**Федеральное государственное бюджетное образовательное учреждение высшего образованиЯ**

**Казанский национальный исследовательский ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. А.Н. Туполева-каи**

**(КНИТУ-КАИ)**

**отделение среднего профессионального образования**

**КУРСОВАЯ РАБОТА**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Тема:** | Разработка desktop-приложения в трёх частях | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | |
|  |  | | | | | | | | |  | | | |
| **Руководитель** | | | | |  | | | | |  | |
| к.т.н. доцент каф. ПМИ | | | | | |  |  |  | П.И. Тутубалин | | | | |
| *(должность)* | | | | | |  | *(подпись)* |  | *(И.О. Фамилия)* | | | | |
| **Студент** | | |  | | | | | | | | | | |
| 4441 | | | | | |  |  |  | И.Р. Зарипов | | | | |
| *(группа)* | | | | | |  | *(подпись)* |  | *(И.О. Фамилия)* | | | | |
| **Направление** | | | | 09.02.03 Программирование в компьютерных системах | | | | | | | | | |
|  | | | | | (шифр и наименование направления) | | | | | | | |

**Казань 2019г**

ОГЛАВЛЕНИЕ

[Задание 3](#_Toc5864891)

[Руководство пользователя 4](#_Toc5864892)

[Исходные коды 12](#_Toc5864893)

# Задание

Часть 1.

Вывести на экран в графическом режиме графики двух функций на

интервале от *х*нач до *х*кон с шагом *dx*. Первая функция задана с помощью ряда

Тейлора. Графики должны быть плавными и различаться цветами.

Вариант 4:

Часть 2.

Вариант 17:

Дан файл, содержащий сведения об ассортименте обуви фирмы.

Структура записи файла: артикул, наименование, количество, стоимость

одной пары. Артикул начинается с буквы Д для дамской обуви, М для

мужской, П для детской.

Написать программу, выдающую информацию:

− о наличии и стоимости обуви артикула Х;

− ассортиментный список детской обуви с указанием наименования и имеющегося в наличии числа пар каждой модели;

− ассортиментный список дамской обуви;

− ассортиментный список мужской обуви.

Часть 3.

Вариант 32:

Автоматизированная система (АС) «Салон продажи сотовых телефонов».

# Руководство пользователя

Часть 1.

При запуске программы на окне сразу рисуются две функции: – желтым и – синим цветом (Рис. 1). В левом верхнем углу окна расположена панель управления значениями функции.

Обозначения:

«X\_нач» – начальная координата аргумента функций;

«X\_кон» – конечная координата аргумента функций;

«n\_max» – количество суммирований в ряду Тейлора для функции ;

«b» – аргумент, обозначающий смещение функции .

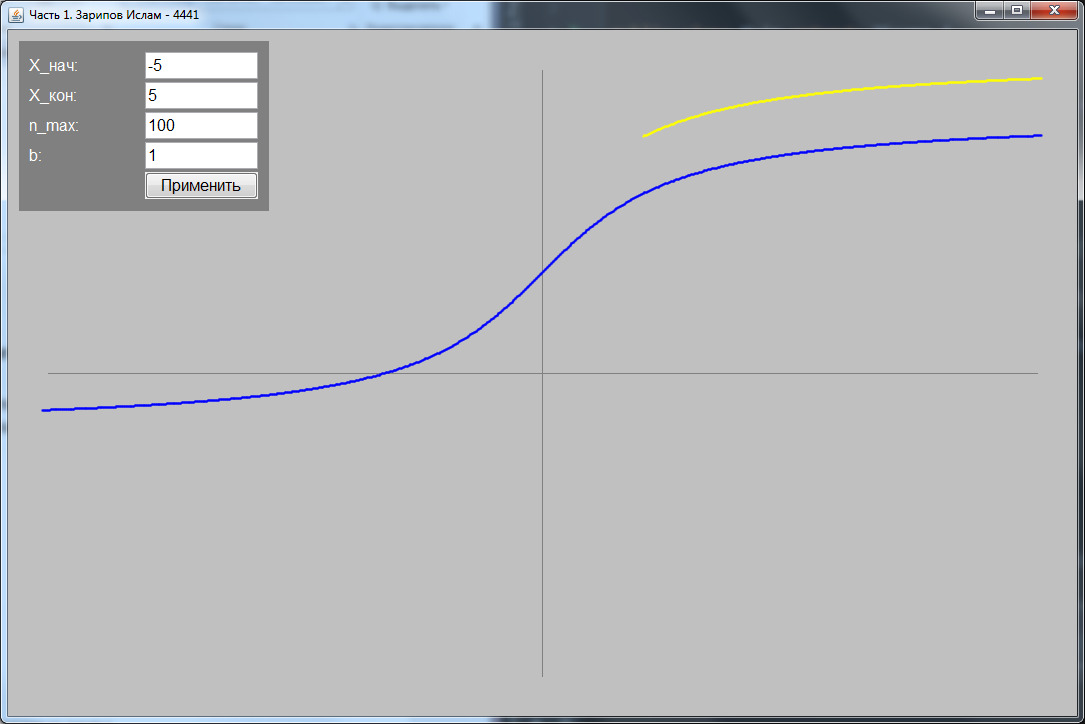


Рис. 1.1. Работа программы

Зададим значение «X\_нач» равное , а для «X\_кон» значение , также зададим смещение «b» на . После нажатия кнопки «Применить» графики будут нарисованы заново, с новыми параметрами (Рис. 1.2).

