Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра систем управления

Дисциплина: Объектное ориентированное программирование в системах управления

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

Объектное ориентированное программирование в системах управления

Вариант 24

Студент: гр. 222401 Саркисов А. В.

Руководитель: Снисаренко С. В.

Минск 2024

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет информационных технологий и управления

УТВЕРЖДАЮ

Заведующий кафедрой СУ

*–––––––––––––––––––––*

(подпись)

–––––––––––––––––––––2024 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту     Саркисову Алексею Владимировичу *–––––*

1. Тема проекта   Объектное ориентированное программирование в системах управления .

2. Срок сдачи студентом законченного проекта–––––19.04.2024 *––*

3. Исходные данные к проекту

Задание №1. Создать класс библиографических описаний книг CBookCard. Класс должен содержать поля согласно таблицам 1 – 3, в соответствии с номером варианта.

Задание №2. Необходимо реализовать иерархию классов предметной области на языке Java в соответствии с указанным вариантом задания.

Задание №3. Создать класс, соответствующий индивидуальному варианту задания. Создать коллекцию для хранения экземпляров созданного класса. Вид коллекции выбрать самостоятельно. Написать Windows-приложение для работы с этой коллекцией.

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

- титульный лист

- лист задания

- содержание

- введение

- задание (по вариантам)

- исходный код

- скриншоты работы приложения

- заключение

- список использованной литературы

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

Задание №1. Перечень скриншотов с работой диалогового окна.

Задание №2. Перечень скриншотов с работой диалогового окна.

Задание №3. Перечень скриншотов с работой windows-приложения.

6. Консультант по работе (с обозначением разделов работы)     Снисаренко Светлана Валерьевна*––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––*

7. Дата выдачи задания 26.02.2024

8. Календарный график работы над проектом на весь период проектирования   
(с обозначением сроков выполнения и трудоемкости отдельных этапов):

Задание №1. 10.02.2024 – 15.03.2024

Задание №2. 16.03.2024 – 17.04.2024

Задание №3. 18.04.2024 – 06.06.2024

РУКОВОДИТЕЛЬ*––––––––––––*

(подпись)

Задание принял к исполнению *–––––––\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_––*

(дата и подпись студента)

Содержание

[**ВВЕДЕНИЕ** 5](#_Toc164364124)

[**1** **КЛАССЫ И ОБЪЕКТЫ. ИНКАПСУЛЯЦИЯ** 7](#_Toc164364125)

[**1.1** **Условие первого задания** 7](#_Toc164364126)

[**1.2** **Листинг первого задания** 7](#_Toc164364127)

[**1.3** **Перечень скриншотов с работой программы:** 10](#_Toc164364128)

[**2** **НАСЛЕДОВАНИЕ. ПОЛИМОРФИЗМ** 11](#_Toc164364129)

[**2.1** **Условие второго задания** 11](#_Toc164364130)

[**2.2** **Листинг второго задания** 11](#_Toc164364131)

[**2.3** **Перечень скриншотов с работой программы:** 15](#_Toc164364132)

[**3** **КОЛЛЕКЦИИ. WINDOWS-ПРИЛОЖЕНИЯ** 16](#_Toc164364133)

[**3.1** **Условие третьего задания** 16](#_Toc164364134)

[**3.2** **Листинг третьего задания** 17](#_Toc164364135)

[**3.3** **Перечень скриншотов с работой windows приложения** 27](#_Toc164364136)

[**Заключение** 32](#_Toc164364137)

[**Список использованных источников** 33](#_Toc164364138)

# **ВВЕДЕНИЕ**

Объектно-ориентированное программирование (ООП) играет важную роль в различных сферах программирования, включая системы управления. Системы управления — программные комплексы, предназначенные для автоматизации и оптимизации управления различными объектами и процессами, такими как промышленные, транспортные, энергетические, информационные и т.д. Системы управления являются сложными и динамичными, требующими высокого уровня абстракции, модульности, надежности и расширяемости. Именно здесь ООП становится полезным, так как оно позволяет эффективно моделировать и реализовывать эти системы, используя принципы абстракции, инкапсуляции, наследования и полиморфизма. Курсовой проект был направлен на изучение и закрепление знаний по следующим темам: классы и объекты, инкапсуляция, полиморфизм, наследование, коллекции и создание Windows-приложений.

Всего в курсовой работе будет три раздела которые будут включать в себя:

Первый раздел включает в себя решение задания: Создать класс библиографических описаний книг CBookCard. Класс должен содержать поля согласно таблицам 1 – 3, в соответствии с номером варианта.

1. Все поля должны быть закрытыми. Для всех их должны быть созданы методы доступа (или свойства).
2. В конструкторе за исключением рейтинга и комментария должна осуществляться инициализация всех полей.
3. Для всех вариантов необходимо создать метод, выводящий описание книги в строку (переопределение метода ToString()), в которой будут присутствовать все значения полей класса, кроме поля – «комментария». В данном методе значения полей ISBN, УДК, ББК выводятся через пробел без использования знаков препинания. Значения атрибутов рейтинг и тираж должны выводиться в следующей форме: Тираж: <значение тиража>. Рейтинг: <значение рейтинга>.
4. Создать два объекта библиографических описаний книг. Вызвать для них метод, осуществляющий вывод строки библиографического описания. Можно использовать как реально существующие книги, так и чисто гипотетические.
5. Создать массив (размерностью не более 5) объектов библиографических описаний книг. Создать статический метод для упорядочивания (по году издания) данного массива. Вывести на экран значения массива до сортировки и после сортировки.

Второй раздел включает себя решение задания: Необходимо реализовать иерархию классов предметной области на языке Java в соответствии с указанным вариантом задания.

Третий раздел включает в себя решение задания: Создать класс, соответствующий индивидуальному варианту задания. Создать коллекцию для хранения экземпляров созданного класса. Вид коллекции выбрать самостоятельно. Написать Windows-приложение для работы с этой коллекцией, которое позволит выполнять:

1. добавление элемента в коллекцию с клавиатуры;
2. считывание данных из файла;
3. запись данных в тот же или указанный файл;
4. сортировку данных по различным критериям;
5. поиск элемента по заданному полю;
6. вывод всех элементов, удовлетворяющих заданному условию;
7. удаление элемента из коллекции.

Приложение должно содержать меню, диалоговые окна и предусматривать обработку ошибок.

# **КЛАССЫ И ОБЪЕКТЫ. ИНКАПСУЛЯЦИЯ**

## **Условие первого задания**

Создать класс библиографических описаний книг CBookCard. Класс должен содержать поля согласно таблицам 1 – 3, в соответствии с номером варианта.

