МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ОТЧЕТ

по лабораторной работе №9
по дисциплине «Объектно-ориентированное программирование»
Тема: «Использование наследования с применением интерфейсов»
Вариант №7

Студент гр. 2301 _____ Комиссаров П.Е.

Преподаватель: _____ Васильев А.А.

Санкт-Петербург
2023

Цель работы

Целью курсовой работы является закрепление теоретических знаний и практических навыков разработки программного обеспечения на основе объектно-ориентированного подхода.

Изучение механизмов наследования с помощью интерфейсов и абстрактных классов.

1. Формулировка задания

Разработать программу для обеспечения работы деканата. Вариант №4 первого раздела курсовой работы.

2. Теоретический аспект задачи

Описание задач:

1) Написать несколько классов и интерфейсов по заданной предметной области.

2) На основе созданных классов, написать программу, которая обеспечивает работу деканата.

В данной работе, деканат отслеживает успеваемость студентов. Первым делом деканат составляет список студентов и групп, после чего работает с ними.

Разработка программного продукта, который будет включать в себя функционал для обеспечения работы деканата:

Вывод списка студентов группы на экран, где у каждого студента есть такие параметры, как: фио, возраст, успеваемость, кол-во объяснительных, группа, студенческий билет. Если студент является старостой, то программа выводит соответствующее сообщение.

Исключение студентов по списку, либо исключать список неуспевающих студентов

Отправка студента писать объяснительную за пропуск пары

Вывод на экран кол-ва студентов в группе, среднюю успеваемость группы, старосту группы (если старосты нет, то назначается новый староста).

3. Формализация задачи

1. В программе нужно реализовать чтение из файлов (чтение списка групп, чтение списка студентов каждой группы). Для этого создается соответствующего 2 класса: databaseForGroups, databaseForStudents.

2. В программе необходимо реализовать работу со студентами в группе. Для этого создается класс: student.

3. В программе необходимо реализовать работу с группами. Это нужно, чтобы впоследствии работать со студентами отдельных групп. Для этого создается класс: group.

4. В программе необходимо реализовать работу деканата. В классе должны быть реализованы основные функции деканата. Для этого создается класс: dekanatFunctions.

5. Также необходимо реализовать интерфейс для работы с функциями деканата. Для этого создается класс: GroupMenu.

В программе “Работа деканата” было создано 9 классов (2 абстрактных) и 3 интерфейса, которые реализуют работу деканата.

Диаграмма классов представлена на Рисунок 1.

UML Диаграмма классов:

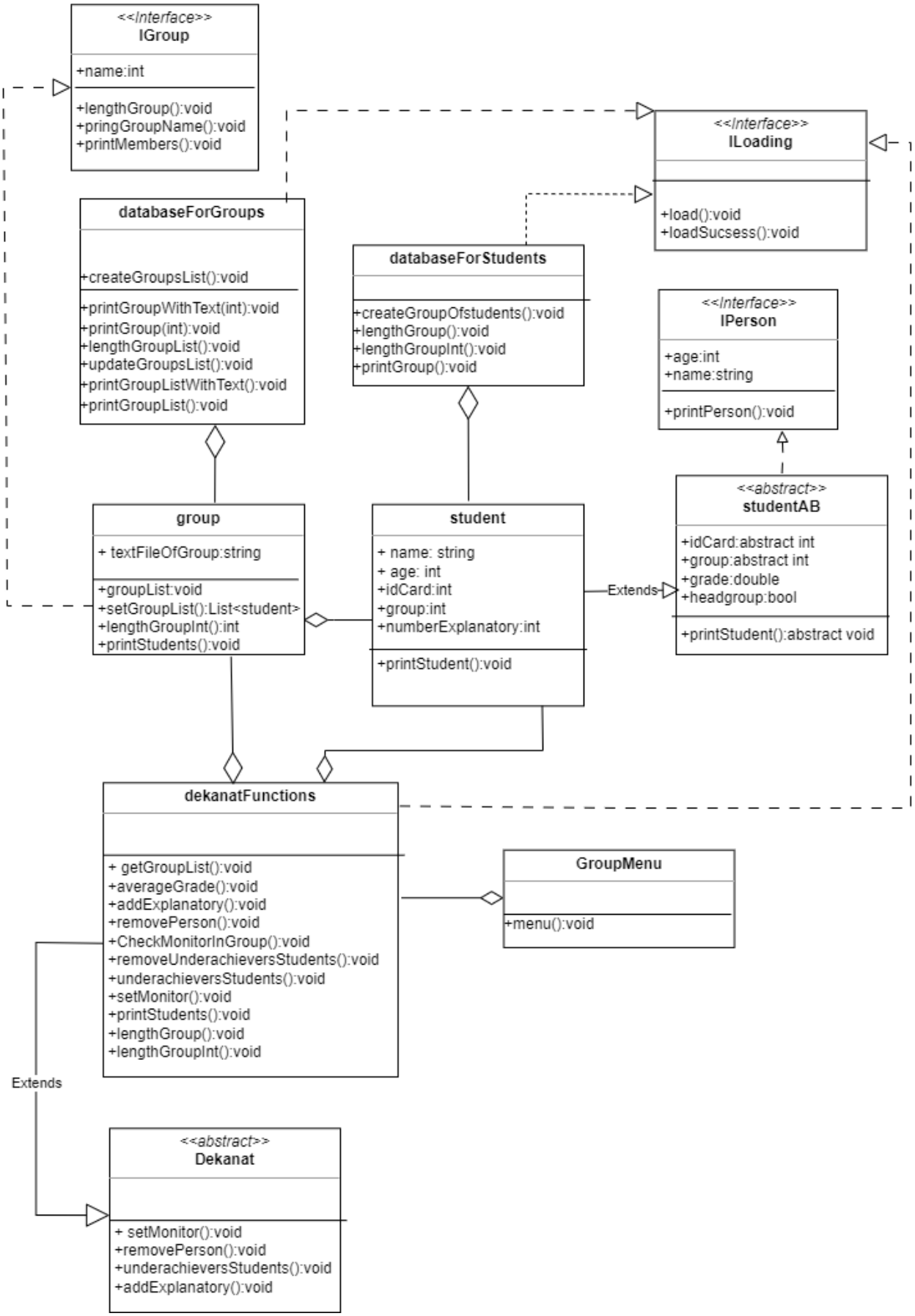

Рисунок 1 – Диаграмма классов UML

4. Спецификация программы

Описание методов класса Program представлено в табл.1.1

Таблица 1.1

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
load	void	public	-		Имитация загрузки
loadSucsess	void	public	-		Завершение загрузки
main	void	static	string[] args		Точка входа приложения

Описание методов интерфейса ILoading представлено в табл.1.2

Таблица 1.2

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
load	void	-	-		Имитация загрузки
loadSucsess	void	-	-		Завершение загрузки

Описание методов интерфейса IPerson представлено в табл.1.3

Таблица 1.3

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
printPerson	void	-	-		Вывод информации