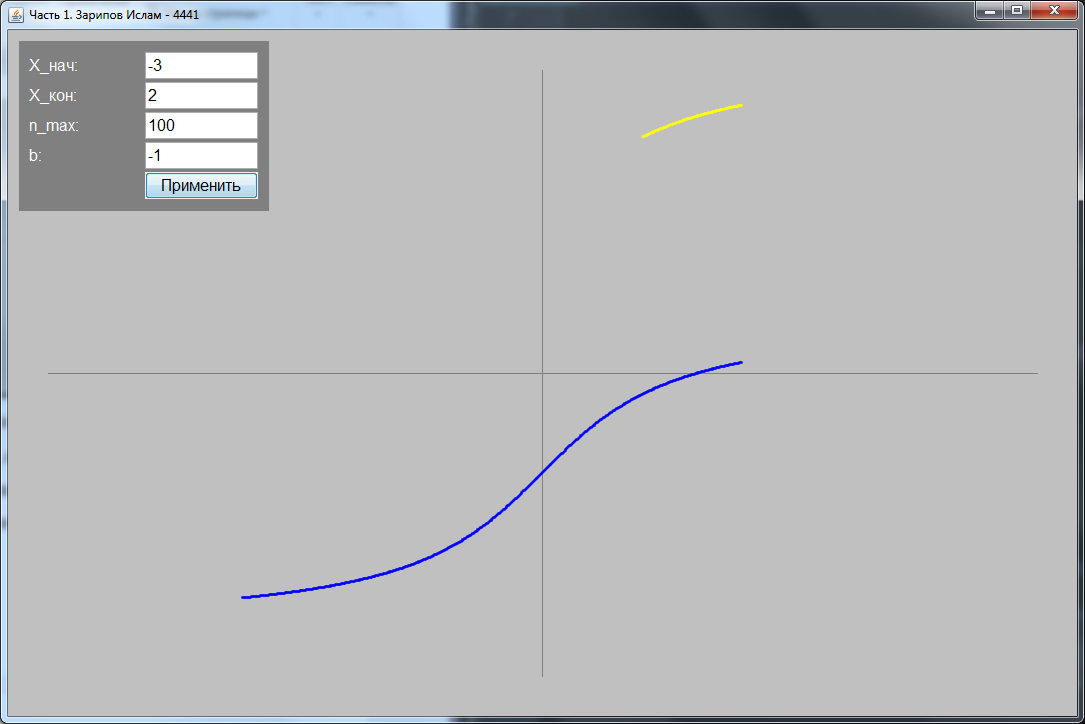


Рис. 1.2. Графики с новыми параметрами

Предусмотрена возможность масштабировать графики функций, для этого необходимо покрутить колесо мыши в необходимом для масштабирования направлении. Например, если покрутить колесо мыши на себя, то можно уменьшить масштаб графиков с сохранением пропорций отображения (Рис. 1.3).

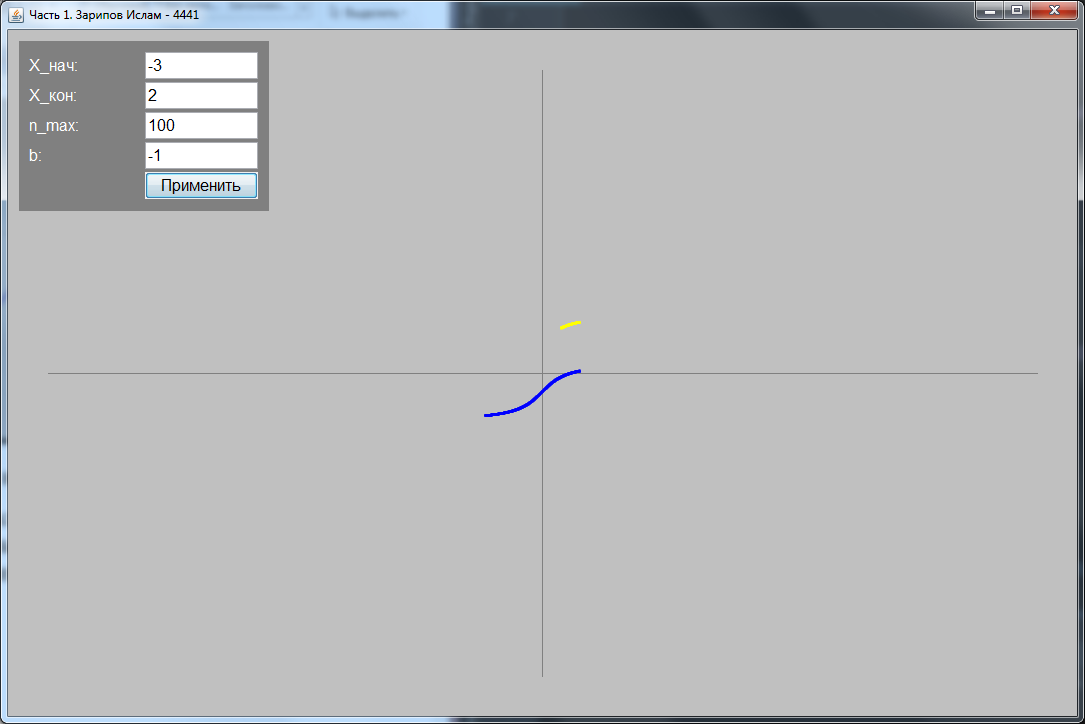


Рис. 1.3. Масштабирование графиков

Часть 2.

После запуска программы на окне отображаются четыре кнопки, и панель для вывода информации (Рис. 2.1).

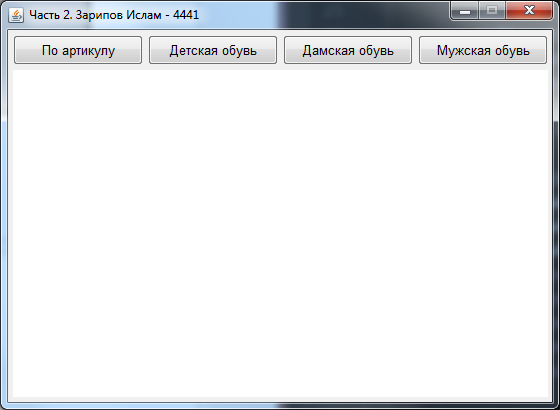


Рис. 2.1. Окно программы

При нажатии на копку «По артикулу» программа запросит ввести артикул, чтобы произвести поиск. Например, введем М004 (Рис. 2.2).

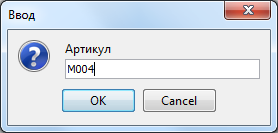


Рис. 2.2. Ввод артикула

После ввода найденный товар отображается в панели вывода информации (Рис. 2.3), если же введённый артикул не найден – отображается сообщение «Артикул не найден» (Рис. 2.4).