1. Все поля должны быть закрытыми.
2. Для всех их должны быть созданы методы доступа (или свойства).
3. В конструкторе за исключением рейтинга и комментария должна осуществляться инициализация всех полей.
4. Для всех вариантов необходимо создать метод, выводящий описание книги в строку (переопределение метода ToString()), в которой будут присутствовать все значения полей класса, кроме поля – «комментария». В данном методе значения полей ISBN, УДК, ББК выводятся через пробел без использования знаков препинания. Значения атрибутов рейтинг и тираж должны выводиться в следующей форме: Тираж: <значение тиража>. Рейтинг: <значение рейтинга>.
5. Создать два объекта библиографических описаний книг. Вызвать для них метод, осуществляющий вывод строки библиографического описания. Можно использовать как реально существующие книги, так и чисто гипотетические.
6. Создать массив (размерностью не более 5) объектов библиографических описаний книг. Создать статический метод для упорядочивания (по году издания) данного массива. Вывести на экран значения массива до сортировки и после сортировки.



Рисунок 1.1 ­­–– Условие для варианта 21-24

## **Листинг первого задания**

**CBookCard.java**

Исходный код Main.java

**package** krsch1;

**import** java.util.Arrays;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

CBookCard book1 = **new** CBookCard("Александр Пушкин", "Евгений Онегин", "Издательство АСТ", 1833, 384, "821.3.38");

book1.setRating(3);

CBookCard book2 = **new** CBookCard("Фёдор Достоевский", "Преступление и наказание", "Издательство Эксмо", 1866, 480, "821.3.34");

book2.setRating(2);

System.***out***.println(book1);

System.***out***.println();

System.***out***.println(book2);

CBookCard[] books = **new** CBookCard[5];

books[1] = **new** CBookCard("Лев Толстой", "Война и мир", "Издательство Азбука-Аттикус", 1869, 1296, "821.161.1");

books[0] = **new** CBookCard("Михаил Булгаков", "Мастер и Маргарита", "Издательство Эксмо", 1966, 416, "616.12-084.5");

System.***out***.print("\n\n");

System.***out***.println("\nДо сортировки:");

**for** (**int** i = 0; i < books.length; i++) {

**if** (books[i] != **null**) {

System.***out***.println(books[i] + "\n");

}

}

*sortByYear*(books);

System.***out***.println("\nПосле сортировки:");

**for** (**int** i = 0; i < books.length; i++) {

**if** (books[i] != **null**) {

System.***out***.println(books[i] + "\n");

}

}

// чтобы консоль не закрывалась

(**new** Scanner(System.***in***)).nextLine();

}

**public** **static** **void** sortByYear(CBookCard[] books) {

Arrays.*sort*(books, (bookA, bookB) -> {

**if** (bookA == **null** || bookB == **null**) {

**return** 0;

}

**return** bookA.getYear() - bookB.getYear();

});

}

}

Исходный код CBookCard.java

**package** krsch1;

**public** **class** CBookCard {

**private** String author;

**private** String title;

**private** String publisher;

**private** **int** year;

**private** **int** pages;

**private** String udc;

**private** **int** rating;

**public** CBookCard(String author, String title, String publisher, **int** year, **int** pages, String udc) {

**this**.author = author;

**this**.title = title;

**this**.publisher = publisher;

**this**.year = year;

**this**.pages = pages;

**this**.udc = udc;

**this**.rating = 0;

}

**public** String getAuthor() {

**return** author;

}

**public** String getTitle() {

**return** title;

}

**public** String getPublisher() {

**return** publisher;

}

**public** **int** getYear() {

**return** year;

}

**public** **int** getPages() {

**return** pages;

}

**public** String getUdc() {

**return** udc;

}

**public** **int** getRating() {

**return** rating;

}

**public** **void** setRating(**int** rating) {

**if** (rating >= 0 && rating <= 3) {

**this**.rating = rating;

} **else** {

**throw** **new** IllegalArgumentException("Рейтинг должен быть в пределах от 0 до 3");

}

}

@Override

**public** String toString() {

**return** "Автор: " + author +

"\nЗаглавие: " + title +

"\nИздательство: " + publisher +

"\nГод издания: " + year +

"\nЧисло страниц: " + pages +

"\nУДК: " + udc.replace(".", " ") +

"\nРейтинг: " + rating;

}

}

## **Перечень скриншотов с работой программы:**

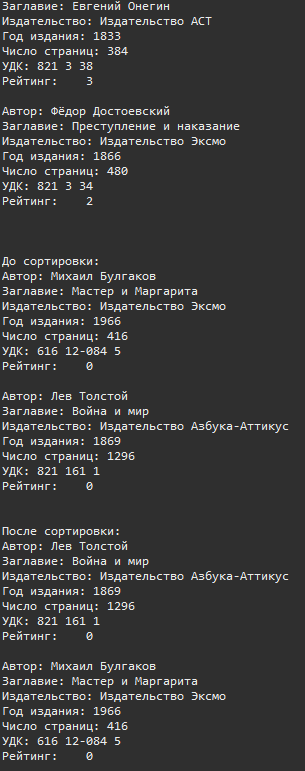


Рисунок 1.2 ­­–– Вывод в консоль

# **НАСЛЕДОВАНИЕ. ПОЛИМОРФИЗМ**

## **Условие второго задания**

Необходимо реализовать иерархию классов предметной области на языке Java в соответствии с указанным вариантом задания.

**Требования и рекомендации**

1. Иерархия классов должна состоять не менее чем из 3-х уровней.
2. Каждый класс должен содержать уникальный набор полей.
3. В классах не должно быть открытых полей (должны быть модификаторы доступа private или protected). Для доступа к полям класса необходимо использовать свойства.
4. В процессе реализации иерархии классов использовать абстрактные классы.
5. Классы рекомендуется оформить в виде библиотеки классов. Проверку функциональности осуществить в консольном проекте.
6. В рамках решения создать диаграммы классов.
7. Оформить отчет по заданию №2 курсового проекта.
8. ***Аппаратное обеспечение.*** Определить иерархию устройств, обеспечивающих функционирования ПЭВМ. Произвести сборку компьютеров различного типа (игровых, «для работы», серверов). Посчитать стоимость компьютеров.