Описание методов абстрактного класса studentAb представлено в табл.1.4

Таблица 1.4

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
printPerson	void	public	-		Вывод информации о человеке

printStudent	void	abstract public	-		Вывод информации о студенте

Описание методов класса student представлено в табл.1.5

Таблица 1.5

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
printStudent	void	public	-		Вывод информации о студенте

Описание методов интерфейса IGroup представлено в табл.1.6

Таблица 1.6

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
printMembers	void	-	-		Вывод членов группы
lengthGroup	void	-	-		Вывод длины группы

Описание методов класса GroupMenu представлено в табл.1.7

Таблица 1.7

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
GroupMenu	-	public	dekanatFunctios		Конструктор
menu	void	public	List<student>		Меню для работы с группой

Описание методов класса group представлено в табл.1.8

Таблица 1.8

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
groupList	void	public	List<student>		Присваиванние студентов группе
setGroupList	List<student>	public	-		Передача студентов
lengthGroupInt	int	public	-		Кол-во студентов (значение)
lengthGroup	void	public	-		Кол-во студентов
printGroupName	void	public	-		Вывод номера группы
printMembers	void	public	-		Вывод студентов в группе

Описание методов класса dekanatFunction представлено в табл.1.9

Таблица 1.9

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Назначение
load	void	public	-	Имитация загрузки
loadSucsess	void	public	-	Завершение загрузки
getGroupList	void	public	-	Присваивание списка студентов
averageGrade	double	public	-	Средний балл группы
CheckMonitorInGroup	void	public	-	Вывод старосты (+ назначение)
removeUnderachieversStudents	void	public	-	Отчисление неуспевающих
addExplanatory	void	public	int	Написать объяснительную
removePerson	void	public	int	Отчисление студента
underachieversStudents	void	public	-	Список неуспевающих
setMonitor	void	public	int	Назначить старосту
printStudents	void	public	-	Вывод студентов
lengthGroup	void	public	-	Кол-во студентов
lengthGroupInt	int	public	-	Кол-во студентов (значение)

Описание методов абстрактного класса Dekanat представлено в табл.1.10

Таблица 1.10

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Назначение
addExplanatory	void	public	int	Написать объяснительную
removePerson	void	public	int	Отчисление студента
underachieversStudents	void	public	-	Список неуспевающих

setMonitor	void	public	int	Назначить старосту

Описание методов абстрактного класса dataBaseForStudents представлено в табл.1.11

Таблица 1.11

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Назначение
load	void	public	-	Имитация загрузки
loadSucsess	void	public	-	Завершение загрузки
databaseForStudents	-	public	List<student> string	Конструктор
createGroupOfStudents	void	public	-	Считывание файла со студентами
lengthGroup	void	public	-	Кол-во студентов
lengthGroupInt	int	public	-	Кол-во студентов(значение)
printGroup	void	public	-	Вывод студентов

Описание методов абстрактного класса dataBaseForGroups представлено в табл.1.12

Таблица 1.12

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Назначение
load	void	public	-	Имитация загрузки
loadSucsess	void	public	-	Завершение загрузки
databaseForGroups	-	public	List<group>	Конструктор
createGroupsList	void	public	-	Считывание файл с группами
printGroupWithText	void	public	int	Вывод группы с текст файлом студентов
printGroupListWithText	void	public	-	Вывод групп с текст файлом студентов
printGroup	void	public	int	Вывод номера группы
printGroupList	void	public	-	Вывод номеров групп
lengthGroupList	int	public	-	Кол-во групп
updateGroupsList	void	public	-	Обновить группы

Файлы, необходимые для программы(данные отделены запятой):Ш

Название файла	Пример	Что хранится в файле
group.txt	2322,students2322.txt	Название группы, текст документ со студентами группы
students2117.txt	Буй Олег,2117,19, 1113321,false,4.6,6	ФИО, группа, возраст, номер студенческого, является ли студент старостой, успеваемость, кол-во объяснительных

5. Руководство оператора

С доступными функциями (с выборами) пользователь работает с помощью консоли. Он должен вводить соответствующий номер функции или слово для того или иного выбора. При вводе несуществующего числа, программа предлагает снова сделать выбор.

Изначально программа запрашивает разрешение пользователя о чтении файла с группами (название группы и название текстового файла со студентами группы). Группы заносятся в соответствующий список групп, после чего пользователю предлагается вывести на консоль считанные данные (список групп и соответствующие им текстовые файлы)

Пользователь может выбрать, с какой группой он будет работать. После выбора выходит меню доступных функций. После методов исключения из группы, исключенные студенты также удаляются из исходных списков (текстовых документов).

Список доступных функций работы с группой:

- Вывод списка студентов группы на экран, где у каждого студента есть такие параметры, как: фио, возраст, успеваемость, кол-во

объяснительных, группа, студенческий билет. Если студент является старостой, то программа выводит соответствующее сообщение.

- Исключение студентов по списку, либо исключать список неуспевающих студентов
- Отправка студента писать объяснительную за пропуск пары
- Вывод на экран кол-ва студентов в группе,
- Вывод средней успеваемости группы,
- Вывод старосты группы (если старосты нет, то назначается новый староста).

После проведенной работы над группой, пользователь может вернуться к выбору групп. Также он может выйти из программы.

6. Руководство программиста

Характеристика программы

Программа является консольным приложением Console App (.NET Framework), созданным в среде Microsoft Visual Studio. Используемый язык программирования - C#

Выходные и выходные данные

Входными данными являются текстовые файлы: файл со списком групп (название группы и название файла, в котором хранится список студентов группы), списки групп (в них находятся характеристики студентов). Все данные вводятся с разделителем в виде запятой (‘,’).

Выходными данными являются измененные входные файлы (при отчислении студентов, они удаляются из текстовых документов)

7. Контрольный пример

Далее представлены результаты выполнения программы (Рисунки 2,3,4,5,6,7,8).