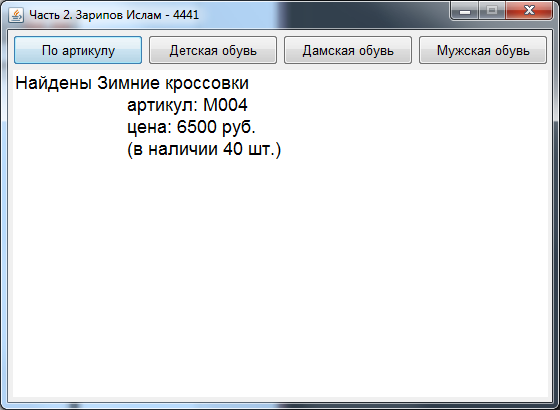


Рис. 2.3. Информация о найденном артикуле

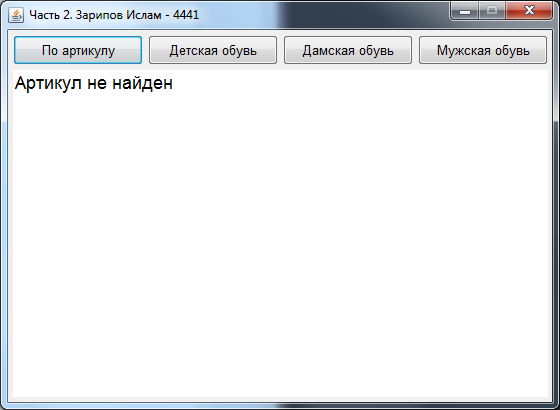


Рис. 2.4. Сообщение об ошибке

По нажатию на кнопки «Детская обувь», «Дамская обувь» или «Мужская обувь» отображается ассортиментный список соответствующей обуви с указанием наименования и имеющегося в наличии числа пар каждой модели (Рис. 2.5, 2.6 и 2.7).

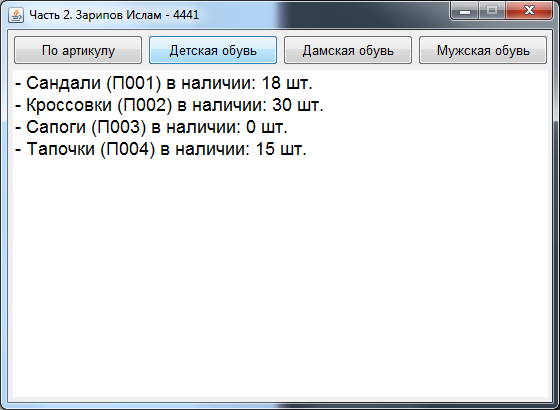


Рис. 2.5. Ассортиментный список детской обуви

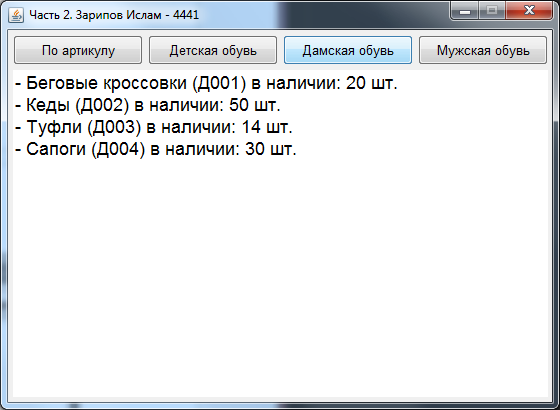


Рис. 2.6. Ассортиментный список дамской обуви

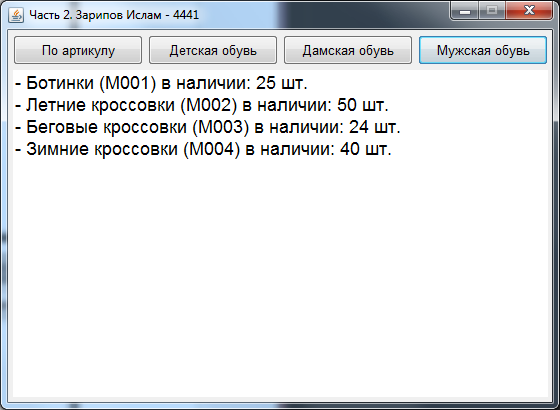


Рис. 2.6. Ассортиментный список мужской обуви

Часть 3.

В запущенном окне приложения отображается таблица заказов и копка «Добавить заказ» (Рис. 3.1), при нажатии на которую, отображается диалоговое окно добавления заказа (Рис. 3.2).