## **Листинг второго задания**

**ElectricalAppliance.java**

Исходный код Main.java

**package** krsch2;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Device[] devices1 = {

**new** CPU("Intel Core i5", 200, 4, 3.1),

**new** RAM("Corsair Vengeance LPX 16GB", 80, 16),

**new** VideoCard("Gigabyte GeForce GTX 1650", 150, 6.13),

**new** Motherboard("ASUS ROG Strix B450-F Gaming", 150.0, "AMD B450")

};

Device[] devices2 = {

**new** CPU("AMD Ryzen 7 3700X", 300, 8, 3.6),

**new** RAM("G.Skill Ripjaws V 32GB", 160, 32),

**new** VideoCard("ASUS ROG Strix Radeon RX 5700 XT", 400, 20.58),

**new** Motherboard("Gigabyte Aorus X570 Pro Wi-Fi", 250.0, "AMD X570")

};

Device[] devices3 = {

**new** CPU("Intel Xeon E5-2690 v4", 1000, 14, 2.6),

**new** RAM("Kingston ValueRAM 64GB", 320, 64),

**new** VideoCard("NVIDIA Quadro RTX 6000", 4000, 130),

**new** Motherboard("ASUS TUF B460M-PRO", 150.0, "Intel B460")

};

Computer workComputer = **new** Computer("Work computer", devices1);

Computer gamingComputer = **new** Computer("Gaming computer", devices2);

Computer server = **new** Computer("Server", devices3);

System.***out***.println(workComputer + "\n");

System.***out***.println(gamingComputer + "\n");

System.***out***.println(server + "\n");

}

}

Исходный код Device.java

**package** krsch2;

**public** **abstract** **class** Device {

**private** String name;

**private** **double** price;

**public** Device(String name, **double** price) {

**this**.name = name;

**this**.price = price;

}

**public** String getName() {

**return** name;

}

**public** **double** getPrice() {

**return** price;

}

}

Исходный код CPU.java

**package** krsch2;

**public** **class** CPU **extends** Device {

**private** **int** cores;

**private** **double** frequency;

**public** CPU(String name, **double** price, **int** cores, **double** frequency) {

**super**(name, price);

**this**.cores = cores;

**this**.frequency = frequency;

}

**public** **int** getCores() {

**return** cores;

}

**public** **double** getFrequency() {

**return** frequency;

}

@Override

**public** String toString() {

**return** "CPU: " + **this**.getName() + " Price: $" +

**this**.getPrice() + " Cores: " +

**this**.cores + " Frequency: " + **this**.frequency;

}

}

Исходный код RAM.java:

**package** krsch2;

**public** **class** RAM **extends** Device {

**private** **int** capacityGB;

**public** RAM(String name, **double** price, **int** capacityGB) {

**super**(name, price);

**this**.capacityGB = capacityGB;

}

**public** **int** getCapacityGB() {

**return** capacityGB;

}

@Override

**public** String toString() {

**return** "RAM: " + **this**.getName() + " Price: $" +

**this**.getPrice() + " Capacity: " +

**this**.capacityGB + " GB";

}

}

Исходный код Motherboard.java

**package** krsch2;

**public** **class** Motherboard **extends** Device {

**private** String chipset;

**public** Motherboard(String name, **double** price, String chipset) {

**super**(name, price);

**this**.chipset = chipset;

}

**public** String getChipset() {

**return** chipset;

}

@Override

**public** String toString() {

**return** "Motherboard: " + **this**.getName() + " Price: $" +

**this**.getPrice() + " Chipset: " +

**this**.chipset;

}

}

Исходный код Videocard.java

**package** krsch2;

**public** **class** VideoCard **extends** Device {

**private** **double** teraflops;

**public** VideoCard(String name, **double** price, **double** teraflops) {

**super**(name, price);

**this**.teraflops = teraflops;

}

**public** **double** getTeraflops() {

**return** teraflops;

}

**public** **void** setTeraflops(**int** teraflops) {

**this**.teraflops = teraflops;

}

@Override

**public** String toString() {

**return** "Videocard: " + **this**.getName() + " Price: $" +

**this**.getPrice() + " Teraflops: " +

**this**.teraflops;

}

}

Исходный код Computer.java

**package** krsch2;

**public** **class** Computer {

**protected** String name;

**protected** Device[] devices;

**public** Computer(String name, Device[] devices) {

**this**.name = name;

**this**.devices = devices;

}

**public** **double** getCost() {

**double** cost = 0;

**for** (Device device : devices) {

cost += device.getPrice();

}

**return** cost;

}

@Override

**public** String toString() {

String str = **new** String();

str = "Computer: " + **this**.name;

**for** (Device device : devices) {

str += "\n" + device.toString();

}

str += "\nTotal price: $" + getCost();