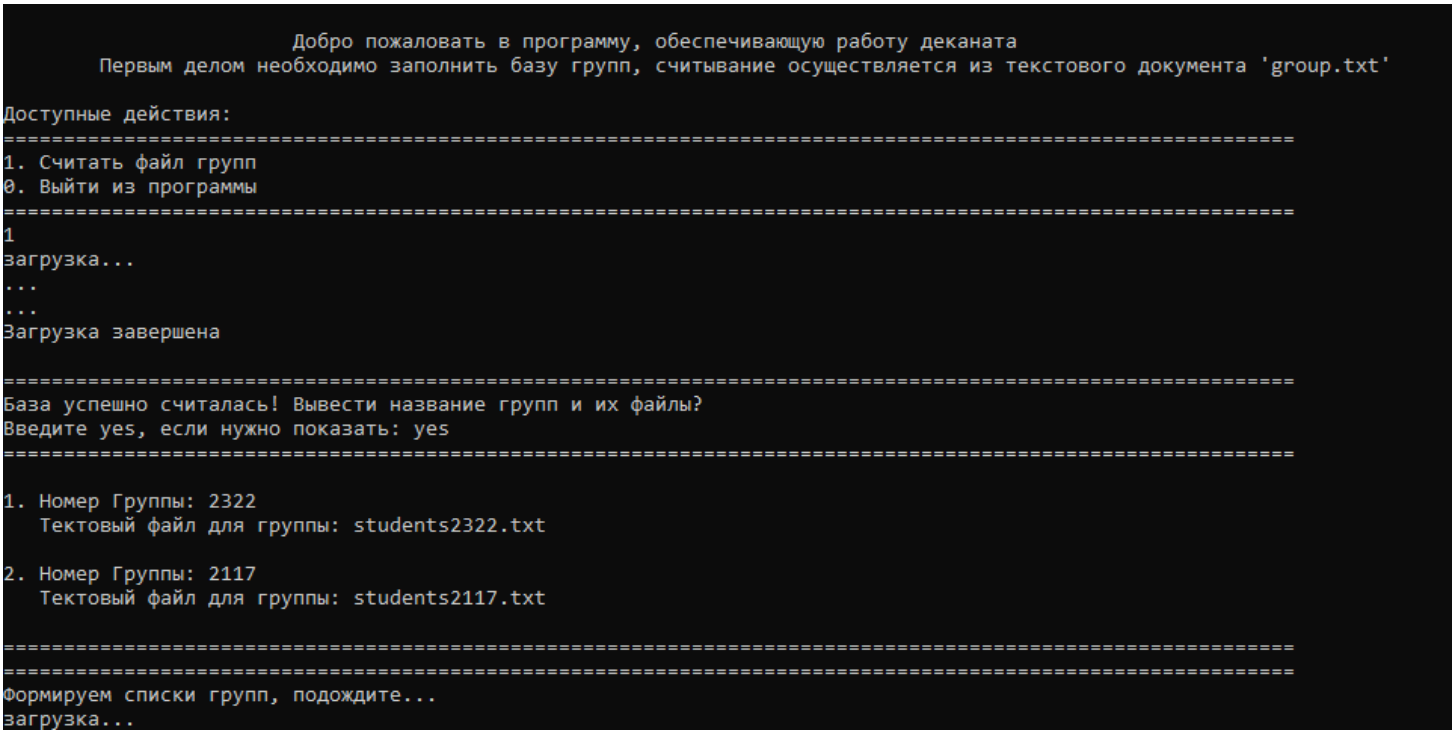

Рисунок 2 - Контрольный пример для считывания групп

Из рисунка видно, что программа считывает файл ‘group.txt’, откуда считывает данные о 2 группах, в которых хранятся название группы и текст файлы, в которых хранятся списки студентов

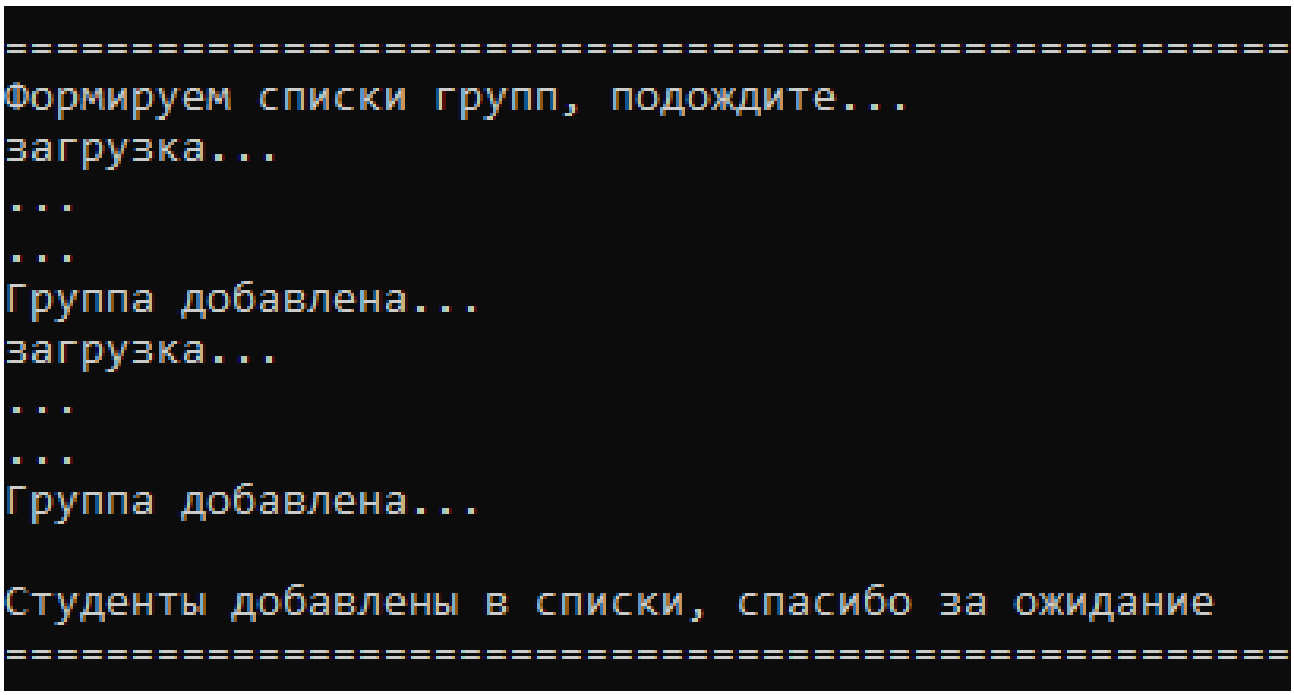

Рисунок 3 - Контрольный пример для имитации загрузки

Из рисунка видно, что программа считывает списки групп, также имитируя процесс загрузки.

