** ChangeSale(Sales s){  
 JDialog dialog=**new** JDialog();  
 dialog.setTitle(**"Изменение продажи"**);  
 JPanel Pane1=**new** JPanel(**new** GridLayout(5,2));  
 JTextField id=**new** JTextField();id.setText(String.*valueOf*(s.getId()));  
 JTextField data=**new** JTextField();data.setText(s.getData());  
 SpinnerModel n = **new** SpinnerNumberModel(0, 0, 500, 10);  
 JSpinner number = **new** JSpinner(n);number.setValue(s.getQuantity());  
 Pane1.add(**new** JLabel(**"Id продажи:"**));Pane1.add(id);  
 Pane1.add(**new** JLabel(**"Выберите товар справа в таблице:"**));Pane1.add(**new** JLabel(**" "**));  
 Pane1.add(**new** JLabel(**"Дата:"**));Pane1.add(data);  
 Pane1.add(**new** JLabel(**"Количество:"**));Pane1.add(number);  
 JButton changeSale=**new** JButton(**"Изменить"**);  
 MyTableModelProduct Product=**new** MyTableModelProduct();  
 JTable tableProduct=**new** JTable(Product);  
 **int** e=Product.getRow(s.getProduct().getId());  
 tableProduct.setRowSelectionInterval(e,e);  
 changeSale.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **try**{  
 **int** row= tableProduct.getSelectedRow();  
 **if** (row==-1){  
 JOptionPane.*showMessageDialog*(dialog,  
 **" Вы ничего не выбрали"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 **return**;  
 }  
 Product p= **new** Product(  
 Integer.*parseInt*(String.*valueOf*( Product.getValueAt(tableProduct.getSelectedRow(),0))),  
 String.*valueOf*(Product.getValueAt(tableProduct.getSelectedRow(),1)),  
 Float.*valueOf*(String.*valueOf*( Product.getValueAt(tableProduct.getSelectedRow(),2)))  
 );  
 MyFrame.*Sales*.changeRow(s.getId(),**new** Sales(  
 Integer.*parseInt*(id.getText()),  
 p,  
 String.*valueOf*(data.getText()),  
 Integer.*parseInt*(String.*valueOf*( number.getValue()))  
 ));  
 dialog.dispose();  
 MyFrame.*Existence*.update();  
 }**catch** (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(dialog,  
 **" Можно вводить только целые числа"**,  
 **"Ошибка"**,  
 JOptionPane.***ERROR\_MESSAGE***);  
 }  
 id.setText(**""**);  
 data.setText(**""**);  
 number.setValue(0);  
 }  
 });  
 dialog.add(Pane1,BorderLayout.***CENTER***);  
 dialog.add(changeSale,BorderLayout.***SOUTH***);  
 dialog.add(**new** JScrollPane(tableProduct),BorderLayout.***EAST***);  
 dialog.add(**new** JLabel(**" "**),BorderLayout.***WEST***);  
 dialog.setSize(700,200);  
 dialog.setModal(**true**);  
 dialog.setVisible(**true**);  
 dialog.setLocationRelativeTo(**null**);  
 dialog.pack();  
 }  
}

## **7.4 класс MyTableModelProduct**

**import** javax.swing.table.AbstractTableModel;  
**import** java.util.List;  
  
**public class** MyTableModelProduct **extends** AbstractTableModel {  
  
 **private** List<Product> **data**;  
  
 **public** MyTableModelProduct(){  
 DBWorker.*initDB*();  
 **data**=DBWorker.*selectProduct*();  
 DBWorker.*closeDB*();  
 }  
 **public void** update(){  
 DBWorker.*initDB*();  
 **data**=DBWorker.*selectProduct*();  
 DBWorker.*closeDB*();  
 **this**.fireTableDataChanged();  
 }  
 @Override  
 **public int** getRowCount() {  
 **return data**.size();  
 }  
 @Override  
 **public boolean** isCellEditable(**int** rowIndex, **int** columnIndex) {  
 **return false**;  
 }  
 @Override  
 **public int** getColumnCount() {  
 **return** 3;  
 }  
  