**return** str;

}

}

## **Перечень скриншотов с работой программы:**

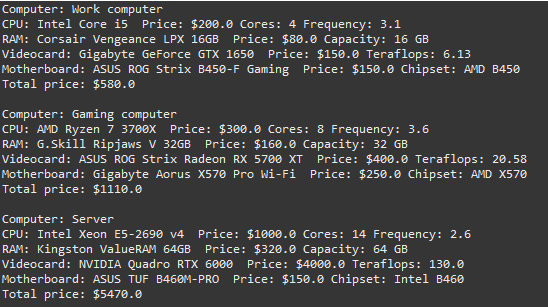


Рисунок 2.1 ­­–– Вывод в консоль

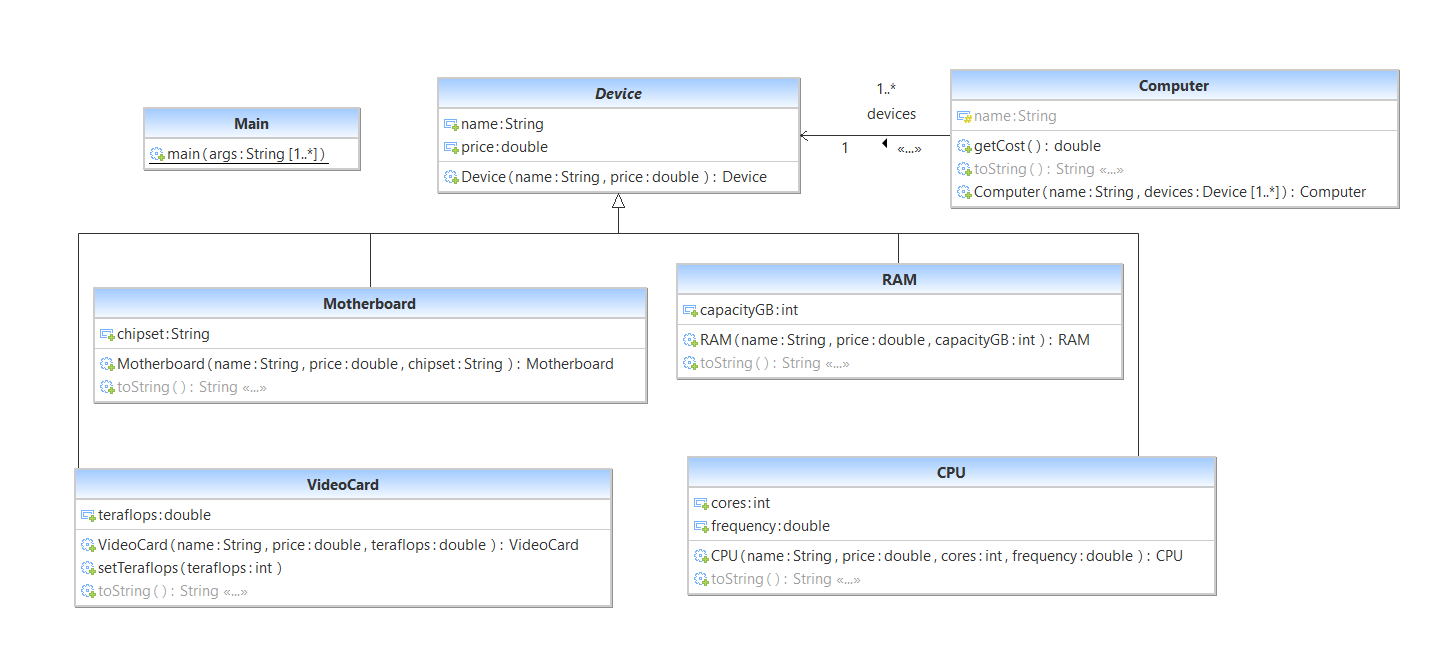


Рисунок 2.2­­–– Диаграмма классов

# **КОЛЛЕКЦИИ. WINDOWS-ПРИЛОЖЕНИЯ**

## **Условие третьего задания**

Создать класс, соответствующий индивидуальному варианту задания. Создать коллекцию для хранения экземпляров созданного класса. Вид коллекции выбрать самостоятельно. Написать Windows-приложение для работы с этой коллекцией, которое позволит выполнять:

1) добавление элемента в коллекцию с клавиатуры;

2) считывание данных из файла;

3) запись данных в тот же или указанный файл;

4) сортировку данных по различным критериям;

5) поиск элемента по заданному полю;

6) вывод всех элементов, удовлетворяющих заданному условию;

7) удаление элемента из коллекции.

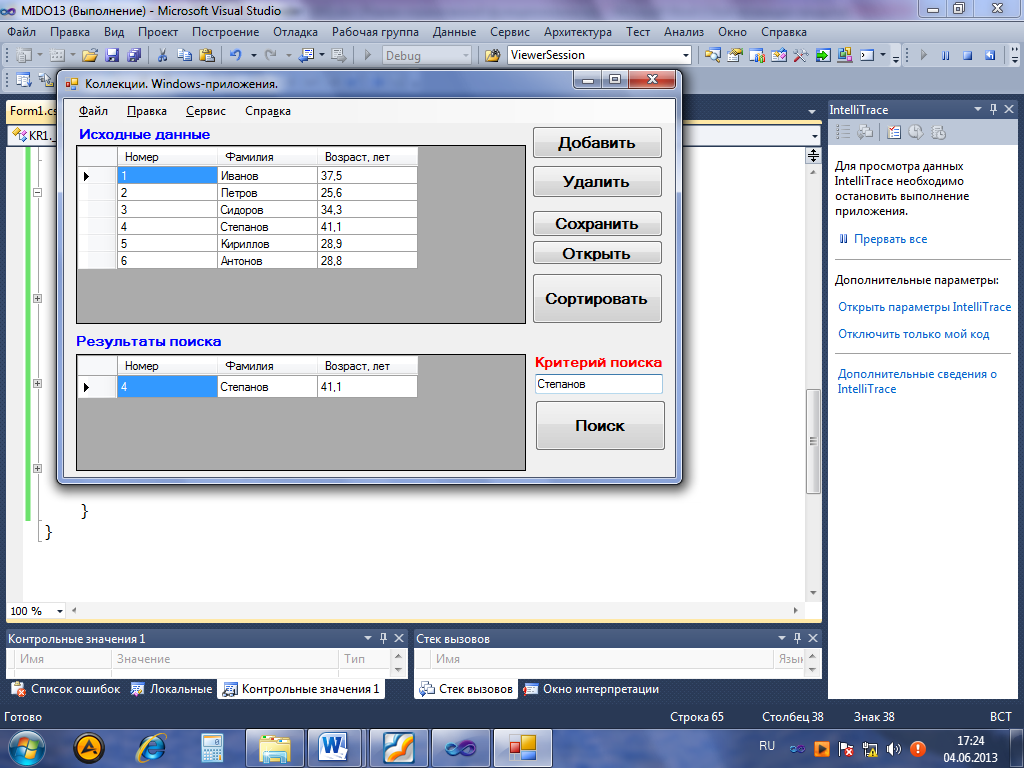


Рисунок 3.1 – Пример графического интерфейса приложения

Вариант 24:

Описать класс с именем STUDENT, содержащий следующие поля:

* фамилия и инициалы;
* номер группы;
* успеваемость (массив из пяти элементов).

Написать программу, выполняющую следующие действия:

* см. общее задание;
* ввод с клавиатуры данных (записи должны быть упорядочены по алфавиту);
* вывод на экран фамилий и номеров групп для всех студентов, имеющих все положительные оценки (если таких студентов нет, вывести соответствующее сообщение).

## **Листинг третьего задания**

Исходный код App.cs

using System;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Windows.Forms;

namespace task3.Forms

{

public partial class App : Form

{

// SortableBindingList предоставляет возможность сортировки элементов

private SortableBindingList<Student> \_students;

public SortableBindingList<Student> Students //привязка приватного поля к публичному

{

get => \_students;

set

{

\_students = value;

OnPropertyChanged();

}

}

public App()

{

InitializeComponent();

Students = new SortableBindingList<Student>();

// Настройка столбцов DataGridView1

dataGridView1.AutoGenerateColumns = false;

dataGridView2.AutoGenerateColumns = false;

DataGridViewTextBoxColumn fullNameColumn1 = new DataGridViewTextBoxColumn

{

Name = "FullName",

HeaderText = "Фамилия и инициалы",

DataPropertyName = "FullName"

};

dataGridView1.Columns.Add(fullNameColumn1);

DataGridViewTextBoxColumn groupNumberColumn1 = new DataGridViewTextBoxColumn

{

Name = "GroupNumber",

HeaderText = "Номер группы",

DataPropertyName = "GroupNumber"

};

dataGridView1.Columns.Add(groupNumberColumn1);

string[] gradeNames = { "Математика", "Физика", "Русский язык", "Белорусский язык", "Информатика" };

string[] grades = { "MathGrade", "PhysicsGrade", "RussianGrade", "BelorussianGrade", "InformaticsGrade" };

for (int i = 0; i < 5; i++)

{

DataGridViewTextBoxColumn gradeColumn = new DataGridViewTextBoxColumn

{

Name = $"Grade{i}",

HeaderText = gradeNames[i],

DataPropertyName = grades[i]

};

dataGridView1.Columns.Add(gradeColumn);

}

dataGridView1.DataSource = Students;

}

private void SaveButton\_Click(object sender, EventArgs e)

{

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*"; // Фильтр файлов для сохранения

saveFileDialog.FilterIndex = 1; // Устанавливаем индекс выбранного фильтра

saveFileDialog.RestoreDirectory = true; // Восстанавливаем путь к последней директории

saveFileDialog.FileName = "save.txt"; // Устанавливаем имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK) // Если пользователь выбрал файл для сохранения

{

using (StreamWriter sw = new StreamWriter(saveFileDialog.FileName))

{

foreach (var student in Students)

{

sw.WriteLine(student.toString());