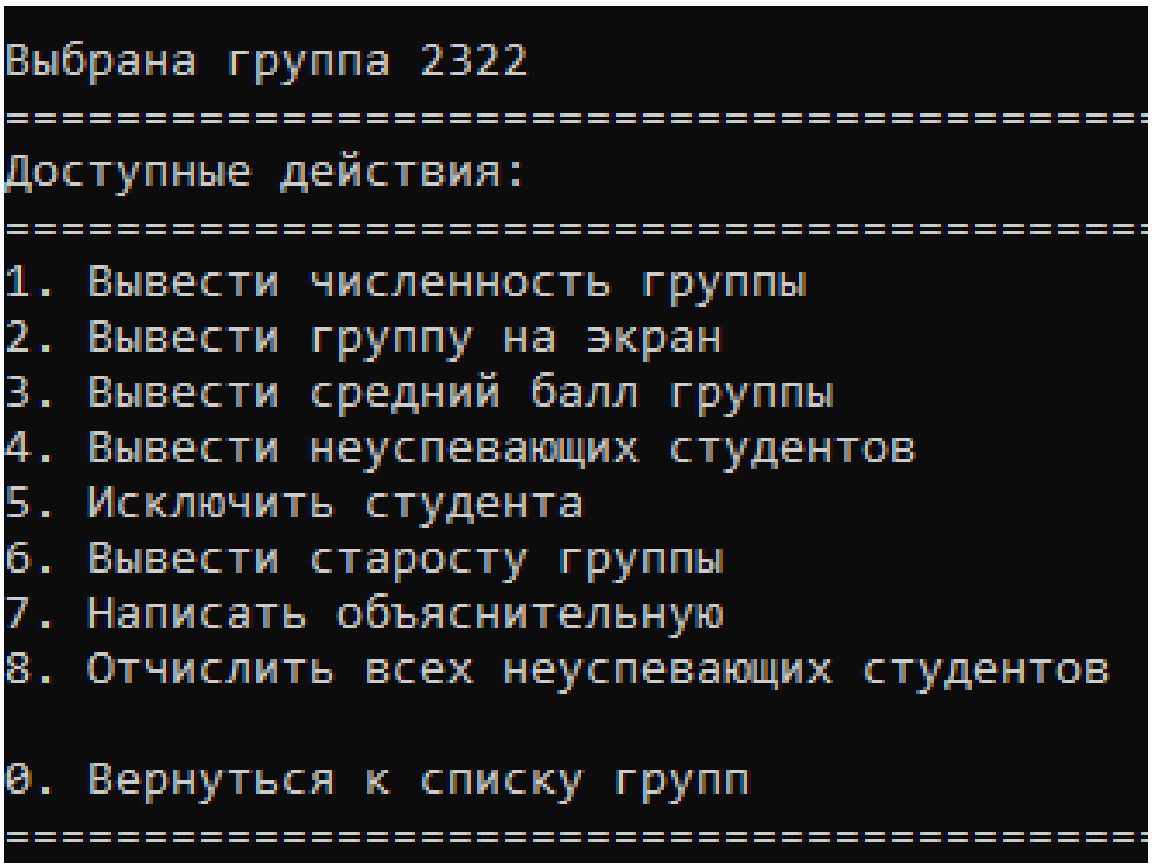

Рисунок 4 - Контрольный пример для отображения функций деканата

Из рисунка видно, какие функции доступны деканату при работе с той или иной группой.

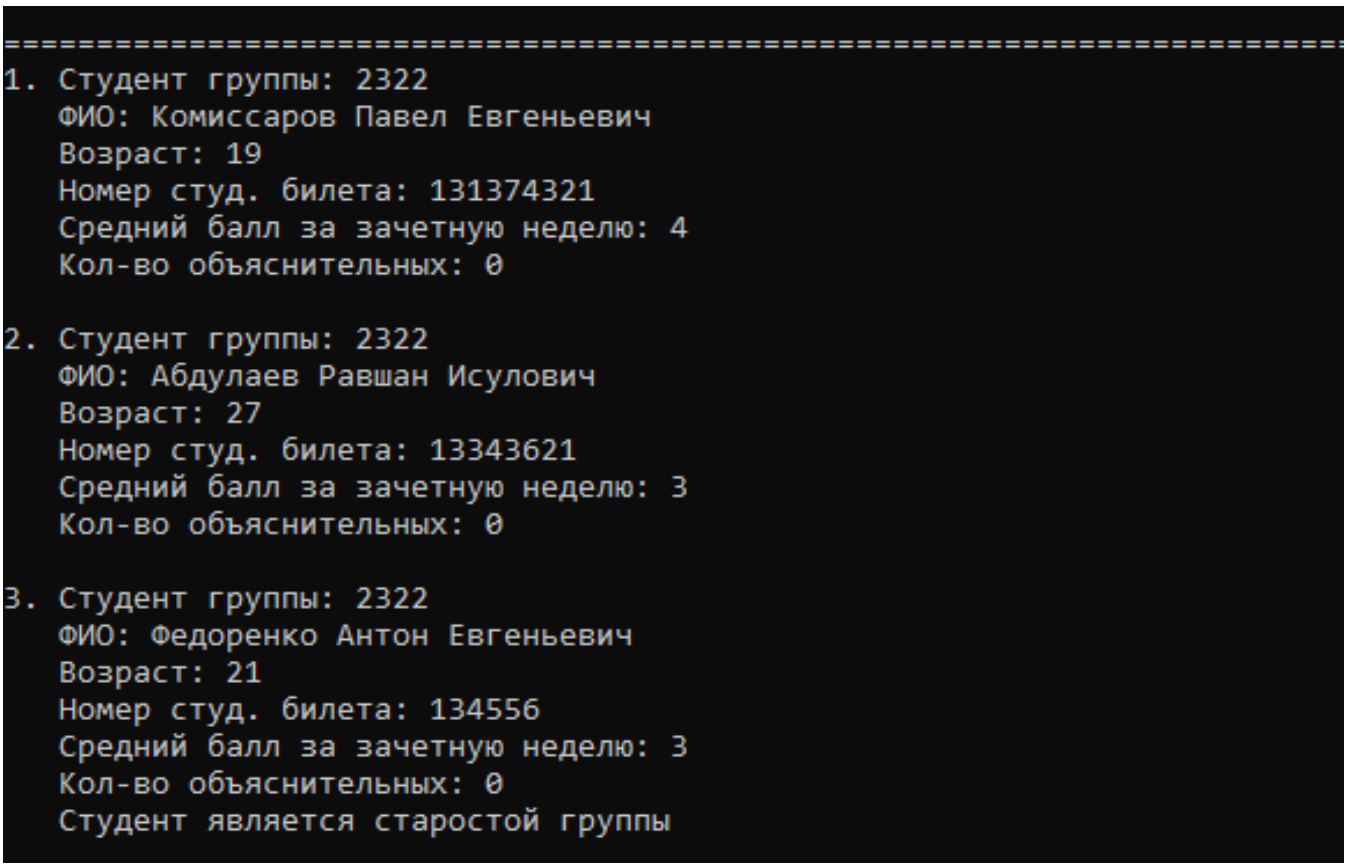

Рисунок 4 - Контрольный пример для отображения списка студентов

Из рисунка видно, как выводится список студентов. Они выводятся с нумерацией порядкового номера. Каждый студент набор данных, которые хранятся в файле со всеми студентами в группе.