 @Override  
 **public** String getColumnName(**int** column) {  
 **switch** (column) {  
 **case** 0:  
 **return "ID продукта"**;  
 **case** 1:  
 **return "Название"**;  
 **case** 2:  
 **return "Цена"**;  
 }  
 **return null**;  
 }  
  
 @Override  
 **public** Class<?> getColumnClass(**int** columnIndex) {  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** Integer.**class**;  
 **case** 1:  
 **return** String.**class**;  
 **case** 2:  
 **return** Float.**class**;  
 }  
 **return null**;  
 }  
  
 @Override  
 **public** Object getValueAt(**int** rowIndex, **int** columnIndex) {  
 Product p= **data**.get(rowIndex);  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** p.getId();  
 **case** 1:  
 **return** p.getName();  
 **case** 2:  
 **return** p.getPrice();  
 }  
 **return null**;  
 }  
 **public void** addRow(Product p){  
 DBWorker.*initDB*();  
 DBWorker.*addProduct*(p);  
 DBWorker.*closeDB*();  
 update();  
 }  
 **public void** changeRow(**int** id,Product p){  
 DBWorker.*initDB*();  
 DBWorker.*changeProduct*(id,p);  
 DBWorker.*closeDB*();  
 update();  
 }  
 **public void** deleteRow(**int** id){  
 DBWorker.*initDB*();  
 DBWorker.*deleteProduct*(id);  
 DBWorker.*closeDB*();  
 update();  
 }  
 **public int** getRow(**int** id){  
 **int** y=0;  
 **for** (**int** i = 0; i < **data**.size(); i++) {  
 **if** (id==**data**.get(i).getId()){  
 y=i;  
 }  
 }  
 **return** y;  
 }  
 **public** Object[] getId() {  
 DBWorker.*initDB*();  
 List<Product> l=DBWorker.*selectProduct*();  
 String[] j=**new** String[l.size()];  
 **for** (**int** i = 0; i < l.size(); i++) {  
 j[i]=l.get(i).getId()+**" "**+l.get(i).getName();  
 }  
 DBWorker.*closeDB*();  
 **return** j;  
 }  
}

## **7.5 класс MyTableModelExistence**

**import** javax.swing.table.AbstractTableModel;  
**import** java.util.List;  
  
**public class** MyTableModelExistence **extends** AbstractTableModel {  
  
 **private** List<String[]> **data**;  
  
 **public** MyTableModelExistence(){  
 DBWorker.*initDB*();  
 **data**=DBWorker.*selectExistence*();  
 DBWorker.*closeDB*();  
 }  
 **public void** update(){  
 DBWorker.*initDB*();  
 **data**=DBWorker.*selectExistence*();  
 DBWorker.*closeDB*();  
 **this**.fireTableDataChanged();  
 }  
  
 @Override  
 **public int** getRowCount() {  
 **return data**.size();  
 }  
  
 @Override  
 **public int** getColumnCount() {  
 **return** 5;  
 }  
  
 @Override  
 **public** Object getValueAt(**int** rowIndex, **int** columnIndex) {  
 String[] c= **data**.get(rowIndex);  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** c[0];  
 **case** 1:  
 **return** c[1];  
 **case** 2:  
 **return** c[2];  
 **case** 3:  
 **return** c[3];  
 **case** 4:  
 **return** c[4];  
 }  
 **return null**;  
 }  
  
 @Override  
 **public** Class<?> getColumnClass(**int** columnIndex) {  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** Integer.**class**;  
 **case** 1:  
 **return** String.**class**;  
 **case** 2:  
 **return** Integer.**class**;  
 **case** 3:  
 **return** Integer.**class**;  
 **case** 4:  
 **return** Integer.**class**;  
 }  
 **return null**;  
 }  
  
 @Override  
 **public** String getColumnName(**int** column) {  
 **switch** (column) {  
 **case** 0:  
 **return "ID продукта"**;  
 **case** 1:  
 **return "Название продукта"**;  
 **case** 2:  
 **return "Поставлено"**;  
 **case** 3:  
 **return "Продано"**;  
 **case** 4:  
 **return "В наличие"**;  
 }  
 **return null**;  
 }  
}