}

}

}

}

}

private void AddButton\_Click(object sender, EventArgs e)

{

InputForm inputForm = new InputForm();

inputForm.ShowDialog();

if (inputForm.GetOrder() != null)

{

var student = inputForm.GetOrder();

Students.Add(student);

dataGridView1.Refresh();

}

dataGridView1.Sort(dataGridView1.Columns[0], ListSortDirection.Ascending);

}

private void DelButton\_Click(object sender, EventArgs e)

{

// удаляем выбранные элементы

foreach (DataGridViewRow row in dataGridView1.SelectedRows)

{

dataGridView1.Rows.Remove(row);

}

}

private void LoadButton\_Click(object sender, EventArgs e)

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*"; // Фильтр файлов для открытия

openFileDialog.FilterIndex = 1; // Устанавливаем индекс выбранного фильтра

openFileDialog.RestoreDirectory = true; // Восстанавливаем путь к последней директории

if (openFileDialog.ShowDialog() == DialogResult.OK) // Если пользователь выбрал файл для открытия

{

Students.Clear(); // Очистка коллекции перед загрузкой данных из файла

using (StreamReader sr = new StreamReader(openFileDialog.FileName))

{

string line;

while ((line = sr.ReadLine()) != null) //

{

// разделяет прочитанную строку на массив строк с помощью символа-разделителя `

string[] splited = line.Split('`');

// добавляем сами строки

Students.Add(new Student(splited[0], int.Parse(splited[1]), int.Parse(splited[2]), int.Parse(splited[3]),

int.Parse(splited[4]), int.Parse(splited[5]), int.Parse(splited[6])));

}

sr.Close();

}

}

}

dataGridView1.Sort(dataGridView1.Columns[0], ListSortDirection.Ascending);

}

// обрабатывает кнопку "Полученная сумма" в которой мы вызываем диалоговое окно

// с выбором нужного нам получателя затем мы узнаём сумму деняк у него на счету

private void pioneerButton\_Click(object sender, EventArgs e)

{

SortableBindingList<Student> pioneers = new SortableBindingList<Student>();

dataGridView2.Columns.Clear();

DataGridViewTextBoxColumn fullNameColumn2 = new DataGridViewTextBoxColumn

{

Name = "FullName",

HeaderText = "Фамилия и инициалы",

DataPropertyName = "FullName"

};

dataGridView2.Columns.Add(fullNameColumn2);

DataGridViewTextBoxColumn groupNumberColumn2 = new DataGridViewTextBoxColumn

{

Name = "GroupNumber",

HeaderText = "Номер группы",

DataPropertyName = "GroupNumber"

};

dataGridView2.Columns.Add(groupNumberColumn2);

dataGridView2.DataSource = pioneers;

pioneers.Clear();

foreach (var row in Students)

{

if (row.RussianGrade > 3 && row.InformaticsGrade > 3 && row.BelorussianGrade > 3 &&

row.PhysicsGrade > 3 && row.MathGrade > 3)

{

pioneers.Add(row);

}

}

dataGridView2.Refresh();

}

private void searchButton\_Click(object sender, EventArgs e)

{

dataGridView2.Columns.Clear();

SortableBindingList<Student> finded = new SortableBindingList<Student>();

DataGridViewTextBoxColumn fullNameColumn2 = new DataGridViewTextBoxColumn

{

Name = "FullName",

HeaderText = "Фамилия и инициалы",

DataPropertyName = "FullName"

};

dataGridView2.Columns.Add(fullNameColumn2);

DataGridViewTextBoxColumn groupNumberColumn2 = new DataGridViewTextBoxColumn

{

Name = "GroupNumber",

HeaderText = "Номер группы",

DataPropertyName = "GroupNumber"

};

dataGridView2.Columns.Add(groupNumberColumn2);

string[] gradeNames = { "Математика", "Физика", "Русский язык", "Белорусский язык", "Информатика" };

string[] grades = { "MathGrade", "PhysicsGrade", "RussianGrade", "BelorussianGrade", "InformaticsGrade" };

for (int i = 0; i < 5; i++)

{

DataGridViewTextBoxColumn gradeColumn = new DataGridViewTextBoxColumn

{

Name = $"Grade{i}",

HeaderText = gradeNames[i],

DataPropertyName = grades[i]

};

dataGridView2.Columns.Add(gradeColumn);

}

dataGridView2.DataSource = finded;

foreach (var row in Students)

{

if (row.FullName.Contains(searchText.Text) || row.GroupNumber.ToString().Contains(searchText.Text))

{

finded.Add(row);

}

}

dataGridView2.Refresh();

}

// Информирование об изменении свойства. Этот шаблон используется для

// уведомления подписчиков об изменении значения свойства объекта.

// Объявление события PropertyChanged, которое используется для

// уведомления подписчиков об изменении любого свойства объекта

public event PropertyChangedEventHandler PropertyChanged;

// Защищенный виртуальный метод OnPropertyChanged, который вызывается из сеттеров свойств и вызывает событие PropertyChanged

// Параметр propertyName содержит имя измененного свойства и автоматически получается с помощью атрибута [CallerMemberName]

protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

// Вызов события PropertyChanged с помощью оператора ?, который проверяет, не является ли событие null, прежде чем вызывать его

// Если событие не равно null, оно вызывается с помощью метода Invoke, который принимает в качестве

// параметров объект, вызвавший событие (this) и объект PropertyChangedEventArgs, содержащий имя измененного свойства

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}}

Исходный код Student.cs

using System.ComponentModel;

using System.Runtime.CompilerServices;

namespace task3

{

public class Student

{

private string \_fullName;

private int \_groupNumber;

private int[] \_grades;

[DisplayName("Фамилия и инициалы")]

public string FullName

{

get => \_fullName;

set

{

\_fullName = value;

OnPropertyChanged();

}

}

[DisplayName("Номер группы")]

public int GroupNumber

{

get => \_groupNumber;

set

{

\_groupNumber = value;

OnPropertyChanged();

}

}

public int[] Grades

{

get => \_grades;

set

{

\_grades = value;

OnPropertyChanged(nameof(MathGrade));

OnPropertyChanged(nameof(PhysicsGrade));

OnPropertyChanged(nameof(RussianGrade));

OnPropertyChanged(nameof(BelorussianGrade));

OnPropertyChanged(nameof(InformaticsGrade));

}

}

public int MathGrade

{

get => \_grades[0];

set

{

\_grades[0] = value;

OnPropertyChanged();

}

}

public int PhysicsGrade

{

get => \_grades[1];

set

{

\_grades[1] = value;

OnPropertyChanged();

}

}

public int RussianGrade

{

get => \_grades[2];

set

{

\_grades[2] = value;

OnPropertyChanged();

}

}

public int BelorussianGrade

{

get => \_grades[3];

set

{

\_grades[3] = value;

OnPropertyChanged();

