МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ

Московский технический университет связи и информатики

Курсовая работа

по дисциплине «Java-программирование»

на тему «Разработка приложения: Организация деталей складского помещения»

Выполнила: студентка группы БИБ2104 Ярцева Светлана Сергеевна

Проверила: доктор технических наук Смоленцева Татьяна Евгеньевна

Оценка _____

Дата _____

_____

Москва 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире информационных технологий, правильная организация и управление складским помещением — это ключевой фактор для всех успешных бизнесов. С каждым днем возрастает необходимость в удобных и интуитивно понятных инструментах для создания эффективной системы учета и управления складом. В данной курсовой работе будет рассмотрена разработка приложения на языке программирования Java, которое позволит компаниям более эффективно управлять ресурсами склада. В курсовой работе будет создано простое и понятное приложение, которое учитывает основные детали складского помещения, такие как инвентаризация, контроль над запасами, отслеживание товарных позиций и многое другое. Основным объективом является создание удобного и надежного приложения для оптимизации деятельности складского помещения, которое поможет предотвратить ошибки и сократить время на обработку информации. В курсовой работе будет описан процесс разработки приложения.

ОСНОВНАЯ ЧАСТЬ

1. Техническое задание на разработку

Требуется создать программу на языке программирования Java, реализующая управление базой данных деталей складского помещения. Программа должна выполнять следующие задачи:

- Предоставлять понятный пользователю интерфейс
- Создание базы данных, т.е. нового файла, где будет храниться вся информация
- Иметь возможность добавления новой записи в базу данных
- Иметь возможность удаления из базы данных записи
- Иметь возможность просматривать все данные в удобном формате

2. Описание программной реализации

Программа содержит всего 4 класса. 3 класса обеспечивают «внутреннюю» работу, а 4-ый, объединяя методы всех трех классов, создает удобное для пользователя приложение.

2.1. Первый класс «Hangar» создает структуру, содержит в себе 4 поля, обозначающие необходимые для описания детали ключи: Шифр, Наименование, Завод-производитель, Количество. В классе содержится удобный инициализатор, для того чтобы из текста, введенным пользователем, создался объект с правильными типами полей. Также в классе описаны методы получения всех полей (пример: public int getId()), это понадобится для других методов.

2.2. Второй класс «Filework» выполняет работу с файлом, содержит в себе поля типа File (файл) и List<Hangar> (список). Конструктор класса создает список и файл, и открывает его для чтения. В классе также есть метод public List<Hangar> getHangarList() для получения списка всех записей, public void addNewHangar(Hangar hangar) добавляет одну запись. Метод public void readAllInObject() читает из файла все записи базы данных в объект массива, а метод public void saveAllData() сохраняет в файл все записи базы данных из объекта массива. Метод public void removeRowByIndex(int index) будет

необходим для удаления определенной записи из базы данных, он получает индекс записи.

2.3. Третий класс «HangarComparator» нужен для переопределения метода public int compare(Hangar a, Hangar b) для того, чтобы изменить способ сравнения. В таблице базы данных сортировка осуществляется с помощью этого класса.

2.4. Четвертый класс «App» создает приложение. В нем множество полей для интерфейса такие как: кнопки, таблица, поля для ввода данных. Так же есть поле базы данных: Filework database. Точка входа public static void main(String[] args) создает новый объект класса App и применяет к нему метод createAndShowGUI. Метод public void createAndShowGUI() создает графический интерфейс и привязывает функции к кнопкам. Далее идут методы обработки нажатия кнопок:

public void addRecordButtonClick(ActionEvent e) – метод реализует нажатие кнопки «добавить запись», он выбирает соответствующее окно, которое будет видеть пользователь, позволяет ввести данные, но не сохранить.

public void showRecordsButtonClick(ActionEvent e) – метод реализует нажатие кнопки «показать записи», он выбирает соответствующее окно, которое будет видеть пользователь, строит таблицу данных и отображает ее.

public void sendButtonClick(ActionEvent e) – метод реализует нажатие кнопки «отправить», он сохраняет данные, введенные пользователем в соответствующие поля.

public void backButtonClick(ActionEvent e) – метод реализует нажатие кнопки «назад». Находясь в любом окне, кроме начального, можно вернутся на главную страницу. Метод делает невидимыми все окна, кроме начального.

public void exitButtonClick(ActionEvent e) – метод реализует нажатие кнопки «выход». Находясь в любом окне приложения, его можно закрыть. Метод закрывает программу.