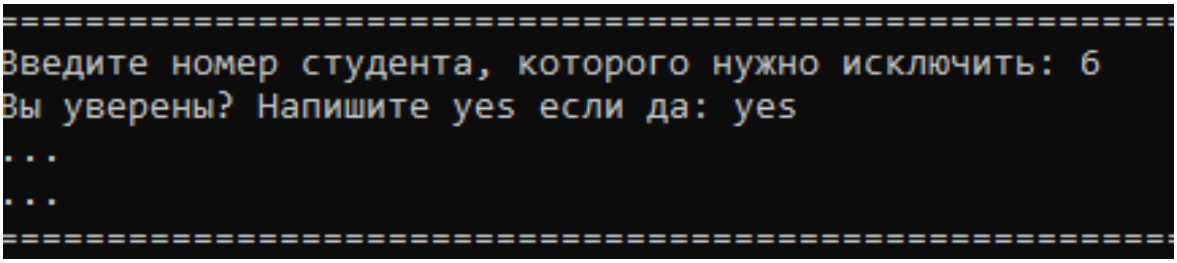

Рисунок 5 - Контрольный пример доп. подтверждения

Из рисунка видно, что при исключении студента требуется дополнительное подтверждение выбора, чтобы избежать случайного выбора

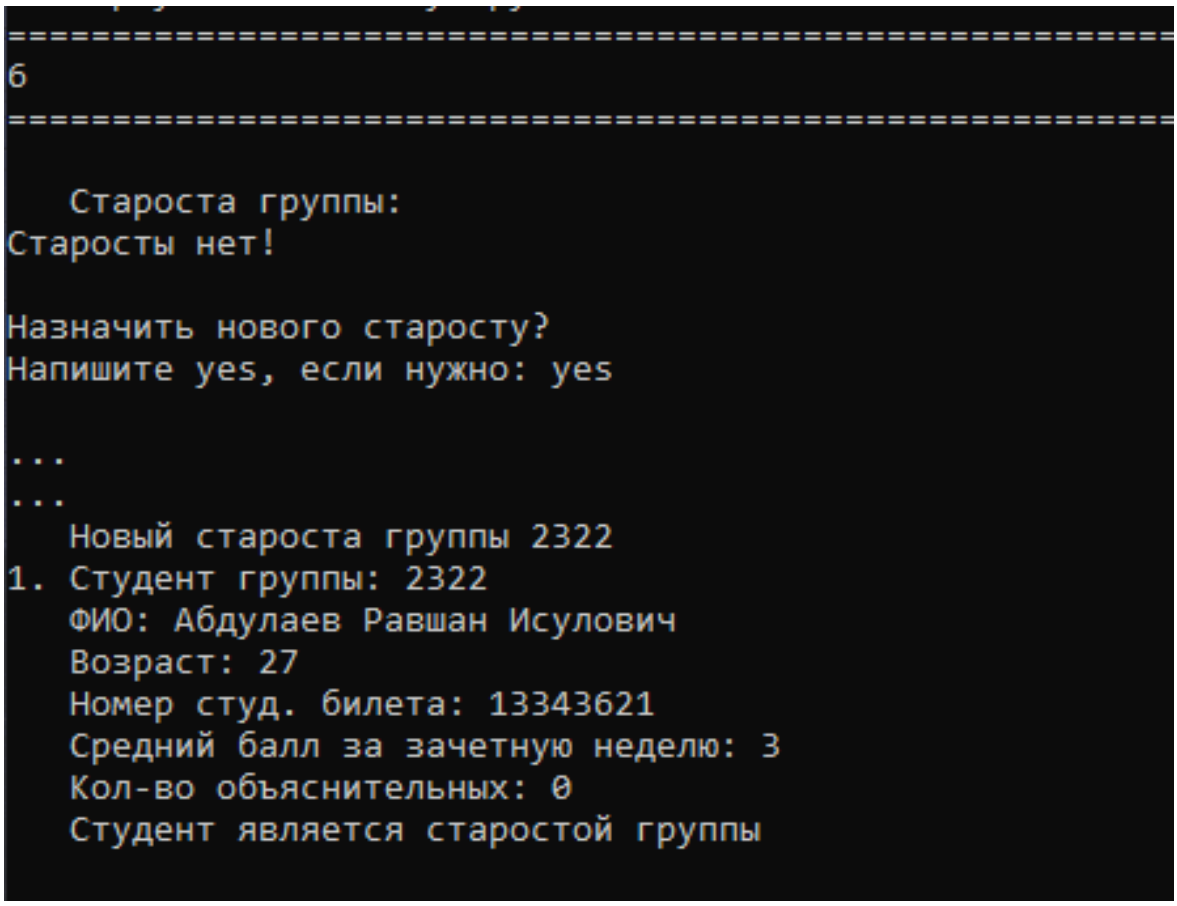

Рисунок 6 - Контрольный пример для поиска старосты группы.

Из рисунка видно, что старосты в группе нет, поэтому программа предлагает назначить нового старосту группы (т.к. назначение старосты является выбором группы, в программе староста выбирается путем случайного распределения)

```
================================================================================
8
================================================================================

Список неуспевающих студентов:
1. Студент группы: 2322
   ФИО: Кук Кукуев
   Возраст: 21
   Номер студ. билета: 1232131
   Средний балл за зачетную неделю: 2
   Кол-во объяснительных: 0

Отчисление студента Кук Кукуев, средний балл студента: 2
...
...
```

Рисунок 7 - Контрольный пример для удаления неуспевающих студентов

Из рисунка видно, что при выборе исключения списка неуспевающих студентов, программа изначально выводит список таких студентов, далее поочередно их исключает, выводя имя студента и его средний балл

8. Листинг программы

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace oopCourse
{
    class Program : ILoading
    {
        public void load()
        {
            Console.WriteLine("загрузка...");
            System.Threading.Thread.Sleep(500);
            Console.WriteLine("...");
            System.Threading.Thread.Sleep(500);
            Console.WriteLine("...");
            System.Threading.Thread.Sleep(500);
        }       //функция для имитации загрузки
        public void loadSucsess()
        {
            Console.WriteLine("Загрузка завершена");
        }

         static void Main(string[] args)
```

```
        {

            List<group>groups=new List<group>();//список групп(название+текст файл)
            List<student> studentsInGroup2322 = new List<student>(); //список студент
ов группы 2322
            List<student> studentsInGroup2117 = new List<student>(); //список студент
ов группы 2117

            databaseForGroups DataGroups = new databaseForGroups(groups);  //база для
 групп (считывание файла)

            Console.WriteLine();
            Console.WriteLine("\t\t\tДобро пожаловать в программу, обеспечивающую раб
оту деканата");
            Console.WriteLine("\tПервым делом необходимо заполнить базу групп, считыв
ание осуществляется из текстового документа 'group.txt'");
            Console.WriteLine();
            int choiceInt;
            string choiceStr;
            bool choiceBool = false;

            do                          //меню для считывания из файла
            {
                try
                {
                    Console.WriteLine("Доступные действия: ");
                    Console.WriteLine("==========================================
============================================================");
                    Console.WriteLine("1. Считать файл групп");
                    Console.WriteLine("0. Выйти из программы");
                    Console.WriteLine("==========================================
============================================================");
                    choiceInt = int.Parse(Console.ReadLine());
                }
                catch
                {
                    Console.WriteLine("Введено не число! Попробуйте снова");
                    choiceInt = -1;
                };
                switch (choiceInt)
                {
                    case 1:
                        DataGroups.createGroupsList();
                        choiceBool = true;
                        break;
                    case 0:
                        Environment.Exit(0);
                        break;

                    default:
                        Console.WriteLine("Такого пункта нет! Попробуйте снова");
                        Console.WriteLine();
                        break;
                }
            } while (!choiceBool);      //проверки на ввод
            choiceBool = false;


            databaseForStudents DataStudents2322 = new databaseForStudents(studentsIn
Group2322, groups[0].textFileOfGroup); //создание базы для студентов группы 2322
```