}

}

public int InformaticsGrade

{

get => \_grades[4];

set

{

\_grades[4] = value;

OnPropertyChanged();

}

}

public Student(string fullName, int groupNumber, int[] grades)

{

this.FullName = fullName;

this.GroupNumber = groupNumber;

this.Grades = grades;

}

public Student(string fullName, int groupNumber, int mathGrade, int physicsGrade,

int russianGrade, int belorussianGrade, int informaticsGrade)

{

this.FullName = fullName;

this.GroupNumber = groupNumber;

int[] grades = new int[] { mathGrade, physicsGrade, russianGrade, belorussianGrade, informaticsGrade };

this.Grades = grades;

}

public string GetFullName()

{

return this.FullName;

}

public int GetGroupNumber()

{

return this.GroupNumber;

}

public int[] GetGrades()

{

return this.Grades;

}

public string toString()

{

return this.FullName + "`" + this.GroupNumber + "`" + string.Join("`", this.Grades);

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)

=> PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

Исходный код SortableBindingList.cs

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Windows.Forms;

namespace task3

{

public class SortableBindingList<T> : BindingList<T>

{

private bool isSortedValue;

ListSortDirection sortDirectionValue;

PropertyDescriptor sortPropertyValue;

public SortableBindingList() //конструктор без параметров

{

}

public SortableBindingList(IEnumerable<T> list) //конструктор для конвертации любой коллекции в эту

{

foreach (T o in list)

{

this.Add(o);

}

}

protected override void ApplySortCore(PropertyDescriptor prop, ListSortDirection direction) //применение сортировочного ядра к коллекции

{

sortPropertyValue = prop;

sortDirectionValue = direction;

IEnumerable<T> query = base.Items;

query = direction == ListSortDirection.Ascending

? query.OrderBy(i => prop.GetValue(i))

: query.OrderByDescending(i => prop.GetValue(i)); // определение направления сортировки

int newIndex = 0;

foreach (T item in query)

{

this.Items[newIndex] = item;

newIndex++;

}

isSortedValue = true;

this.OnListChanged(new ListChangedEventArgs(ListChangedType.Reset, -1)); //отправка ивента об изменении значений

}

protected override PropertyDescriptor SortPropertyCore => sortPropertyValue;

protected override ListSortDirection SortDirectionCore => sortDirectionValue;

protected override bool SupportsSortingCore => true;

protected override bool IsSortedCore => isSortedValue;

}

}

Исходный код Program.cs

using System;

using System.Windows.Forms;

using task3.Forms;

namespace task3

{

internal static class Program

{

// Главная точка входа для приложения.

[STAThread]

static void Main()

{

// включает визуальные стили Windows Forms

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

// запускает главный цикл приложения

Application.Run(new App());

}

}

}

Исходный код InputForm.cs

using System;

using System.Windows.Forms;

namespace task3.Forms

{

public partial class InputForm : Form

{

private int \_groupNum;

private int \_mathGrade;

private int \_physicsGrade;

private int \_russianGrade;

private int \_informaticsGrade;

private int \_belorussianGrade;

public InputForm()

{

InitializeComponent();

}

private void addToListButton\_Click(object sender, EventArgs e)

{

// @todo придумать более красивый вариант для этой функции

if (!ValidateGrade(mathTextBox, out \_mathGrade))

{

MessageBox.Show("Значение оценки математики является некорректным!");

}else if (!ValidateGrade(physicsTextBox, out \_physicsGrade))

{

MessageBox.Show("Значение оценки физики является некорректным!");

}else if (!ValidateGrade(russianTextBox, out \_russianGrade))

{

MessageBox.Show("Значение оценки русского языка является некорректным!");

}

else if (!ValidateGrade(informaticsTextBox, out \_informaticsGrade))

{

MessageBox.Show("Значение оценки информатики является некорректным!");

}

else if (!ValidateGrade(belorussianTextBox, out \_belorussianGrade))

{

MessageBox.Show("Значение оценки белорусского языка является некорректным!");

}else if (!int.TryParse(groupnumTextBox.Text, out \_groupNum))

{

MessageBox.Show("Значение номера группы является некорректным!");

}

else if (fieldsIsEmpty())

{

MessageBox.Show("Поля не должны быть пустыми");

}

else

{

this.Close();

}

}

private bool fieldsIsEmpty()

{

if (fullnameTextBox.Text == "" || groupnumTextBox.Text == "" || mathTextBox.Text == "" || physicsTextBox.Text == ""

|| russianTextBox.Text == "" || informaticsTextBox.Text == "" || belorussianTextBox.Text == "")

{

return true;

}

return false;

}

private bool ValidateGrade(TextBox textBox, out int grade)

{

if (!int.TryParse(textBox.Text, out grade) || grade < 0 || grade > 10)

{

return false;

}

return true;

}

// internal означает что этот метод будет доступен только из текущей сборки

public Student GetOrder()

{

int[] grades = new int[] { \_mathGrade, \_physicsGrade, \_russianGrade, \_belorussianGrade, \_informaticsGrade };

try

{

return new Student(this.fullnameTextBox.Text, \_groupNum, grades);

}

catch (System.FormatException e)

{

return null;