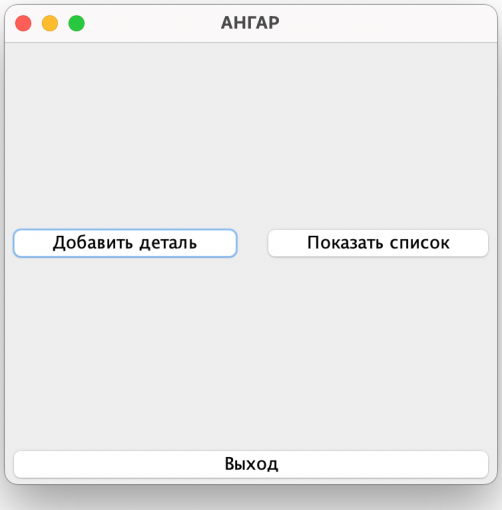
private void deleteButtonClick(ActionEvent e) – метод реализует нажатие кнопки «удалить». Находясь в окне со всеми записями и выделив

определенную запись, пользователь может ее удалить. Метод задействует описывающийся ранее метод removeRowByIndex и удаляет из базы данных запись с этим индексом.

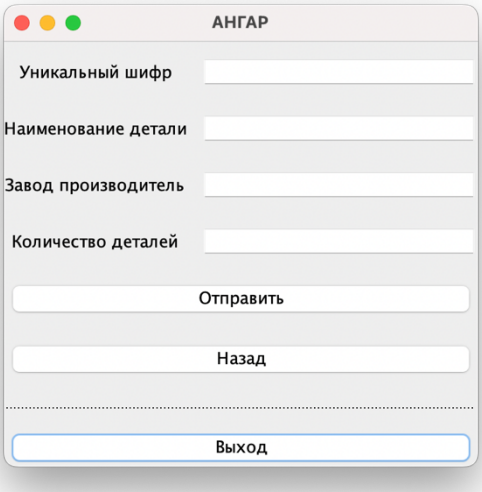
Также в главном классе «App» есть конструктор, создающий объект database, который, в свою очередь, является объектом класа Filework.

3. Описание результатов тестирования

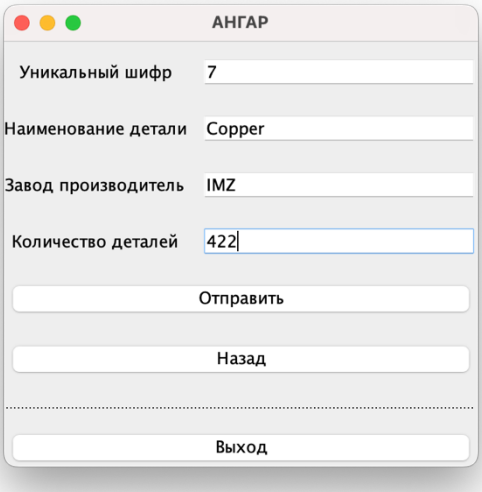
3.1. Запуск приложения. Главное окно



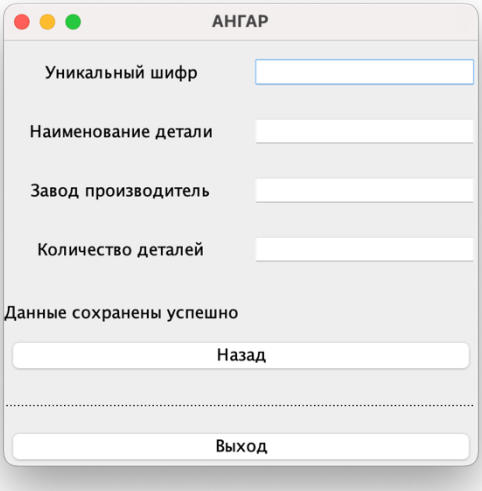
3.2. Выбор кнопки «Добавить деталь»