## **7.6 класс MyTableModelSales**

**import** javax.swing.table.AbstractTableModel;  
**import** java.util.List;  
  
**public class** MyTableModelSales **extends** AbstractTableModel {  
 **private** List<Sales> **data**;  
  
 **public** MyTableModelSales(){  
 DBWorker.*initDB*();  
 **data**=DBWorker.*selectSales*();  
 DBWorker.*closeDB*();  
 }  
  
 **public void** update(){  
 DBWorker.*initDB*();  
 **data**=DBWorker.*selectSales*();  
 DBWorker.*closeDB*();  
 **this**.fireTableDataChanged();  
 }  
  
 @Override  
 **public int** getRowCount() {  
 **return data**.size();  
 }  
  
 @Override  
 **public int** getColumnCount() {  
 **return** 7;  
 }  
  
 @Override  
 **public** String getColumnName(**int** column) {  
 **switch** (column) {  
 **case** 0:  
 **return "ID продажи"**;  
 **case** 1:  
 **return "ID продукта"**;  
 **case** 2:  
 **return "Название"**;  
 **case** 3:  
 **return "Дата"**;  
 **case** 4:  
 **return "Цена"**;  
 **case** 5:  
 **return "Количество"**;  
 **case** 6:  
 **return "Сумма"**;  
 }  
 **return null**;  
 }  
 @Override  
 **public** Class<?> getColumnClass(**int** columnIndex) {  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** Integer.**class**;  
 **case** 1:  
 **return** Integer.**class**;  
 **case** 2:  
 **return** String.**class**;  
 **case** 3:  
 **return** String.**class**;  
 **case** 4:  
 **return** Float.**class**;  
 **case** 5:  
 **return** Integer.**class**;  
 **case** 6:  
 **return** Float.**class**;  
 }  
 **return null**;  
 }  
 @Override  
 **public** Object getValueAt(**int** rowIndex, **int** columnIndex) {  
 Sales c= **data**.get(rowIndex);  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** c.getId();  
 **case** 1:  
 **return** c.getProduct().getId();  
 **case** 2:  
 **return** c.getProduct().getName();  
 **case** 3:  
 **return** c.getData();  
 **case** 4:  
 **return** c.getProduct().getPrice();  
 **case** 5:  
 **return** c.getQuantity();  
 **case** 6:  
 **return** c.getSum();  
 }  
 **return null**;  
 }  
 **public void** addRow(Sales s){  
 DBWorker.*initDB*();  
 DBWorker.*addSales*(s);  
 DBWorker.*closeDB*();  
 update();  
 }  
 **public void** changeRow(**int** id,Sales s){  
 DBWorker.*initDB*();  
 DBWorker.*changeSales*(id,s);  
 DBWorker.*closeDB*();  
 update();  
 }  
 **public void** deleteRow(**int** id){  
 DBWorker.*initDB*();  
 DBWorker.*deleteSales*(id);  
 DBWorker.*closeDB*();  
 update();  
 }  
 @Override  
 **public boolean** isCellEditable(**int** rowIndex, **int** columnIndex) {  
 **return false**;  
 }  
}

## **7.7 класс MyTableModelDeliveries**

**import** javax.swing.table.AbstractTableModel;  
**import** java.util.List;  
  
**public class** MyTableModelDeliveries **extends** AbstractTableModel {  
  
 **private** List<Deliveries> **data**;  
  
 **public** MyTableModelDeliveries(){  
 DBWorker.*initDB*();  
 **data**=DBWorker.*selectDeliveries*();  
 DBWorker.*closeDB*();  
 }  
 **public void** update(){  
 DBWorker.*initDB*();  
 **data**=DBWorker.*selectDeliveries*();  
 DBWorker.*closeDB*();  
 **this**.fireTableDataChanged();  
 }  
  