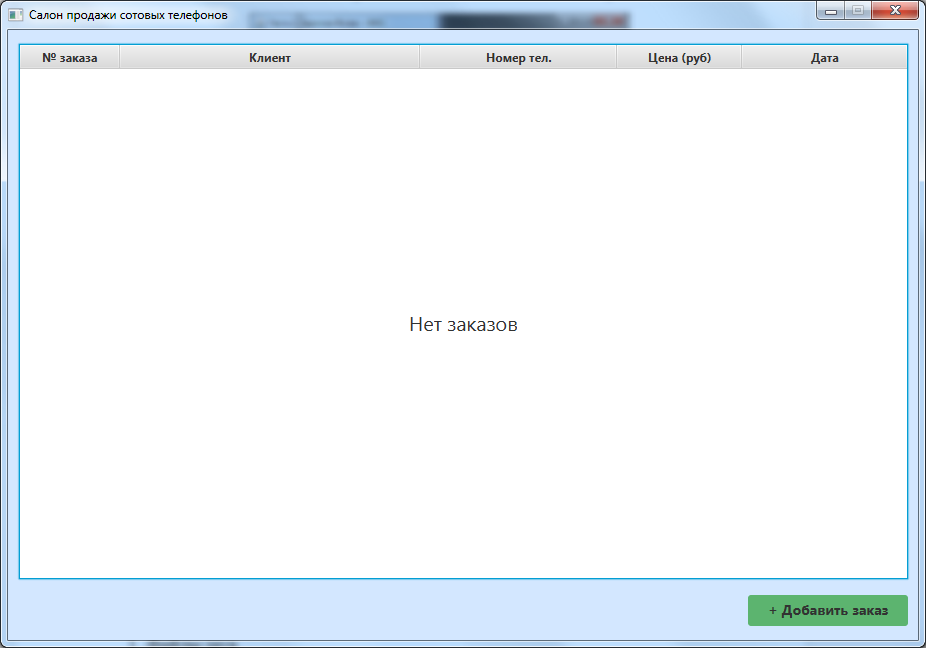


Рис. 3.1. Основное окно приложения

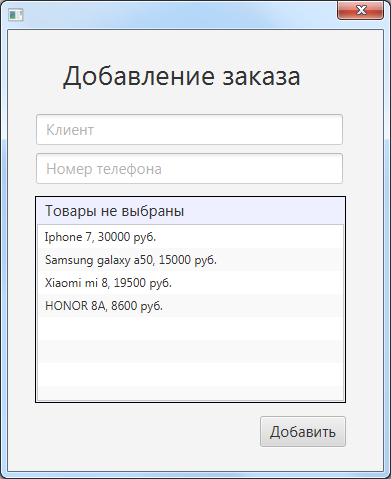


Рис. 3.2. Окно добавления заказа

Для того, чтобы добавить заказ нужно ввести имя клиента, его номер телефона, а также выбрать один или более товаров к заказу (Рис. 3.3).

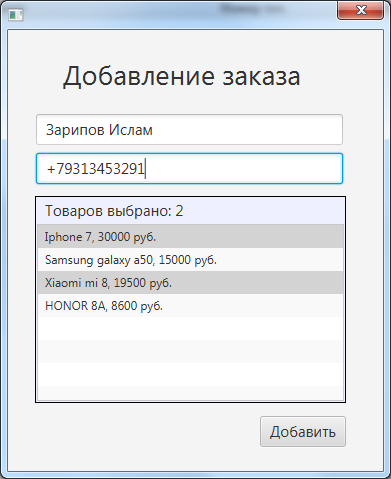


Рис. 3.3. Ввод информации о заказе

После нажатия кнопки «Добавить» новый заказ появится в списке всех заказов (Рис. 3.4), в столбце «Цена (руб)» будет указана итоговая цена заказа с учетом всех добавленных товаров, а в столбце «Дата» – дата и время добавления заказа.

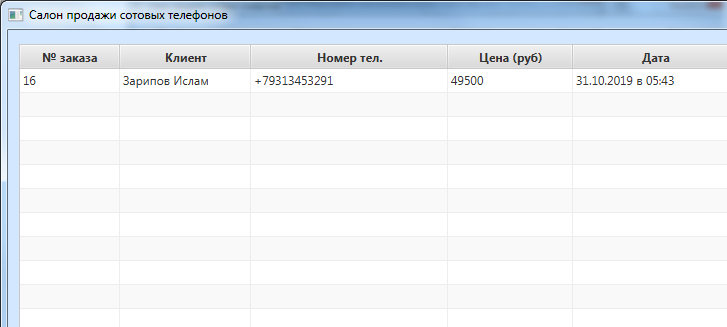


Рис. 3.4. Отображения нового заказа

Если совершить двойной клик на записи заказа, то откроется окно редактирования заказа (Рис. 3.5).

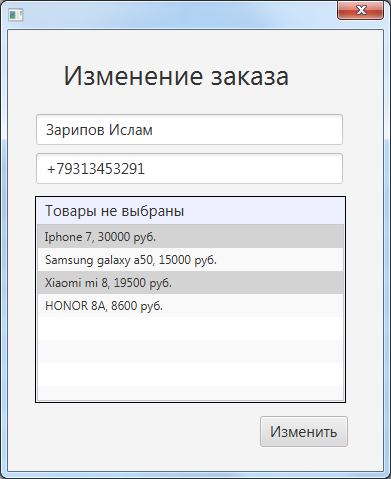


Рис. 3.5. Окно редактирования заказа

После нажатия кнопки «Изменить» заказ будет сохранен с новыми данными.

Для того, чтобы удалить заказ необходимо кликнуть правой клавишей мыши по записи заказа, и нажать на кнопку «Удалить» в контекстном меню (Рис. 3.6).

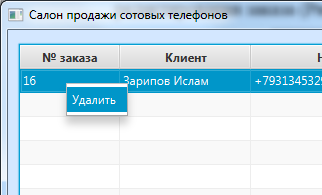


Рис. 3.6. Удаление заказа

# Исходные коды

1. Файлы java

package ru.inkrot.kit.part1;

import javax.swing.\*;

import javax.swing.border.EmptyBorder;

import java.awt.\*;

import java.text.NumberFormat;

public class Main extends JFrame {

private JPanel valuesPanel;

private JTextField xStartField, xEndField, nMaxField, bField;

private JButton apply;

private ChartsPanel chartsPanel;

private Font font = new Font("Arial", 0, 16);

Main() {

setTitle("Часть 1. Зарипов Ислам - 4441");

setSize(900, 800);

setExtendedState(JFrame.MAXIMIZED\_BOTH);

setLocationRelativeTo(null);

setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

setLayout(null);

initUI();

setVisible(true);

repaint();

}

private void initUI() {

valuesPanel = new JPanel();

valuesPanel.setLayout(new GridLayout(5,2, 3, 3));

valuesPanel.setBackground(Color.GRAY);

valuesPanel.setBorder(new EmptyBorder(10, 10, 10, 10));

valuesPanel.setBounds(11, 11, 250, 170);

add(valuesPanel);

valuesPanel.add(newLabel("X\_нач:"));

valuesPanel.add(xStartField = newField());

valuesPanel.add(newLabel("X\_кон:"));

valuesPanel.add(xEndField = newField());

valuesPanel.add(newLabel("n\_max:"));

valuesPanel.add(nMaxField = newField());

valuesPanel.add(newLabel("b:"));

valuesPanel.add(bField = newField());

valuesPanel.add(newEmptyComponent());

valuesPanel.add(apply = new JButton("Применить"));

apply.setFont(font);

apply.setFocusPainted(false);

apply.addActionListener(e -> applyValues());

chartsPanel = new ChartsPanel();

chartsPanel.setBounds(0, 0, getWidth(), getHeight());

xStartField.setText(String.valueOf(chartsPanel.xStart));

xEndField.setText(String.valueOf(chartsPanel.xEnd));

nMaxField.setText(String.valueOf(chartsPanel.nMax));

bField.setText(String.valueOf(chartsPanel.b));

add(chartsPanel);