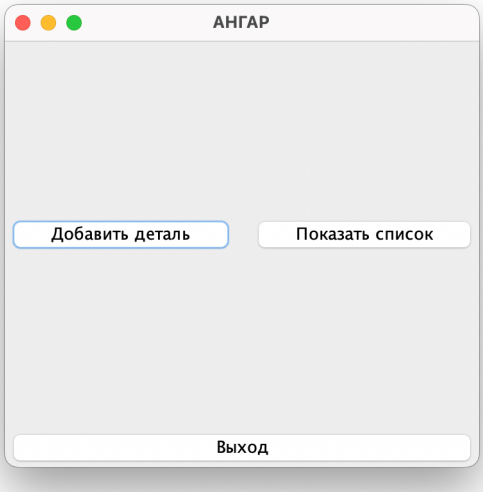
3.2.1 Заполнение полей



3.2.2. Нажатие кнопки «Сохранить»

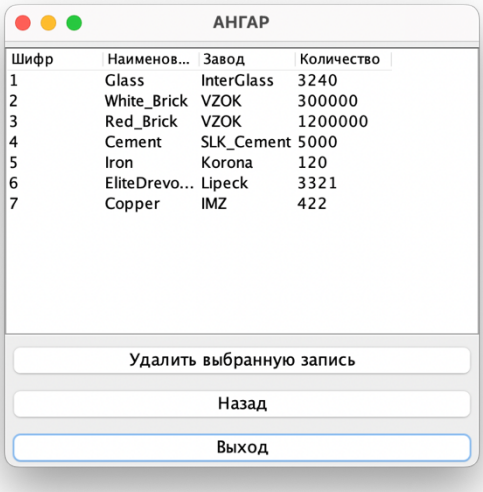


3.2.3. Нажатие кнопки «Назад», находясь на экране ввода данных



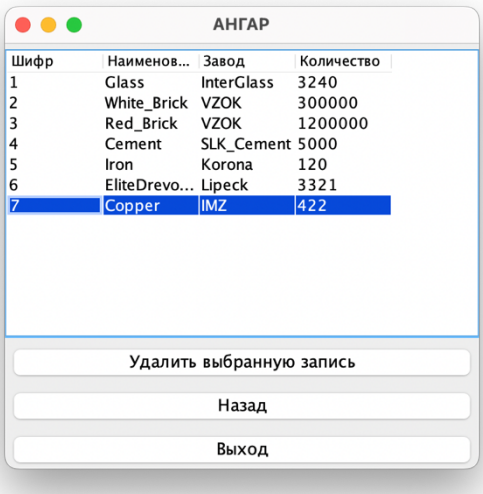
Вернулись на главный экран.

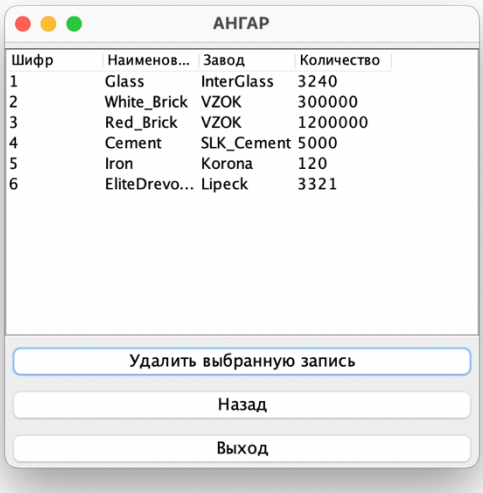
3.3. Выбор кнопки «Показать список»



Введенная запись сохранилась в базе данных.

3.3.1 Выбор записи и нажатие кнопки «Удалить выбранную запись»





Запись успешно удалена. Кнопка «Назад» в этом окне, так же возвращает на главный экран. Выход из приложения тоже реализуется.

4. Руководство пользователя

Шаг 1: Загрузка и установка

Скачайте файл «javacoursework.zip» из репозитория GitHub (yarssset) и установите Java Development Kit (JDK) на ваш компьютер, если его еще нет.

Шаг 2: Запуск приложения

Откройте терминал, перейдите в каталог, в котором находится скомпилированный файл приложения, путь к файлу должен выглядеть так:

(…/javacoursework/out/artifacts/javacoursework_jar). Затем введите команду "java -jar javacoursework.jar " и нажмите Enter. Приложение запустится.

Шаг 3: Добавление записи в базу данных

Для добавления записи в базу данных необходимо нажать кнопку "Добавить деталь" на главной странице приложения. Затем введите данные в соответствующие поля и нажмите кнопку "Сохранить". Запись будет добавлена в базу данных.