 @Override  
 **public int** getRowCount() {  
 **return data**.size();  
 }  
  
 @Override  
 **public int** getColumnCount() {  
 **return** 6;  
 }  
  
 @Override  
 **public** Class<?> getColumnClass(**int** columnIndex) {  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** Integer.**class**;  
 **case** 1:  
 **return** String.**class**;  
 **case** 2:  
 **return** Integer.**class**;  
 **case** 3:  
 **return** String.**class**;  
 **case** 4:  
 **return** String.**class**;  
 **case** 5:  
 **return** Integer.**class**;  
 }  
 **return null**;  
 }  
  
 @Override  
 **public** Object getValueAt(**int** rowIndex, **int** columnIndex) {  
 Deliveries c= **data**.get(rowIndex);  
 **switch** (columnIndex) {  
 **case** 0:  
 **return** c.getId();  
 **case** 1:  
 **return** c.getDistributor();  
 **case** 2:  
 **return** c.getProduct().getId();  
 **case** 3:  
 **return** c.getProduct().getName();  
 **case** 4:  
 **return** c.getProducer();  
 **case** 5:  
 **return** c.getNumber();  
 }  
 **return null**;  
 }  
  
 @Override  
 **public** String getColumnName(**int** column) {  
 **switch** (column) {  
 **case** 0:  
 **return "ID поставки"**;  
 **case** 1:  
 **return "Поставщик"**;  
 **case** 2:  
 **return "ID продукта"**;  
 **case** 3:  
 **return "Название"**;  
 **case** 4:  
 **return "Производитель"**;  
 **case** 5:  
 **return "Количество"**;  
 }  
 **return null**;  
 }  
 **public void** addRow(Deliveries p){  
 DBWorker.*initDB*();  
 DBWorker.*addDeliveries*(p);  
 DBWorker.*closeDB*();  
 update();  
 }  
 @Override  
 **public boolean** isCellEditable(**int** rowIndex, **int** columnIndex) {  
 **return false**;  
 }  
 **public void** changeRow(**int** id,Deliveries p){  
 DBWorker.*initDB*();  
 DBWorker.*changeDeliveries*(id,p);  
 DBWorker.*closeDB*();  
 update();  
 }  
 **public void** deleteRow(**int** id){  
 DBWorker.*initDB*();  
 DBWorker.*deleteDeliveries*(id);  
 DBWorker.*closeDB*();  
 update();  
 }  
}

## **7.8 класс Product**

**public class** Product {  
 **private int id**;  
 **private** String **name**;  
 **private float price**;  
  
 **public** Product(**int** Id,String Name,**float** Price){  
 **this**.**id**=Id;  
 **this**.**name**=Name;  
 **this**.**price**=Price;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public float** getPrice() {  
 **return price**;  
 }  
  
 **public int** getId() {  
 **return id**;  
 }  
}

## **7.9 класс Sales**

**public class** Sales {  
  
 **private int id**;  
 **private** Product **product**;  
 **private** String **data**;  
 **private int quantity**;  
 **private float sum**;  
  
 **public** Sales(**int** Id,Product pr,String Data,**int** Quantity){  
 **this**.**id**=Id;  
 **this**.**product**=pr;  
 **this**.**data**=Data;  
 **this**.**quantity**=Quantity;  
 **this**.**sum**=Quantity\*pr.getPrice();  
 }  
  
 **public int** getId() {  
 **return id**;  
 }  
  
 **public** Product getProduct() {  
 **return product**;  
 }  
  
 **public int** getQuantity() {  
 **return quantity**;  
 }  
  
 **public** String getData() {  
 **return data**;  
 }  
  
 **public float** getSum() {  
 **return sum**;  
 }  
}

## **7.10 класс Deliveries**

**public class** Deliveries {  
 **private int id**;  
 **private** String **distributor**;  
 **private** Product **product**;  
 **private** String **producer**;  
 **private int number**;  
  
 Deliveries(**int** id,String distributor,Product product,String producer,**int** number){  
 **this**.**id**=id;  
 **this**.**distributor**=distributor;  
 **this**.**product**=product;  
 **this**.**producer**=producer;  
 **this**.**number**=number;  
 }  
  