```
            databaseForStudents DataStudents2117 = new databaseForStudents(studentsIn
Group2117, groups[1].textFileOfGroup); //создание базы для студентов группы 2117

            Console.WriteLine("=================================================
================================================");
            Console.WriteLine("Формируем списки групп, подождите... ");



            DataStudents2322.createGroupOfStudents();   //создание списков студентов
            groups[0].groupList(studentsInGroup2322);//присваивание группе списка сту
дентов
            DataStudents2117.createGroupOfStudents();
            groups[1].groupList(studentsInGroup2117);

            Console.WriteLine();
            Console.WriteLine("Студенты добавлены в списки, спасибо за ожидание");
            Console.WriteLine("=================================================
================================================");
            Console.WriteLine();

            GroupMenu menuGroup;            //класс, где находится меню
            do
            {
                Console.WriteLine("=================================================
================================================");
                Console.WriteLine("Доступные группы: ");
                DataGroups.printGroupList();        //вывод списка групп
                Console.WriteLine("=================================================
================================================");
                Console.WriteLine("Чтобы выйти из программы, нажмите 0 ");
                Console.Write("Выберите группу(ее порядковый номер): ");
                try
                {
                    choiceInt = int.Parse(Console.ReadLine());
                }
                catch
                {
                    Console.WriteLine("Введено не число! Попробуйте снова");
                    choiceInt = -1;
                };

                switch (choiceInt)      //выбор группы для работы с ней
                {
                    case 1:
                        Console.WriteLine();
                        Console.WriteLine("Выбрана группа {0}", groups[0].name);
                       dekanatFunctions groupList1 = new dekanatFunctions(groups[0]);
  //создание объекта класса для работы с группой
                        groupList1.getGroupList();//присваивание студентов
                        menuGroup = new GroupMenu(groupList1);  //генерация меню для
работы деканата
                        menuGroup.menu(studentsInGroup2322);

                        break;

                    case 2:
                        Console.WriteLine();
                        Console.WriteLine("Выбрана группа {0}", groups[1].name);
                        dekanatFunctions groupList2 = new dekanatFunctions(groups[1])
;
```

```
                        groupList2.getGroupList();//присваивание студентов
                        menuGroup = new GroupMenu(groupList2);
                        menuGroup.menu(studentsInGroup2117);

                        break;
                    case 0:
                        choiceBool = true;
                        break;

                    default:
                        Console.WriteLine("Такого пункта нет! Попробуйте снова");
                        Console.WriteLine();
                        break;
                }
            } while (!choiceBool);
            choiceBool = false;
            Console.WriteLine();
            Console.WriteLine("Для завершения программы нажмите любую клавишу...");
            Console.Read();
        }

    }
}
```

IPerson.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace oopCourse
{
    interface IPerson   //интерфейс человек
    {
        string name { get; set; }
        int age { get; set; }
        void printPerson();

    }
}
```

ILoading.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace oopCourse
{
    interface ILoading
    {
        void load();
        void loadSucsess();
    }
}
```

IGruop.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace oopCourse
{
    interface IGroup
    {
        void printMembers();        //вывести участников группы
        void lengthGroup();         //длина группы
        void printGroupName();      //вывод названия группы
        int name { get; set; }      //имя группы
    }
}
```

studentAB.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace oopCourse
{
    abstract class studentAb : IPerson
    {
       abstract public string name { get; set; }
       abstract public int idCard { get; set; }     //студак
       abstract public int age { get; set; }
       abstract public int group { get; set; }
       public double grade { get; set; }   //успеапемость
       public bool headgroup { get; set; }          //является ли старостой
        public void printPerson()       //вывод информации о человеке
        {
            Console.WriteLine("   ФИО: {0}", name);
            Console.WriteLine("   Возраст: {0}", age);
        }

        abstract public void printStudent();
    }
}
```

student.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace oopCourse
{
    class student :studentAb
    {
        public override string name { get; set; }
        public override int idCard { get; set; }    //студенческий билет
        public override int age { get; set; }
```

```
        public override int group { get; set; }
        public int numberExplanatory { get; set; }  //кол-во объяснительных
        public override void printStudent()     //вывод информации о студенте
        {
            if (group != 0)
            {
                Console.WriteLine("Студент группы: {0}", group);
            }
            printPerson();
            Console.WriteLine("   Номер студ. билета: {0}", idCard);
            Console.WriteLine("   Средний балл за зачетную неделю: {0}", grade);
            Console.WriteLine("   Кол-во объяснительных: {0}", numberExplanatory);
            if (headgroup == true) Console.WriteLine("   Студент является старостой г
руппы");
        }
    }
}
```

Group.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Globalization;

namespace oopCourse
{
    class group : IGroup
    {

        List<student> Group;
        public int name { get; set; }
        public string textFileOfGroup { get; set; } //файл со студентами
        public void groupList(List<student> GroupList)
        {
            Group = GroupList;
        }   //присваивание стписка студентов
        public List<student> setGroupList()
        {
            return Group;
        }   //передача студентов
        public int lengthGroupInt()
        {
            return Group.Count;
        }    //кол-во студентов в инте
        public void lengthGroup()
        {
            Console.WriteLine("Количество студентов в группе: {0}", lengthGroupInt())
;
        }    //кол-во студентов
        public void printGroupName()
        {
            Console.WriteLine("Номер Группы: {0}", name);
        }  //номер группы
        public void printMembers()
        {
```

```
            int count = 1;
            Console.WriteLine("=====================================================
=====================================================");
            foreach (student Student in Group)
            {
                Console.Write("{0}. ", count++);
                Student.printStudent();
                Console.WriteLine();

            }
            Console.WriteLine("=====================================================
=====================================================");
        }    //вывод всех студентов в группе