Шаг 4: Просмотр всей базы данных

Для просмотра всей базы данных необходимо нажать кнопку "Показать список" на главной странице приложения. Будут отображены все записи, которые были добавлены в базу данных.

Шаг 5: Удаление записи из базы данных

Для удаления записи из базы данных необходимо выбрать запись, которую нужно удалить, и нажать кнопку "Удалить выбранную запись". Запись будет удалена из базы данных.

Шаг 6: Закрытие приложения

Чтобы закрыть приложение, нажмите кнопку "Выход" на любой странице приложения.

Это простое и понятное приложение на языке Java, которое поможет организовать детали складского помещения. Это руководство поможет вам использовать его эффективно и без проблем.

ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы было разработано элементарное приложение на языке Java, предназначенное для организации деталей складского помещения. Главной целью работы было создание функционала, позволяющего управлять записями в базе данных склада. В результате разработки были реализованы основные функции приложения, включая добавление записей, их удаление и отображение всей базы данных. База данных была организована в файле, что обеспечивает удобство хранения и доступа к информации. Разработанное приложение предоставляет удобный интерфейс. Такое приложение может быть полезным инструментом для предприятий, занимающихся складским хозяйством. Однако, стоит отметить, что разработанное приложение является лишь примером и может быть доработано и расширено в дальнейшем. Возможные улучшения могут включать добавление дополнительных функций, таких как поиск и сортировка записей по определенному ключу, а также расширение возможностей хранения данных. В целом, данная курсовая работа позволила ознакомиться с процессом разработки приложения на языке Java. Полученный опыт и знания могут быть полезными при разработке подобных приложений в будущем.

ЛИТЕРАТУРА

1. Герберт Шилдт «Java. Полное руководство", 10-е изд: Пер. с англ. – СПб. : ООО «Альфакнига», 2018. — 1488с. : ил. — Парал, тит. англ.
2. Кей Хорстманн, Гари Корнелл «Java. Библиотека профессионала», том 1. Основы. 9-е изд. : Пер. с англ. — М. : ООО "И.Д.Вильямс", 2014. — 864 с. :ил. — Парал, тит. англ.
3. Sonoo Jaiswal «Cyber Security Tools» [Электронный ресурс]. URL: https://www.javatpoint.com/.

ПРИЛОЖЕНИЕ

Файл «Hangar.java»

```
package main;

import java.io.Serializable;

public class Hangar implements Serializable {
    private int id; // шифр
    private String name; // имя детали
    private String from; // название завода
    private int count; // количество на складе

    public Hangar(int id, String name, String from, int count){
        this.id = id;
        this.name = name;
        this.from = from;
        this.count = count;
    }
    public Hangar(String id, String name, String from, String count) {
        this.id = Integer.parseInt(id);
        this.name = name;
        this.from = from;
        this.count = Integer.parseInt(count);
    }
    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getFrom() {
        return from;
    }

    public int getCount() {
        return count;
    }

    public void setId(int id) {
        this.id = id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setFrom(String from) {
        this.from = from;
    }

    public void setCount(int count) {
        this.count = count;
    }

}
```

Файл «Filework.java»

```
package main;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class Filework {
    private File file;
    private List<Hangar> hangarList;

    public Filework(String filename) throws IOException{
        this.hangarList = new ArrayList<>();
        File file = new File(filename);
        if (!file.exists()) file.createNewFile();
        this.file = file;
        this.readAllInObject();
    }

    public List<Hangar> getHangarList() {
        return hangarList;
    }
    public void addNewHangar(Hangar hangar){
        this.hangarList.add(hangar);
    }

    public void readAllInObject() {
        try (FileInputStream fis = new FileInputStream(this.file);
             ObjectInputStream ois = new ObjectInputStream(fis)) {
            this.hangarList = (List<Hangar>) ois.readObject();
        } catch (ClassNotFoundException | EOFException e ) {
            System.err.println("Ошибка");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void saveAllData() {
        try (FileOutputStream fos = new FileOutputStream(this.file);
             ObjectOutputStream oos = new ObjectOutputStream(fos)) {
            oos.writeObject(this.hangarList);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void removeRowByIndex(int index) {hangarList.remove(index);}
}
```

Файл «HangarComparator.java»

```
package main;

public class HangarComparator implements java.util.Comparator<Hangar>{
    @Override
    public int compare(Hangar a, Hangar b) {
        return a.getId() - b.getId();
    }
}
```