 **public** Product getProduct() {  
 **return product**;  
 }  
  
 **public** String getDistributor() {  
 **return distributor**;  
 }  
  
 **public int** getNumber() {  
 **return number**;  
 }  
  
 **public int** getId() {  
 **return id**;  
 }  
  
 **public** String getProducer() {  
 **return producer**;  
 }  
}

## **7.11 класс DBWorker**

**import** java.sql.\*;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** DBWorker {  
 **public static final** String PATH\_TO\_DB\_FILE=**"MyShop.db"**;  
 **public static final** String URL=**"jdbc:sqlite:"**+PATH\_TO\_DB\_FILE;  
 **public static** Connection connection;  
  
 **public static void** initDB(){  
 **try** {  
 connection= DriverManager.getConnection(URL);  
 **if** (connection!=**null**){  
 DatabaseMetaData metaData=connection.getMetaData();  
 DBWorker.createDB();  
 }  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
  
 **public static void** closeDB(){  
 **try** {  
 connection.close();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
  
 **public static void** createDB(){  
 **try** {  
 Statement statement=connection.createStatement();  
 statement.execute(**"CREATE TABLE if not exists 'product' "** +  
 **"("** +  
 **" 'id\_product' int PRIMARY KEY,"** +  
 **" 'name' text,"** +  
 **" 'price' float );"**);  
 statement.execute(**"CREATE TABLE if not exists 'existence' "** +  
 **"("** +  
 **" 'id\_product' int PRIMARY KEY,"** +  
 **" 'name' text,"** +  
 **" 'number\_deliver' int,"** +  
 **" 'number\_sold' int,"** +  
 **" 'number' int );"**);  
 statement.execute(**"CREATE TABLE if not exists 'sales' "** +  
 **"("** +  
 **" 'id\_sales' int PRIMARY KEY,"** +  
 **" 'id\_product' int, "** +  
 **" 'name' text, "** +  
 **" 'data' text, "** +  
 **" 'price' float, "** +  
 **" 'quantity' int ,"** +  
 **" 'sum' float );"**);  
 statement.execute(**"CREATE TABLE if not exists 'deliveries' "** +  
 **"("** +  
 **" 'id\_deliveries' int PRIMARY KEY,"** +  
 **" 'distributor' text ,"** +  
 **" 'id\_product' int,"** +  
 **" 'name' text,"** +  
 **" 'producer' text ,"** +  
 **" 'number' int );"**);  
 statement.close();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
  
 **public static void** addProduct(Product p){  
 **try** {  
 PreparedStatement S = connection.prepareStatement(**"INSERT INTO product ('id\_product','name', 'price') "**+  
 **"VALUES(?,?,?)"**);  
 S.setObject(1,p.getId());  
 S.setObject(2,p.getName());  
 S.setObject(3,p.getPrice());  
 S.execute();  
 S.close();  
 PreparedStatement S1 = connection.prepareStatement(**"INSERT INTO existence ('id\_product','name', 'number\_deliver','number\_sold','number') "**+  
 **"VALUES(?,?,?,?,?)"**);  
 S1.setObject(1,p.getId());  
 S1.setObject(2,p.getName());  
 S1.setObject(3,0);  
 S1.setObject(4,0);  
 S1.setObject(5,0);  
 S1.execute();  
 S1.close();  
 updateExistence();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
 **public static void** addSales(Sales s){  
 **try** {  
 PreparedStatement S = connection.prepareStatement(**"INSERT INTO sales ('id\_sales','id\_product','name', 'data','price','quantity','sum') "**+  
 **"VALUES(?,?,?,?,?,?,?)"**);  
 S.setObject(1,s.getId());  
 S.setObject(2,s.getProduct().getId());  
 S.setObject(3,s.getProduct().getName());  
 S.setObject(4,s.getData());  
 S.setObject(5,s.getProduct().getPrice());  
 S.setObject(6,s.getQuantity());  
 S.setObject(7,s.getSum());  
 S.execute();  
 S.close();  
 PreparedStatement S1 = connection.prepareStatement(**"UPDATE existence SET 'number\_sold'=? WHERE id\_product =? "**);  
 S1.setObject(1,s.getQuantity());  
 S1.setObject(2,s.getProduct().getId());  
 S1.execute();  
 S1.close();  
 updateExistence();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
 **public static void** addDeliveries(Deliveries d){  
 **try** {  
 PreparedStatement S = *connection*.prepareStatement(**"INSERT INTO deliveries ('id\_deliveries','distributor','id\_product','name','producer' ,'number') "**+  
 **"VALUES(?,?,?,?,?,?)"**);  
 S.setObject(1,d.getId());  
 S.setObject(2,d.getDistributor());  
 S.setObject(3,d.getProduct().getId());  
 S.setObject(4,d.getProduct().getName());  
 S.setObject(5,d.getProducer());  
 S.setObject(6,d.getNumber());  
 S.execute();  
 S.close();  
 PreparedStatement S1 = *connection*.prepareStatement(**"UPDATE existence SET 'number\_deliver'=? WHERE id\_product =? "**);  
 S1.setObject(1,d.getNumber());  
 S1.setObject(2,d.getProduct().getId());  
 S1.execute();  
 S1.close();  
 *updateExistence*();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
  