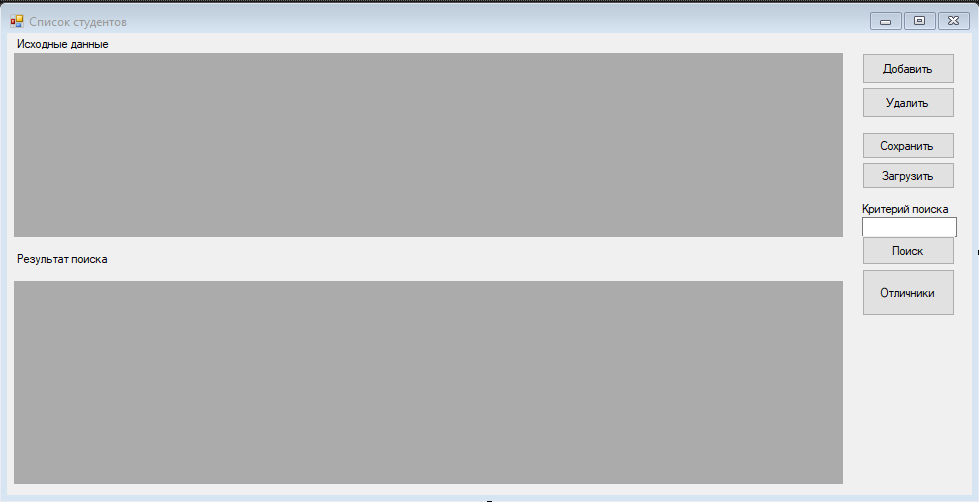
}

}

}

}

## **3.3 Перечень скриншотов с работой windows приложения**



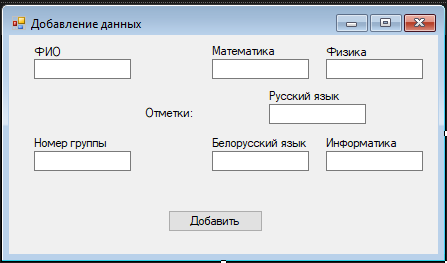


Рисунок 3.1 –Графический интерфейс приложения

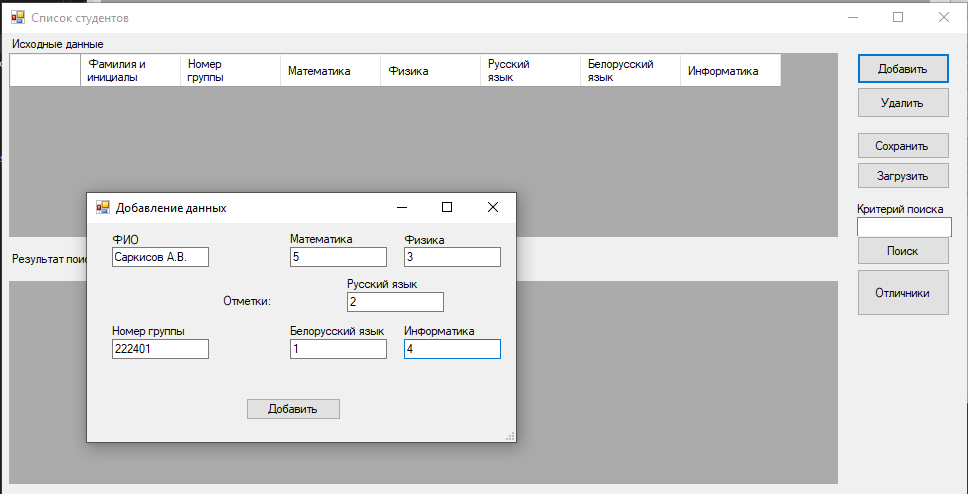


Рисунок 3.2 – Работа кнопки “Добавить”

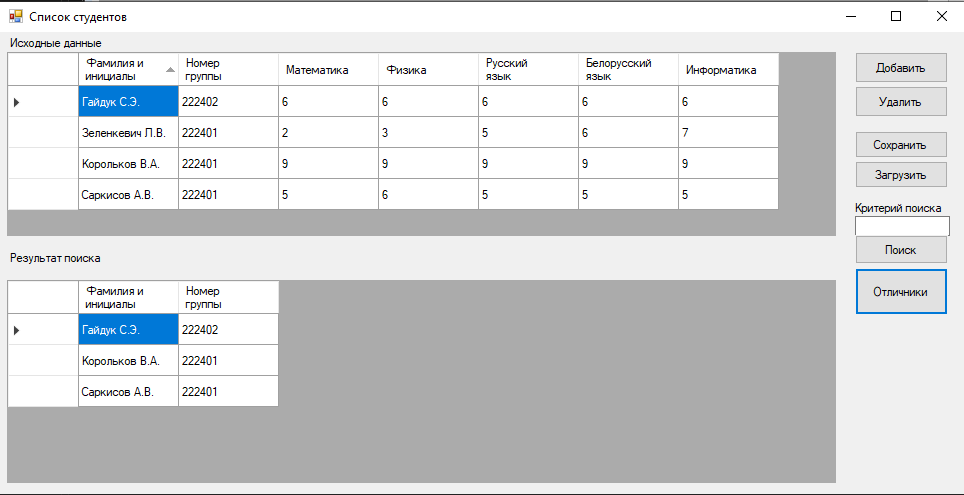


Рисунок 3.3 – Работа кнопки “Отличники”

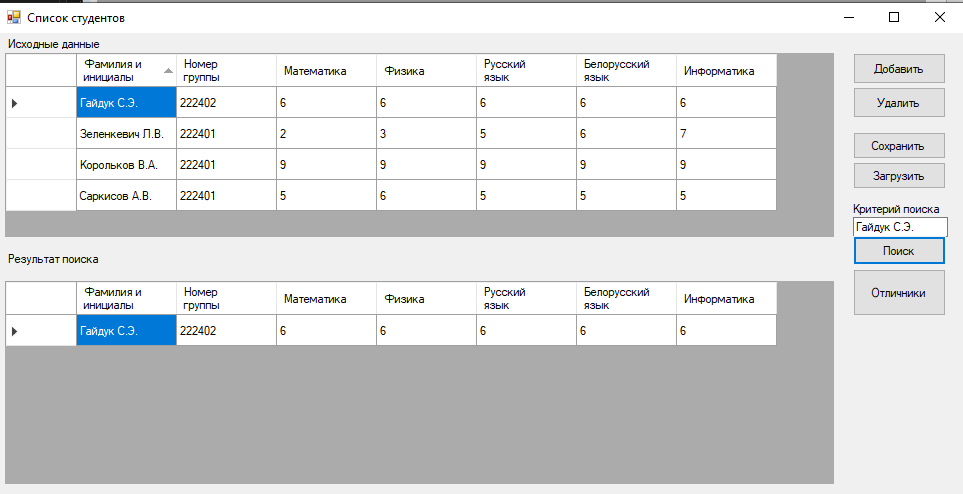


Рисунок 3.4 – Работа кнопки “Поиск”

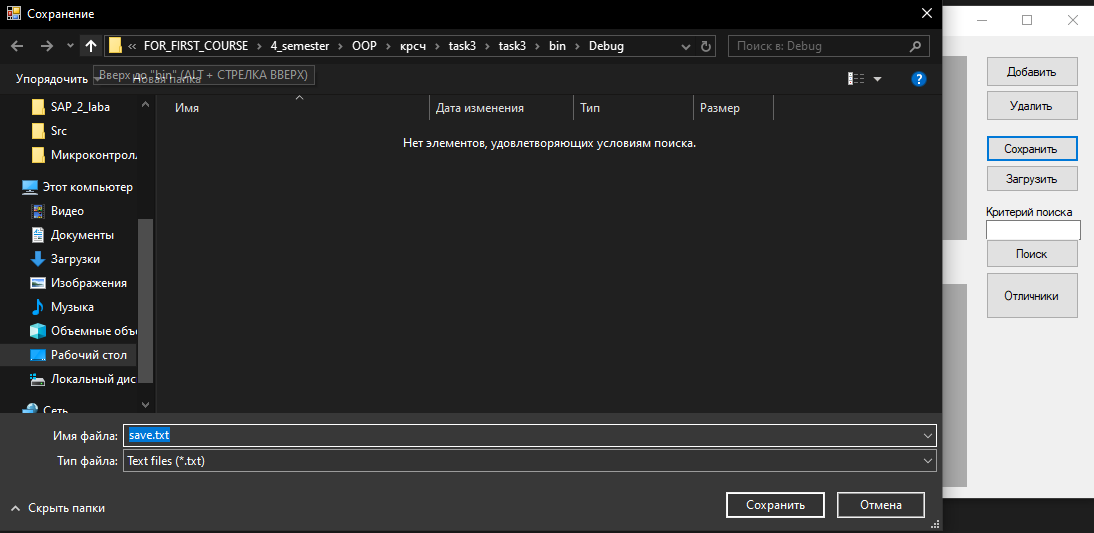


Рисунок 3.5 –Работа кнопки “Сохранить”