    }
}
```

databaseForStudents.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Globalization;

namespace oopCourse
{
    class databaseForStudents : ILoading      //создаем список группы
    {
        List<student> StudentsOfGroup;

        string fileGroup;
        public databaseForStudents(List<student> StudentsList,string fileGroup)
        {
            StudentsOfGroup = StudentsList; //список студентов
            this.fileGroup = fileGroup;     //файл со студентами
        }

        public void load()
        {
            Console.WriteLine("загрузка...");
            System.Threading.Thread.Sleep(500);
            Console.WriteLine("...");
            System.Threading.Thread.Sleep(500);
            Console.WriteLine("...");
            System.Threading.Thread.Sleep(500);
        }       //функция для имитации загрузки
        public void loadSucsess()
        {
            Console.WriteLine("Группа добавлена...");
        }
        public void createGroupOfStudents() //считывание файла
        {
            using (StreamReader reader = new StreamReader(fileGroup))   //файл для чт
ения
            {
                string line;
                while ((line = reader.ReadLine()) != null)  //считыванием строку
                {
                    string[] parts=line.Split(',');//делим на части с помощью запятой
```

```
                    student student = new student
                    {                                   //записываем студента из файла
                        name = parts[0],
                        group = int.Parse(parts[1]),
                        age = int.Parse(parts[2]),
                        idCard = int.Parse(parts[3]),
                        headgroup = bool.Parse(parts[4]),
                        grade=double.Parse(parts[5], CultureInfo.InvariantCulture), /
/конвертируем точку в запятую
                        numberExplanatory= int.Parse(parts[6])
                    };
                    StudentsOfGroup.Add(student);       //добавляем в список
                }
                reader.Close();
                load();
                loadSucsess();
            }
        }
        public void lengthGroup()
        {
            Console.WriteLine("Кол-
во студентов в группе: {0}", StudentsOfGroup.Count);
        }   //вывод длины группы

        public int lengthGroupInt()
        {
            return StudentsOfGroup.Count;
        }   //вывод длины группы в инте
        public void printGroup()        //вывод студентов
        {
            int count = 1;
            foreach (student student in StudentsOfGroup)
            {
                Console.Write("{0}. ",count++); //порядковый номер
                student.printStudent();
                Console.WriteLine();

            }
        }


    }
}
```

dataBaseForGroups.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Globalization;

namespace oopCourse
{
    class databaseForGroups  : ILoading   //создаем список групп
    {
        List<group> GroupsList;
        public databaseForGroups(List<group> GroupsList)
        {
            this.GroupsList = GroupsList;   //список групп (названия+файлы)
```

```
        }

        public void createGroupsList()      //считывание
        {
            using (StreamReader reader = new StreamReader("group.txt"))
            {

                string line;
                while ((line = reader.ReadLine()) != null)
                {
                    string[] parts = line.Split(',');
                    group Group = new group
                    {
                        name = int.Parse(parts[0]),
                        textFileOfGroup = parts[1]

                    };
                    GroupsList.Add(Group);
                }
            }
            load();
            loadSucsess();
            Console.WriteLine();
            Console.WriteLine("=====================================================
=================================================");
            Console.WriteLine("База успешно считалась! Вывести название групп и их фа
йлы?");
            Console.Write("Введите yes, если нужно показать: ");
            string choiceStr = Console.ReadLine();

            Console.WriteLine("=====================================================
=================================================");
            Console.WriteLine();
            if (choiceStr == "yes")     //вывод списка групп с их текст документами
            {

                printGroupListWithText();

            }
            Console.WriteLine("=====================================================
=================================================");
        }

        public void load()
        {
            Console.WriteLine("загрузка...");
            System.Threading.Thread.Sleep(500);
            Console.WriteLine("...");
            System.Threading.Thread.Sleep(500);
            Console.WriteLine("...");
            System.Threading.Thread.Sleep(500);
        }       //функция для имитации загрузки
        public void loadSucsess()
        {
            Console.WriteLine("Загрузка завершена");
        }

        public void printGroupWithText(int number)//вывод одной группы с текст файлом
        {
            Console.WriteLine("Номер Группы: {0}", GroupsList[number].name);
```

```
            Console.WriteLine("   Тектовый файл для группы: {0}", GroupsList[number].
textFileOfGroup);

        }
        public void printGroup(int number)
        {
            Console.WriteLine("Номер Группы: {0}", GroupsList[number].name);
        } //вывод одной группы
        public int lengthGroupList()
        {
            return GroupsList.Count;
        } //кол-во групп
        public void updateGroupsList()
        {
            GroupsList.Clear();
            createGroupsList();
        }   //обновить списки групп
        public void printGroupListWithText()    //вывод групп с текст файлами
        {
            int count = 1;
            foreach (group Groups in GroupsList)
            {
                Console.Write("{0}. ", count);
                printGroupWithText(count-1);
                count++;
                Console.WriteLine();
            }
        }
        public void printGroupList()
        {
            int count = 1;
            foreach (group Groups in GroupsList)
            {
                Console.Write("{0}. ", count++);
                Groups.printGroupName();
                Console.WriteLine();
            }
        }//вывод групп
    }
}
```

Dekanat.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace oopCourse
{
    public abstract class Dekanat
    {
        abstract public void setMonitor(int index); //назначить старосту
        abstract public void removePerson(int index);//отчислить студента
        abstract public void underachieversStudents();//список неуспевающих
        abstract public void addExplanatory(int index);//заставить писать объясн-ую
    }
}
```

dekanatFunctions.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace oopCourse
{
    class dekanatFunctions : Dekanat, ILoading
    {
        List<student> StudentsGroup;
        group Group;
        public dekanatFunctions(group Group)
        {
            this.Group = Group;
        }
        public void getGroupList()
        {
            StudentsGroup=Group.setGroupList();
        }   //присваивание стписка студентов
        public double averageGrade()
        {
            double summ = 0;
            foreach (student Student in StudentsGroup)
            {
                summ += Student.grade;

            }
            return summ / StudentsGroup.Count();
        }   //средний балл всей группы
        public void CheckMonitorInGroup()
        {

            Console.WriteLine("   Староста группы:");

            int count = 1;
            foreach (student Student in StudentsGroup)
            {
                if (Student.headgroup)
                {
                    Console.Write("{0}. ", count++);
                    Student.printPerson();
                    Console.WriteLine();
                }
            }
            if (count == 1)     //если старосты нет, предложение назначить
            {
                string choice;

                Console.Write("Старосты нет!\n\nНазначить нового старосту?\nНапишите
yes, если нужно: ");
                choice = Console.ReadLine();
                Console.WriteLine();
                if (choice == "yes")
                {
                    Random rand = new Random();
                    int random = rand.Next(0, Group.lengthGroupInt());
                    setMonitor(random);
                }
            }
```

```
        }   //проверка на наличие старосты(+добавление)

        public void removeUnderachieversStudents()  //удадение неуспевающих студентов
        {
            underachieversStudents();   //список неуспевающих
            int count = 0;
            List<int> indexForRemove = new List<int>(); //список, в котором хранятся
индексы неуспевающих студентов
            foreach (student Student in StudentsGroup)      //поиск студентов
            {
                count++;
                if (Student.grade < 3.0)
                {
                    Console.WriteLine("Отчисление студента {0}, средний балл студента
: {1} ", Student.name, Student.grade);
                    load();
                    Console.WriteLine();
                    count--;
                }
            }
            foreach (int index in indexForRemove)   //удаление
            {
                removePerson(index);
            }
            indexForRemove.Clear();
        }

        public override void addExplanatory(int student)
        {
            Console.WriteLine("Студент {0} принес(ла) объяснительную за пропуск пары"
, StudentsGroup[student].name);
            StudentsGroup[student].numberExplanatory += 1;
            Console.WriteLine("Кол-
во объяснительных у студента: {0}", StudentsGroup[student].numberExplanatory);
        }   //объяснительная

        public override void removePerson(int number)   //отчисление студента
        {
            StudentsGroup.RemoveAt(number - 1); //удаляем из списка
            FileStream fileStream = File.Open(Group.textFileOfGroup, FileMode.Open);
            fileStream.SetLength(0);    //отчистка файла
            fileStream.Close();

            using (StreamWriter writer = new StreamWriter(Group.textFileOfGroup)) //п
ереписываем файл
            {
                foreach (student student in StudentsGroup)
                {
                    writer.WriteLine($"{student.name},{student.group},{student.age},{
student.idCard},{student.headgroup},{student.grade},{student.numberExplanatory}");
                }
            }
        }
        public void load()
        {
            System.Threading.Thread.Sleep(500);
            Console.WriteLine("...");
            System.Threading.Thread.Sleep(500);
            Console.WriteLine("...");
            System.Threading.Thread.Sleep(500);
        }   //имитация загрузки
```