}

private JLabel newLabel(String text) {

JLabel label = new JLabel(text);

label.setFont(font);

label.setForeground(Color.WHITE);

return label;

}

private JFormattedTextField newField() {

JFormattedTextField field = new JFormattedTextField(NumberFormat.getIntegerInstance());

field.setFont(font);

return field;

}

private JComponent newEmptyComponent() {

JComponent component = new JButton();

component.setVisible(false);

return component;

}

private int getValue(JTextField field) {

return Integer.parseInt(field.getText());

}

private void applyValues() {

int xStart, xEnd, nMax, b;

try {

xStart = getValue(xStartField);

xEnd = getValue(xEndField);

nMax = getValue(nMaxField);

b = getValue(bField);

chartsPanel.xStart = xStart;

chartsPanel.xEnd = xEnd;

chartsPanel.nMax = nMax;

chartsPanel.b = b;

repaint();

} catch (Exception e) { }

}

@Override

public void paint(Graphics g) {

super.paint(g);

//chartsPanel.setSize(Main.this.getWidth() - 22, Main.this.getHeight() - 56);

chartsPanel.setSize(Main.this.getWidth() - 16, Main.this.getHeight() - 38);

}

public static void main(String[] args) throws ClassNotFoundException, UnsupportedLookAndFeelException, InstantiationException, IllegalAccessException {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

new Main();

}

class ChartsPanel extends JPanel {

int xStart = -5;

int xEnd = 5;

int nMax = 100;

int b = 1;

int sc = 100;

ChartsPanel() {

addMouseWheelListener(e -> {

if (e.getWheelRotation() < 0) sc += 3;

else sc -= sc > 1 ? 3 : 0;

Main.this.repaint();

});

}

private int rX(double xCoord) {

return (int) (getWidth() / 2 + xCoord \* sc);

}

private int rY(double yCoord) {

return (int) (getHeight() / 2 - yCoord \* sc);

}

private double y(double x) {

if (x > 1) {

double y = Math.PI;

for (int n = 0; n < nMax; n++) {

y += Math.pow(-1, n + 1) / ((2 \* n + 1) \* Math.pow(x, 2 \* n + 1));

}

return y;

} else throw new IllegalArgumentException("x must be more than 1");

}

private double z(double x) {

return Math.atan(x) + (double) b;

}

@Override

public void paint(Graphics g) {

int w = getWidth(), h = getHeight();

g.setColor(Color.LIGHT\_GRAY);

g.fillRect(0, 0, w, h);

g.setColor(Color.GRAY);

((Graphics2D) g).setStroke(new BasicStroke(1));

g.drawLine(w / 2, 40, w / 2, h - 40);

g.drawLine(40, h / 2, w - 40, h / 2);

Graphics2D g2 = (Graphics2D) g;

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

g2.setRenderingHint(RenderingHints.KEY\_TEXT\_ANTIALIASING, RenderingHints.VALUE\_TEXT\_ANTIALIAS\_ON);

g2.setStroke(new BasicStroke(2));

int rMaxX = w / 2;

double dx = ((xEnd - xStart) / (double) rMaxX);

for (double x = xStart + dx; x <= xEnd; x += dx) {

if (x - dx > 1) {

g.setColor(Color.YELLOW);

g.drawLine(rX(x - dx),

rY(y(x - dx)),

rX(x),

rY(y(x))

);

}

g.setColor(Color.BLUE);

g.drawLine(rX(x - dx),

rY(z(x - dx)),

rX(x),

rY(z(x))

);

}

}

}

}

package ru.inkrot.kit.part2;

import javax.swing.\*;

import java.awt.\*;

import java.io.\*;

import java.util.ArrayList;

public class Main extends JFrame {

private ArrayList<Shoes> shoes = new ArrayList<>();

private Font buttonsFont = new Font("Arial", 0, 13);

private Font resultFont = new Font("Arial", 0, 18);

private JTextArea resultBox;

Main() {

loadDataFile("data.txt");

createUI();

}

private void createUI() {

setTitle("Часть 2. Зарипов Ислам - 4441");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setSize(550, 400);

setLocationRelativeTo(null);

setLayout(null);

setResizable(false);

JButton checkByVendorCodeButton, showBabyShoesButton, showFemaleShoesButton, showMaleShoesButton;

add(checkByVendorCodeButton = newButton(5, 5, 130, 30, "По артикулу"));

add(showBabyShoesButton = newButton(140, 5, 130, 30, "Детская обувь"));

add(showFemaleShoesButton = newButton(275, 5, 130, 30, "Дамская обувь"));

add(showMaleShoesButton = newButton(410, 5, 130, 30, "Мужская обувь"));

resultBox = new JTextArea();

resultBox.setBounds(5, 40, 535, 327);

resultBox.setEditable(false);

resultBox.setFont(resultFont);

add(resultBox);

checkByVendorCodeButton.addActionListener(e -> checkByVendorCode());

showBabyShoesButton.addActionListener(e -> showBabyShoes());

showFemaleShoesButton.addActionListener(e -> showFemaleShoes());

showMaleShoesButton.addActionListener(e -> showMaleShoes());

setVisible(true);

}

private JButton newButton(int x, int y, int w, int h, String text) {

JButton button = new JButton(text);

button.setBounds(x, y, w, h);

button.setFont(buttonsFont);

button.setFocusPainted(false);

return button;

}

private void loadDataFile(String dataFilePath) {

try {

BufferedReader br = new BufferedReader(new FileReader(new File(dataFilePath)));

String line;

while ((line = br.readLine()) != null)

shoes.add(parseDataLine(line));

} catch (IOException e) {

e.printStackTrace();

}

}

private Shoes parseDataLine(String line) {

String arr[] = line.split(",");

return new Shoes(arr[0], arr[1], Integer.valueOf(arr[2]), Integer.valueOf(arr[3]));