 **public static void** deleteProduct(**int** id){  
 **try** {  
 PreparedStatement S = *connection*.prepareStatement(**"DELETE FROM product WHERE id\_product=?"**);  
 S.setObject(1,id);  
 S.execute();  
 S.close();  
 PreparedStatement S1 = *connection*.prepareStatement(**"DELETE FROM existence WHERE id\_product=?"**);  
 S1.setObject(1,id);  
 S1.execute();  
 S1.close();  
 PreparedStatement S2 = *connection*.prepareStatement(**"DELETE FROM sales WHERE id\_product=?"**);  
 S2.setObject(1,id);  
 S2.execute();  
 S2.close();  
 PreparedStatement S3 = *connection*.prepareStatement(**"DELETE FROM deliveries WHERE id\_product=?"**);  
 S3.setObject(1,id);  
 S3.execute();  
 S3.close();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
 **public static void** deleteDeliveries(**int** id){  
 **try** {  
 PreparedStatement S = *connection*.prepareStatement(**"DELETE FROM deliveries WHERE id\_deliveries=?"**);  
 S.setObject(1,id);  
 S.execute();  
 S.close();  
 *updateExistence*();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
 **public static void** deleteSales(**int** id){  
 **try** {  
 PreparedStatement S = *connection*.prepareStatement(**"DELETE FROM sales WHERE id\_sales=?"**);  
 S.setObject(1,id);  
 S.execute();  
 S.close();  
 *updateExistence*();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
  
 **public static void** deleteProduct(){  
 **try** {  
 Statement S = *connection*.createStatement();  
 S.execute(**"DELETE From product"**);  
 S.close();  
 Statement S1 = *connection*.createStatement();  
 S1.execute(**"DELETE From existence"**);  
 S1.close();  
 Statement S2 = *connection*.createStatement();  
 S2.execute(**"DELETE FROM sales"**);  
 S2.close();  
 Statement S3 = *connection*.createStatement();  
 S3.execute(**"DELETE FROM deliveries"**);  
 S3.close();  
  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
 **public static void** deleteDeliveries(){  
 **try** {  
 Statement S = *connection*.createStatement();  
 S.execute(**"DELETE From deliveries"**);  
 S.close();  
 *updateExistence*();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
 **public static void** deleteSales(){  
 **try** {  
 Statement S = *connection*.createStatement();  
 S.execute(**"DELETE From sales"**);  
 S.close();  
 *updateExistence*();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
  