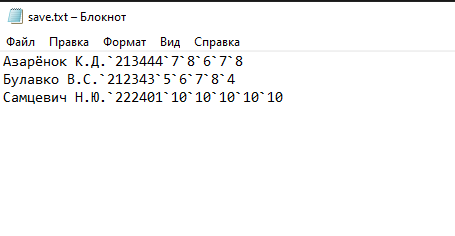


Рисунок 3.5.1 – Результат кнопки “Cохранить”

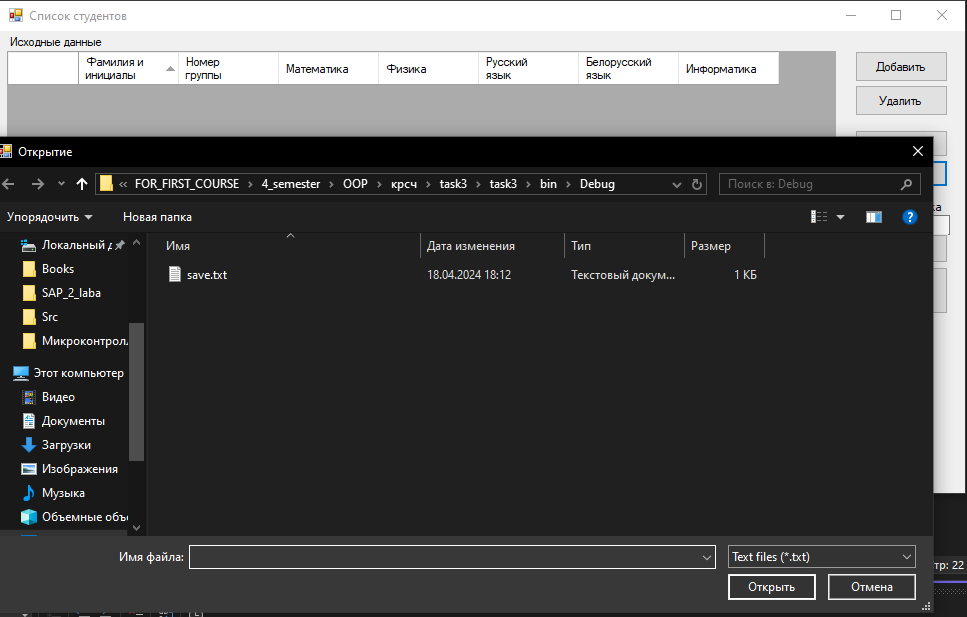


Рисунок 3.6 – Работа кнопки “Загрузить”

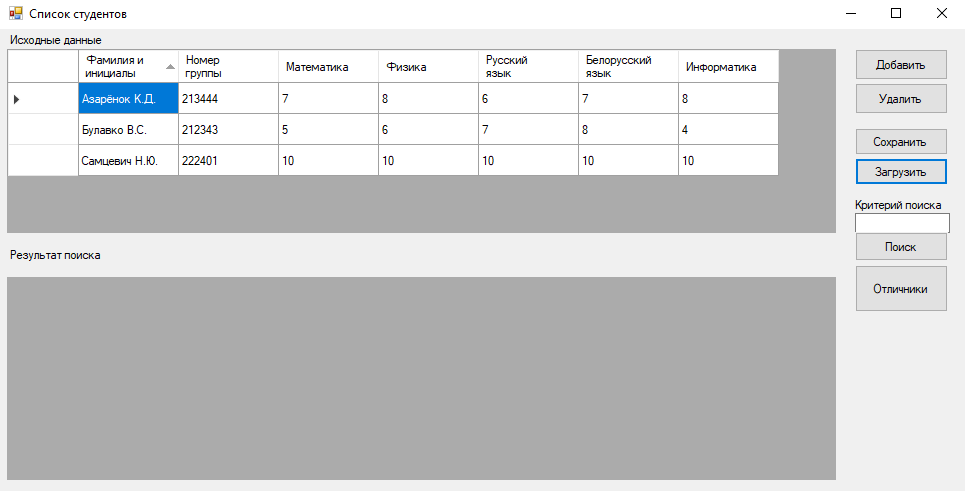


Рисунок 3.6.1 – Результат кнопки “Загрузить”

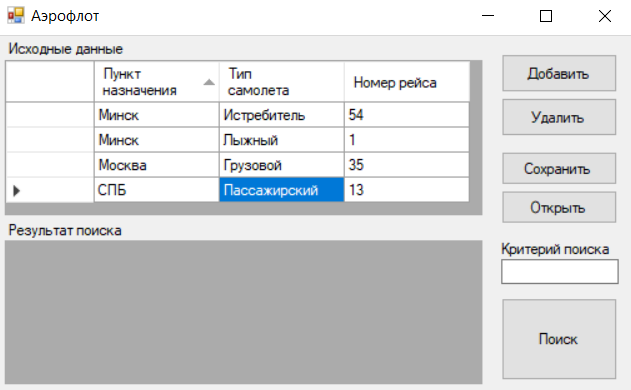


Рисунок 3.6.1 – Работа сортировки по пункту назначения

# **Заключение**

В процессе выполнения курсовой работы были изучены основы объектно-ориентированного программирования и применены на практике с использованием языка программирования Java. Было разработано приложение, которое включает в себя три раздела.

В первом разделе был создан класс CBookCard, содержащий поля, методы доступа и конструктор для инициализации всех свойств. Кроме того, был создан метод, который выводит описание книги в виде строки.

Во втором разделе была реализована иерархия классов для предметной области с использованием языка программирования Java, в соответствии с заданным вариантом.

В третьем разделе был создан класс, который соответствует индивидуальному заданию, и выбрана коллекция для хранения экземпляров этого класса. Также было разработано приложение для Windows, которое работает с этой коллекцией, позволяющее добавлять элементы, считывать и записывать данные в файл, а также сортировать данные по заданным критериям.

В итоге выполнения курсовой работы были закреплены знания по основам объектно-ориентированного программирования и применены на практике на языке программирования Java. Кроме того, был получен опыт разработки приложений для Windows и работы с коллекциями данных.

# **Список использованных источников**

1. Столяров А. В. Программирование: введение в профессию. В трёх томах — Макс Пресс 2021. — 2121 с.
2. Макконнелл С. Совершенный код. Мастер-класс. — Русская редакция 2019. —896 с.
3. Блох, Д. Java. Эффективное программирование. - М. : Лори , 2002. —224 с.
4. https://learn.microsoft.com/ru-ru/dotnet/csharp/