}

private void checkByVendorCode() {

String vendorCode = JOptionPane.showInputDialog(this,"Артикул", "Ввод", JOptionPane.QUESTION\_MESSAGE);

for (Shoes s: shoes) {

if (s.vendorCode.equals(vendorCode)) {

String count = s.count > 0 ? "(в наличии " + s.count + " шт.)" : "(нет в наличии)";

resultBox.setText("Найдены " + s.name + "\n\tартикул: " + vendorCode + "\n\tцена: " + s.price + " руб.\n\t" + count);

return;

}

}

resultBox.setText("Артикул не найден");

}

private void showByShoesVendorCodeType(char type) {

StringBuilder sb = new StringBuilder();

for (Shoes s : shoes) {

if (s.vendorCode.charAt(0) == type) sb.append(s).append('\n');

}

resultBox.setText(sb.toString());

}

private void showBabyShoes() {

showByShoesVendorCodeType('П');

}

private void showFemaleShoes() {

showByShoesVendorCodeType('Д');

}

private void showMaleShoes() {

showByShoesVendorCodeType('М');

}

public static void main(String[] args) throws Exception {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

new Main();

}

class Shoes {

String vendorCode, name;

int count, price;

public Shoes(String vendorCode, String name, int count, int price) {

this.vendorCode = vendorCode;

this.name = name;

this.count = count;

this.price = price;

}

@Override

public String toString() {

return "- " + name + " (" + vendorCode + ") в наличии: " + count + " шт.";

}

}

}

package ru.inkrot.kit.part3;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

import ru.inkrot.kit.part3.model.DataBase;

public class Main extends Application {

public static Stage stage;

@Override

public void start(Stage primaryStage) {

stage = primaryStage;

try {

DataBase.connect();

Parent root = FXMLLoader.load(getClass().getResource("view/main.fxml"));

stage.setTitle("Салон продажи сотовых телефонов");

stage.setScene(new Scene(root, 900, 600));

stage.setResizable(false);

stage.show();

} catch (Exception e) {

e.printStackTrace();

System.exit(0);

}

}

public static void main(String[] args) {

launch(args);

}

}

package ru.inkrot.kit.part3.model;

import java.sql.\*;

import java.util.List;

public class DataBase {

public static Connection connection;

public static final String DB\_FILE\_NAME = "database.db";

public static boolean connect() throws ClassNotFoundException, SQLException {

connection = null;

Class.forName("org.sqlite.JDBC");

String url = "jdbc:sqlite:" + DB\_FILE\_NAME;

connection = DriverManager.getConnection(url);

return true;

}

private static ResultSet selectTable(String table, String selector) {

try {

PreparedStatement stmt = connection.prepareStatement("SELECT " + selector + " FROM " + table);

return stmt.executeQuery();

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

private static ResultSet selectTable(String table) {

return selectTable(table, "\*");

}

public static ResultSet getOrders() {

return selectTable("orders", "id, customer, customer\_tel, COALESCE((SELECT SUM(p.price) FROM phones p WHERE p.id IN (SELECT phone\_id FROM orders\_phones WHERE order\_id=orders.id)), 0) AS price, timestamp");

}

public static ResultSet getPhones() {

return selectTable("phones");

}

public static ResultSet getPhones(int orderId) {

try {

PreparedStatement stmt = connection.prepareStatement("SELECT \* FROM phones WHERE id IN (SELECT phone\_id FROM orders\_phones WHERE order\_id = " + orderId + ")");

return stmt.executeQuery();

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public static void insertOrder(String customer, String customerTel, List<Phone> phones) {

try {

connection.setAutoCommit(false);

PreparedStatement stmt = connection.prepareStatement("INSERT INTO orders(customer, customer\_tel, timestamp) VALUES(?, ?, ?)");

stmt.setString(1, customer);

stmt.setString(2, customerTel);

stmt.setLong(3, System.currentTimeMillis());

stmt.execute();

ResultSet rs = connection.createStatement().executeQuery("SELECT last\_insert\_rowid() as id");

int insertedId = rs.getInt("id");

for (Phone phone : phones) {

addPhoneToOrder(insertedId, phone.getId());

}

connection.commit();

} catch (SQLException e) {

try {

connection.rollback();

} catch (SQLException e1) {

e1.printStackTrace();

}

e.printStackTrace();

}

}

public static Order getOrder(int id) {

try {

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery("SELECT id, customer, customer\_tel, COALESCE((SELECT SUM(p.price) FROM phones p WHERE p.id IN (SELECT phone\_id FROM orders\_phones WHERE order\_id=orders.id)), 0) AS price, timestamp FROM orders WHERE id=" + id);

return new Order(

rs.getInt("id"),

rs.getString("customer"),

rs.getString("customer\_tel"),

rs.getInt("price"),

rs.getTimestamp("timestamp")

);

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public static void updateOrder(int id, String customer, String customerTel, List<Phone> phones) {

try {

connection.setAutoCommit(false);

PreparedStatement stmt = connection.prepareStatement("UPDATE orders SET customer=?, customer\_tel=? WHERE id=" + id);

stmt.setString(1, customer);

stmt.setString(2, customerTel);

stmt.execute();

connection.createStatement().executeUpdate("DELETE FROM orders\_phones WHERE order\_id=" + id);

for (Phone phone : phones) {

addPhoneToOrder(id, phone.getId());

}

connection.commit();

} catch (SQLException e) {

try {

connection.rollback();

} catch (SQLException e1) {

e1.printStackTrace();

}

e.printStackTrace();

}

}

public static void deleteOrder(int id) {

try {

connection.setAutoCommit(false);

connection.createStatement().executeUpdate("DELETE FROM orders\_phones WHERE order\_id=" + id);

PreparedStatement stmt = connection.prepareStatement("DELETE FROM orders WHERE id=?");

stmt.setInt(1, id);

stmt.execute();

connection.commit();

} catch (SQLException e) {

try {

connection.rollback();

} catch (SQLException e1) {

e1.printStackTrace();

}

e.printStackTrace();