Файл «App.java»

```
package main;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.io.IOException;
import java.util.List;

public class App extends JFrame {
    private static final String FILENAME = "file1.bin";
    private JPanel mainPanel;
    private JPanel top;
    private JPanel bottom;
    private JPanel content;
    private JPanel startpage;
    private JButton showRecordsButton;
    private JButton addRecordButton;
    private JPanel writepage;
    private JTextField textField1;
    private JTextField textField3;
    private JTextField textField4;
    private JTextField textField2;
    private JPanel listpage;
    private JTable table1;
```

```
            private JButton sendButton;
            private JButton backButtonInList;
            private JButton backButtonInWrite;
            private JButton exitButton;
            private JLabel text1;
            private JLabel title;
            private JButton deleteButton;
            Filework database;

            public void createAndShowGUI() {
                this.setTitle("АНГАР");
                this.setVisible(true);
                this.setResizable(true);
                this.setPreferredSize(new Dimension(370, 360));
                this.setLocationRelativeTo(null);
                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                addRecordButton.addActionListener(this::addRecordButtonClick);
                showRecordsButton.addActionListener(this::showRecordsButton-
Click);
                sendButton.addActionListener(this::sendButtonClick);
                backButtonInWrite.addActionListener(this::backButtonClick);
                backButtonInList.addActionListener(this::backButtonClick);
                exitButton.addActionListener(this::exitButtonClick);
                deleteButton.addActionListener(this::deleteButtonClick);
                this.pack();
            }
            public void addRecordButtonClick(ActionEvent e) {
                startpage.setVisible(false);
                writepage.setVisible(true);
                listpage.setVisible(false);
                text1.setVisible(false);
                sendButton.setVisible(true);
                textField1.setText("");
                textField3.setText("");
                textField4.setText("");
                textField2.setText("");
                this.pack();
            }
            public void showRecordsButtonClick(ActionEvent e) {
                startpage.setVisible(false);
                listpage.setVisible(true);
                this.pack();
```

```
            List<Hangar> sortedList = database.getHangarList();
            sortedList.sort(new HangarComparator());

            DefaultTableModel model = (DefaultTableModel) table1.getModel();
            model.setColumnIdentifiers(new String[]{"Шифр", "Наименование",
"Завод", "Количество"});
            model.setRowCount(0);

            for (Hangar record : sortedList) {
                model.addRow(new Object[] {record.getId(), record.getName(),
record.getFrom(), record.getCount()});
            }
            table1.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
        }
        public void sendButtonClick(ActionEvent e) {
            text1.setText("Данные сохранены успешно");
            if (textField1.getText().equals("") || text-
Field2.getText().equals("") || textField3.getText().equals("") || text-
Field4.getText().equals("")) {
                sendButton.setVisible(false);
                text1.setText("Ошибка: данные введены неправильно");
                text1.setVisible(true);
                textField1.setText("");
                textField3.setText("");
                textField4.setText("");
                textField2.setText("");
                return;
            }
            sendButton.setVisible(false);
            text1.setVisible(true);
            backButtonInWrite.setVisible(true);
            this.pack();

            Hangar record = new Hangar(
                    textField1.getText().trim(),
                    textField2.getText().trim(),
                    textField3.getText().trim(),
                    textField4.getText().trim());
            database.addNewHangar(record);
            database.saveAllData();
            textField1.setText("");
            textField3.setText("");
            textField4.setText("");
```

```
            textField2.setText("");
        }

        public void backButtonClick(ActionEvent e) {
            startpage.setVisible(true);
            writepage.setVisible(false);
            listpage.setVisible(false);
            database.saveAllData();
            this.pack();
        }
        public void exitButtonClick(ActionEvent e) {
            database.saveAllData();
            System.exit(0);
        }
        private void deleteButtonClick(ActionEvent e) {
            if (table1.getSelectedRow() != -1) {
                DefaultTableModel model = (DefaultTableModel) table1.get-
Model();
                int recordId = table1.getSelectedRow();
                model.removeRow(recordId);
                database.removeRowByIndex(recordId);
            }
        }
        public App() throws IOException {
            database = new Filework(FILENAME);
            this.getContentPane().add(mainPanel);
        }

        public static void main(String[] args) throws IOException {
            App app = new App();
            app.createAndShowGUI();
            app.pack();
        }
    }
```