 **public static** List selectProduct(){  
 **try** {  
 List<Product> List=**new** ArrayList();  
 Statement statmt = *connection*.createStatement();  
 ResultSet resSet = statmt.executeQuery(**"SELECT \* FROM product"**);  
 **while**(resSet.next())  
 {  
 Product p=**new** Product(  
 resSet.getInt(**"id\_product"**),  
 resSet.getString(**"name"**) ,  
 resSet.getFloat(**"price"**));  
 List.add(p);  
 }  
 statmt.close();  
 **return** List;  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 **return null**;  
 }  
 **public static** List selectDeliveries(){  
 **try** {  
 List<Deliveries> List=**new** ArrayList();  
 Statement S = connection.createStatement();  
 ResultSet resSet = S.executeQuery(**"SELECT \* FROM deliveries"**);  
 **while**(resSet.next())  
 {  
 Deliveries p=**new** Deliveries(  
 resSet.getInt(**"id\_deliveries"**),  
 resSet.getString(**"distributor"**) ,  
 **new** Product(resSet.getInt(**"id\_product"**),resSet.getString(**"name"**),0),  
 resSet.getString(**"producer"**),  
 resSet.getInt(**"number"**));  
 List.add(p);  
 }  
 S.close();  
 **return** List;  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 **return null**;  
 }  
 **public static** List selectSales(){  
 **try** {  
 List<Sales> List=**new** ArrayList();  
 Statement S = connection.createStatement();  
 ResultSet resSet = S.executeQuery(**"SELECT \* FROM sales"**);  
 **while**(resSet.next())  
 {  
 Sales p=**new** Sales(  
 resSet.getInt(**"id\_sales"**),  
 **new** Product(resSet.getInt(**"id\_product"**),resSet.getString(**"name"**), resSet.getFloat(**"price"**)),  
 resSet.getString(**"data"**),  
 resSet.getInt(**"quantity"**)  
 );  
 List.add(p);  
 }  
 S.close();  
 **return** List;  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 **return null**;  
 }  
  
 **public static** List selectExistence(){  
 **try** {  
 List<String[]> List=**new** ArrayList();  
 Statement S10 = connection.createStatement();  
 ResultSet resSet10 = S10.executeQuery(**"SELECT \* FROM existence "**);  
 **while**(resSet10.next())  
 {  
 String[] s= **new** String[5];  
 s[0]=String.valueOf(resSet10.getInt(**"id\_product"**));  
 s[1]=String.valueOf(resSet10.getString(**"name"**));  
 s[2]= String.valueOf(resSet10.getInt(**"number\_deliver"**));  
 s[3]= String.valueOf(resSet10.getInt(**"number\_sold"**));  
 s[4]= String.valueOf(resSet10.getInt(**"number"**));  
 *//System.out.println(s[0]+" "+s[1]+" "+s[2]+" "+s[3]+" "+s[4]);* List.add(s);  
 }  
 S10.close();  
 **return** List;  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 **return null**;  
 }  
  
 **public static void** updateExistence(){  
 **try** {  
 Statement S = connection.createStatement();  
 ResultSet resSet = S.executeQuery(**"SELECT id\_product,number FROM deliveries "**);  
 List<String[]> del=**new** ArrayList();  
 **while**(resSet.next())  
 {  
 String[] s= **new** String[2];  
 s[0]=String.valueOf(resSet.getInt(**"id\_product"**));  
 s[1]= String.valueOf(resSet.getInt(**"number"**));  
 del.add(s);  
 }  
 Statement S1 = connection.createStatement();  
 ResultSet resSet1 = S1.executeQuery(**"SELECT id\_product,quantity FROM sales "**);  
 List<String[]> sales=**new** ArrayList();  
 **while**(resSet1.next())  
 {  
 String[] s= **new** String[2];  
 s[0]=String.valueOf(resSet1.getInt(**"id\_product"**));  
 s[1]= String.valueOf(resSet1.getInt(**"quantity"**));  
 sales.add(s);  
 }  
 Statement S2 = connection.createStatement();  
 ResultSet resSet2 = S2.executeQuery(**"SELECT id\_product,name FROM product "**);  
 List<String[]> pr=**new** ArrayList();  
 **while**(resSet2.next())  
 {  
 String[] s= **new** String[2];  
 s[0]=String.valueOf(resSet2.getInt(**"id\_product"**));  
 s[1]=resSet2.getString(**"name"**);  
 pr.add(s);  
 }  
 **for** (**int** i = 0; i <pr.size() ; i++) {  
 String[] p=pr.get(i);  
 **int** number\_deliver=0;  
 **int** number\_sold=0;  
 **int** number=0;  
 **for** (**int** j = 0; j < del.size(); j++) {  
 String[] d=del.get(j);  
 **if**(d[0].equals(p[0])){  
 number\_deliver=number\_deliver+Integer.parseInt(d[1]);  
 }  
 }  
 **for** (**int** j = 0; j < sales.size(); j++) {  
 String[] s=sales.get(j);  
 **if**(s[0].equals(p[0])){  
 number\_sold=number\_sold+Integer.parseInt(s[1]);  
 }  
 }  
 number=number\_deliver-number\_sold;  
 PreparedStatement S6 = connection.prepareStatement(**"UPDATE existence SET number\_deliver=?,number\_sold=?,number=? WHERE id\_product =? "**);  
 S6.setObject(1,number\_deliver);  
 S6.setObject(2,number\_sold);  
 S6.setObject(3,number);  
 S6.setObject(4,p[0]);  
 S6.execute();  
 S6.close();  
 }  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
  