}

}

private static void addPhoneToOrder(int orderId, int phoneId) {

try {

PreparedStatement stmt = connection.prepareStatement("INSERT INTO orders\_phones(order\_id, phone\_id) VALUES(?, ?)");

stmt.setInt(1, orderId);

stmt.setInt(2, phoneId);

stmt.execute();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

package ru.inkrot.kit.part3.model;

public class DataBaseEntity {

private int id;

public DataBaseEntity(int id) {

this.id = id;

}

public int getId() {

return id;

}

}

package ru.inkrot.kit.part3.model;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Timestamp;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.List;

public class Order extends DataBaseEntity {

private static final SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy в HH:mm");

private String customer, customerTel;

private Timestamp timestamp;

private int price;

private List<Phone> phones = new ArrayList<>();

public Order(int id, String customer, String customerTel, int price, Timestamp timestamp) {

super(id);

this.customer = customer;

this.customerTel = customerTel;

this.price = price;

this.timestamp = timestamp;

ResultSet rs = DataBase.getPhones(id);

try {

while(rs.next())

this.phones.add(

new Phone(

rs.getInt("id"),

rs.getString("name"),

rs.getInt("price")

)

);

} catch (SQLException e) {

e.printStackTrace();

}

}

public String getCustomer() {

return customer;

}

public String getCustomerTel() {

return customerTel;

}

public int getPrice() {

return price;

}

public String getDate() {

if (timestamp == null) return "";

return dateFormat.format(timestamp);

}

public List<Phone> getPhones() {

return phones;

}

}

package ru.inkrot.kit.part3.model;

import java.util.Objects;

public class Phone extends DataBaseEntity {

String name;

int price;

public Phone(int id, String name, int price) {

super(id);

this.name = name;

this.price = price;

}

@Override

public String toString() {

return name + ", " + price + " руб.";

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Phone phone = (Phone) o;

return getId() == phone.getId() && price == phone.price &&

Objects.equals(name, phone.name);

}

@Override

public int hashCode() {

return Objects.hash(name, price);

}

}

package ru.inkrot.kit.part3.controller;

import javafx.beans.binding.Bindings;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.stage.Modality;

import javafx.stage.Stage;

import javafx.util.Callback;

import ru.inkrot.kit.part3.model.DataBase;

import ru.inkrot.kit.part3.model.Order;

import java.io.IOException;

import java.sql.ResultSet;

import java.sql.SQLException;

public class MainController {

@FXML

private TableView<Order> ordersTable;

@FXML

private TableColumn<Object, Object> ordersTableIdColumn;

@FXML

private TableColumn<Object, Object> ordersTableCustomerColumn;

@FXML

private TableColumn<Object, Object> ordersTableCustomerTelColumn;

@FXML

private TableColumn<Object, Object> ordersTablePriceColumn;

@FXML

private TableColumn<Object, Object> ordersTableDateColumn;

@FXML

private Button addOrderButton;

public Stage dialogStage;

public static Order selectedOrder;

public static int dialogType;

public static final int ADD\_DIALOG = 1;

public static final int EDIT\_DIALOG = 2;

@FXML

void initialize() {

ordersTable.setRowFactory(tableViewCallback);

ordersTableIdColumn.setCellValueFactory(new PropertyValueFactory<>("id"));

ordersTableCustomerColumn.setCellValueFactory(new PropertyValueFactory<>("customer"));

ordersTableCustomerTelColumn.setCellValueFactory(new PropertyValueFactory<>("customerTel"));

ordersTablePriceColumn.setCellValueFactory(new PropertyValueFactory<>("price"));

ordersTableDateColumn.setCellValueFactory(new PropertyValueFactory<>("date"));

addOrderButton.setOnAction((event -> showOrderDialog(ADD\_DIALOG)));

OrderController.mainController = this;

updateOrdersTable();

}

Callback<TableView<Order>, TableRow<Order>> tableViewCallback = tableView -> {

final TableRow<Order> row = new TableRow<>();

final ContextMenu rowMenu = new ContextMenu();

MenuItem removeItem = new MenuItem("Удалить");

removeItem.setOnAction(event -> deleteOrder(row.getItem().getId()));

rowMenu.getItems().addAll(removeItem);

row.contextMenuProperty().bind(

Bindings.when(Bindings.isNotNull(row.itemProperty()))

.then(rowMenu)

.otherwise((ContextMenu)null));

row.setOnMouseClicked(event -> {

if (event.getClickCount() == 2 && (! row.isEmpty()) ) {

selectedOrder = DataBase.getOrder(row.getItem().getId());

showOrderDialog(EDIT\_DIALOG);

}

});

return row;

};

private void deleteOrder(int id) {

DataBase.deleteOrder(id);

updateOrdersTable();

}

public void updateOrdersTable() {

ordersTable.getItems().clear();

ResultSet rs = DataBase.getOrders();

try {

while(rs.next())

ordersTable.getItems().add(

new Order(

rs.getInt("id"),

rs.getString("customer"),

rs.getString("customer\_tel"),

rs.getInt("price"),

rs.getTimestamp("timestamp")

)

);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void showOrderDialog(int type) {

dialogType = type;

FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("../view/order\_dialog.fxml"));

try {

Parent parent = fxmlLoader.load();

dialogStage = new Stage();

dialogStage.initModality(Modality.APPLICATION\_MODAL);

dialogStage.setScene(new Scene(parent, 365, 431));

dialogStage.setResizable(false);

dialogStage.showAndWait();

} catch (IOException e) {

e.printStackTrace();

}

}

}

package ru.inkrot.kit.part3.controller;

import javafx.fxml.FXML;

import javafx.scene.control.\*;

import ru.inkrot.kit.part3.model.DataBase;

import ru.inkrot.kit.part3.model.Order;

import ru.inkrot.kit.part3.model.Phone;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.List;

import static ru.inkrot.kit.part3.controller.MainController.EDIT\_DIALOG;

import static ru.inkrot.kit.part3.controller.MainController.selectedOrder;

public class OrderController {

@FXML

private Label orderDialogLabel;

@FXML

private TextField customerField;

@FXML

private TextField customerTelField;

@FXML

private ListView<Phone> phonesList;

@FXML

private Label phonesListLabel;

@FXML

private Button addOrderButton;

public static MainController mainController;