```
        public void loadSucsess()
        {
            Console.WriteLine("Готово!");
        }
        public override void underachieversStudents()
        {
            Console.WriteLine("Список неуспевающих студентов:");

            int count = 1;
            foreach (student Student in StudentsGroup)
            {
                if (Student.grade < 3.0)
                {
                    Console.Write("{0}. ", count++);
                    Student.printStudent();
                    Console.WriteLine();
                }
            }
            if (count == 1) { Console.WriteLine("Неуспевающих нет");Console.WriteLine
(); } //если никого не вывели, то все хорошо учатся
        }   //неуспевающие студенты (србалл<3)
        public override void setMonitor(int index)
        {
            load();
            StudentsGroup[index].headgroup = true;  //отмечаем старосту
            Console.WriteLine("   Новый староста группы {0}", Group.name);

            Console.Write("{0}. ", 1);
            StudentsGroup[index].printStudent();
            Console.WriteLine();
        }   //назначить старосту (случайным образом)
        public void printStudents()
        {
            Group.printMembers();
        }    //вывод всех студентов в группе
        public void lengthGroup()
        {
            Group.lengthGroup();
        }    //кол-во студентов
        public int lengthGroupInt()
        {
            return Group.lengthGroupInt();
        }    //кол-во студентов в инте

    }
}
```

groupMenu.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace oopCourse
{
    class GroupMenu
    {

        dekanatFunctions groupList;
```

```
        public GroupMenu(dekanatFunctions groupList)       //конструктор
        {
            this.groupList = groupList;
        }


        public void menu(List<student> students)
        {

            int choiceInt;
            string choiceStr;


            do                      //меню
            {
                try
                {
                    Console.WriteLine("=========================================
============================================================");
                    Console.WriteLine("Доступные действия: ");
                    Console.WriteLine("=========================================
============================================================");
                    Console.WriteLine("1. Вывести численность группы");
                    Console.WriteLine("2. Вывести группу на экран");
                    Console.WriteLine("3. Вывести средний балл группы");
                    Console.WriteLine("4. Вывести неуспевающих студентов");
                    Console.WriteLine("5. Исключить студента");
                    Console.WriteLine("6. Вывести старосту группы");
                    Console.WriteLine("7. Написать объяснительную");
                    Console.WriteLine("8. Отчислить всех неуспевающих студентов");
                    Console.WriteLine();
                    Console.WriteLine("0. Вернуться к списку групп");
                    Console.WriteLine("=========================================
============================================================");
                    choiceInt = int.Parse(Console.ReadLine());
                    Console.WriteLine("=========================================
============================================================");
                    Console.WriteLine();
                }
                catch
                {
                    Console.WriteLine("Введено не число! Попробуйте снова");

                    choiceInt = -1;
                };
                switch (choiceInt)
                {
                    case 1:     //численность группы
                        groupList.lengthGroup();
                        Console.WriteLine();
                        break;

                    case 2:     //вывод студентов на экран
                        groupList.printStudents();
                        break;
                    case 3: //подсчет среднего балла
                        double summ=groupList.averageGrade();
                        Console.Write("Средний балл по группе: {0}", summ);
                        Console.WriteLine();
                        Console.WriteLine();
                        break;
```

```
                    case 4:     //список неуспевающих студентов
                        groupList.underachieversStudents();
                        break;
                    case 5:     //отсчисление
                        groupList.printStudents();//вывод списка студентов для выбора
                        do
                        {
                Console.Write("Введите номер студента, которого нужно исключить: ");

                            try         //ввод с проверками
                            {
                                choiceInt = int.Parse(Console.ReadLine());
                            }
                            catch
                            {
                             Console.WriteLine("Введено не число! Попробуйте снова");
                                choiceInt = -1;
                            }
                         if (choiceInt > groupList.lengthGroupInt() || choiceInt < 1)
                            {
                                Console.WriteLine("Ошибка ввода! Попробуйте снова");
                            }
                        } while (choiceInt > groupList.lengthGroupInt() || choiceInt
< 1) ;
                        Console.Write("Вы уверены? Напишите yes если да: ");
                        choiceStr = Console.ReadLine(); //подтверждение выбора
                        if (choiceStr == "yes")
                        {
                            groupList.load();       //загрузка
                            groupList.removePerson(choiceInt);  //удаление
                        }
                        break;
                    case 6:     //проерка наличия старосты(+добавление)
                        groupList.CheckMonitorInGroup();
                        break;
                    case 7:     //принести объяснительную
                        int number = 0;
                        groupList.printStudents() //вывод списка студентов для выбора
                        do {        //выбор с проверками
                            Console.Write("Напишите номер студента, который должен пр
инести объяснительную: ");
                            try
                            {
                                number = int.Parse(Console.ReadLine());
                            }
                            catch
                            {
                             Console.WriteLine("Введено не число! Попробуйте снова");
                                number = -1;
                            }
                            if(number > groupList.lengthGroupInt() || number < 1)
                            {
                                Console.WriteLine("Ошибка ввода! Попробуйте снова");
                            }
                        } while (number > groupList.lengthGroupInt() || number < 1) ;
                        groupList.load();   //загрузка
                        groupList.addExplanatory(number-1);
                        break;
                    case 8:     //отсчисление всех неуспевающих
                        groupList.removeUnderachieversStudents();
                        break;
```

```
                    case 0:

                        break;
                    default:
                        Console.WriteLine("Такого пункта нет! Попробуйте снова");
                        Console.WriteLine();
                        break;
                }
            } while (choiceInt!=0);
        }
    }
}
```

Список используемой литература

1. Microsoft Learn – сеть разработчиков Microsoft. URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 24.11.2023)

2. Moodle – виртуальный образовательный кластер: https://vec.etu.ru/moodle/course/view.php?id=14794 (дата обращения: 24.11.2023)

3. UML. Руководство пользователя– Джеймс Рамбо, Грейди Буч, Айвар Джекобсон. Издательство: Питер, 2004 г. (дата обращения: 20.11.2023)