 **public static void** changeProduct(**int** id,Product p){  
 **try** {  
 PreparedStatement S = connection.prepareStatement(**"UPDATE product SET id\_product=? ,'name'=? ,'price'=? WHERE id\_product =? "**);  
 S.setObject(1,p.getId());  
 S.setObject(2,p.getName());  
 S.setObject(3,p.getPrice());  
 S.setObject(4,id);  
 S.execute();  
 S.close();  
 PreparedStatement S1 = connection.prepareStatement(**"UPDATE existence SET id\_product=? ,'name'=? WHERE id\_product =? "**);  
 S1.setObject(1,p.getId());  
 S1.setObject(2,p.getName());  
 S1.setObject(3,id);  
 S1.execute();  
 S1.close();  
 PreparedStatement S2 = connection.prepareStatement(**"UPDATE deliveries SET id\_product=? ,'name'=? WHERE id\_product =? "**);  
 S2.setObject(1,p.getId());  
 S2.setObject(2,p.getName());  
 S2.setObject(3,id);  
 S2.execute();  
 S2.close();  
 PreparedStatement S3 = connection.prepareStatement(**"UPDATE sales SET id\_product=? ,'name'=?,'sum'=? WHERE id\_product =? "**);  
 S3.setObject(1,p.getId());  
 S3.setObject(2,p.getName());  
 S3.setObject(3,p.getPrice());  
 S3.setObject(4,id);  
 S3.execute();  
 S3.close();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
 **public static void** changeDeliveries(**int** id, Deliveries d){  
 **try** {  
 PreparedStatement S = connection.prepareStatement(**"UPDATE deliveries SET id\_deliveries=?,'distributor'=?,id\_product=? ,'name'=?,'producer'=? ,'number'=? WHERE id\_deliveries =? "**);  
 S.setObject(1,d.getId());  
 S.setObject(2,d.getDistributor());  
 S.setObject(3,d.getProduct().getId());  
 S.setObject(4,d.getProduct().getName());  
 S.setObject(5,d.getProducer());  
 S.setObject(6,d.getNumber());  
 S.setObject(7,id);  
 S.execute();  
 S.close();  
 updateExistence();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
 **public static void** changeSales(**int** id,Sales s){  
 **try** {  
 PreparedStatement S = connection.prepareStatement(**"UPDATE sales SET id\_sales=?,id\_product=? ,'name'=?,'data'=? ,'price'=?,'quantity'=?,'sum'=? WHERE id\_sales =? "**);  
 S.setObject(1,s.getId());  
 S.setObject(2,s.getProduct().getId());  
 S.setObject(3,s.getProduct().getName());  
 S.setObject(4,s.getData());  
 S.setObject(5,s.getProduct().getPrice());  
 S.setObject(6,s.getQuantity());  
 S.setObject(7,s.getSum());  
 S.setObject(8,id);  
 S.execute();  
 S.close();  
 updateExistence();  
 } **catch** (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 }  
}

Список использованных источников

1. Объектно-ориентированное программирование. Метод. указания по

выполнению лабораторных работ / сост.: В.Л. Аршинский. – Иркутск : Изд-во

ИРНИТУ, 2017. – 24 c.