@FXML

void initialize() {

ResultSet rs = DataBase.getPhones();

try {

while(rs.next())

phonesList.getItems().add(

new Phone(

rs.getInt("id"),

rs.getString("name"),

rs.getInt("price")

)

);

} catch (SQLException e) {

e.printStackTrace();

}

phonesList.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);

phonesList.setOnMouseClicked(event -> checkSelected());

if (MainController.dialogType == EDIT\_DIALOG) {

Order order = selectedOrder;

customerField.setText(order.getCustomer());

customerTelField.setText(order.getCustomerTel());

for (Phone phone : order.getPhones())

phonesList.getSelectionModel().select(phone);

orderDialogLabel.setText("Изменение заказа");

addOrderButton.setText("Изменить");

addOrderButton.setOnAction((event -> editOrder()));

} else {

addOrderButton.setOnAction((event -> addOrder()));

}

}

private void checkSelected() {

int count = phonesList.getSelectionModel().getSelectedItems().size();

if (count > 0) phonesListLabel.setText("Товаров выбрано: " + count);

else phonesListLabel.setText("Товары не выбраны");

}

private void addOrder() {

String customer = customerField.getText();

String customerTel = customerTelField.getText();

List<Phone> selectedItems = phonesList.getSelectionModel().getSelectedItems();

if (customer.length() > 0 && customerTel.length() > 0 && selectedItems.size() > 0) {

DataBase.insertOrder(customer, customerTel, selectedItems);

mainController.dialogStage.getScene().getWindow().hide();

mainController.updateOrdersTable();

}

}

private void editOrder() {

String customer = customerField.getText();

String customerTel = customerTelField.getText();

List<Phone> selectedItems = phonesList.getSelectionModel().getSelectedItems();

if (customer.length() > 0 && customerTel.length() > 0 && selectedItems.size() > 0) {

DataBase.updateOrder(selectedOrder.getId(), customer, customerTel, selectedItems);

mainController.dialogStage.getScene().getWindow().hide();

mainController.updateOrdersTable();

}

}

}

2. Файлы fxml

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TableColumn?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="622.0" prefWidth="910.0" style="-fx-background-color: #d3e7ff;" xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/8.0.172-ea" fx:controller="ru.inkrot.kit.part3.controller.MainController">

<TableView fx:id="ordersTable" editable="true" layoutX="11.0" layoutY="14.0" prefHeight="535.0" prefWidth="889.0">

<placeholder>

<Label text="Нет заказов">

<font>

<Font size="20.0" />

</font>

</Label>

</placeholder>

<columns>

<TableColumn fx:id="ordersTableIdColumn" prefWidth="100.0" text="№ заказа" />

<TableColumn fx:id="ordersTableCustomerColumn" prefWidth="300.0" text="Клиент" />

<TableColumn fx:id="ordersTableCustomerTelColumn" prefWidth="197.0" text="Номер тел." />

<TableColumn fx:id="ordersTablePriceColumn" prefWidth="125.0" text="Цена (руб)" />

<TableColumn fx:id="ordersTableDateColumn" prefWidth="166.0" text="Дата" />

</columns>

</TableView>

<Button fx:id="addOrderButton" layoutX="740.0" layoutY="565.0" mnemonicParsing="false" prefHeight="29.0" prefWidth="160.0" style="-fx-background-color: #5cb46f;" text="+ Добавить заказ">

<font>

<Font name="System Bold" size="14.0" />

</font>

</Button>

</AnchorPane>

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.ListView?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<AnchorPane focusTraversable="true" prefHeight="431.0" prefWidth="365.0" xmlns="http://javafx.com/javafx/8.0.172-ea" xmlns:fx="http://javafx.com/fxml/1" fx:controller="ru.inkrot.kit.part3.controller.OrderController">

<Label fx:id="orderDialogLabel" layoutX="55.0" layoutY="26.0" prefHeight="31.0" prefWidth="246.0"

text="Добавление заказа">

<font>

<Font size="27.0" />

</font>

</Label>

<TextField fx:id="customerField" focusTraversable="false" layoutX="28.0" layoutY="84.0" prefHeight="31.0" prefWidth="307.0" promptText="Клиент">

<font>

<Font size="15.0" />

</font>

</TextField>

<TextField fx:id="customerTelField" focusTraversable="false" layoutX="28.0" layoutY="123.0" prefHeight="31.0" prefWidth="307.0" promptText="Номер телефона">

<font>

<Font size="15.0" />

</font>

</TextField>

<AnchorPane layoutX="27.0" layoutY="166.0" prefHeight="162.0" prefWidth="307.0" style="-fx-border-style: solid; -fx-border-width: 1px; -fx-border-color: black; -fx-background-color: #eff0ff;">

<children>

<Label fx:id="phonesListLabel" layoutX="10.0" layoutY="4.0" prefHeight="21.0" prefWidth="152.0"

text="Товары не выбраны">

<font>

<Font size="15.0" />

</font>

</Label>

<ListView fx:id="phonesList" focusTraversable="false" layoutX="2.0" layoutY="28.0" prefHeight="177.0" prefWidth="307.0" />

</children>

</AnchorPane>

<Button fx:id="addOrderButton" focusTraversable="false" layoutX="252.0" layoutY="386.0" mnemonicParsing="false" text="Добавить">

<font>

<Font size="15.0" />

</font>

</Button>

</AnchorPane>

3. Файл txt

Д001,Беговые кроссовки,20,5000

Д002,Кеды,50,3000

Д003,Туфли,14,8000

Д004,Сапоги,30,2000

М001,Ботинки,25,7000

М002,Летние кроссовки,50,3000

М003,Беговые кроссовки,24,5000

М004,Зимние кроссовки,40,6500

П001,Сандали,18,1000

П002,Кроссовки,30,2000

П003,Сапоги,0,2300

П004,Тапочки,15,800

4. Структура БД

CREATE TABLE "orders" (

"id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

"customer" TEXT,

"customer\_tel" TEXT,

"timestamp" INTEGER

)

CREATE TABLE "orders\_phones" (

"order\_id" INTEGER,

"phone\_id" INTEGER

)

CREATE TABLE "phones" (

"id" INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,

"name" TEXT,

"price" INTEGER

)