МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І.І.МЕЧНИКОВА

Індивідуальне завдання:

на тему:

«Школа-варіант12а»

Виконала:

Студентка 1 курсу зі спеціальності комп’ютерна інженерія

Луценко Аліна Андріївна

Викладач:

Антоненко Олександр Сергійович

Одеса-2023

Задание: Создать классы: «Человек» (ФИО, возраст, рост), «Ученик» (класс, список оценок), «Учитель» (дополнительное поле: предмет, который преподает), «Оценка» (вид: домашняя, контрольная, самостоятельная, экзамен, итоговая; предмет).

Рекомендуется оценки ученика хранить в векторе объектов типа оценка. Для учителя хранить вектор указателей на учеников, которые у него учатся.

Определить метод – вывести итоговую оценку ученика (учитывать текущие оценки и оценку за экзамен, но только по этому предмету). Для каждого учителя определить среднюю оценку из тех, что он выставлял. Для этого перебрать оценки учеников, которые учатся у этого учителя и выбрать только оценки по предмету, который он преподает.

(Подразумеваются также методы: ввода, распечатки, добавления оценок и прочие необходимые).

Классы:

class Human – базовый класс для Студента и Учителя.

Поля:

```
private string fullname; //полное имя человека

private int age; //возраст

private double height; //рост
```

Свойства, для доступа к приватным полям:

```
public string Fullname
    {
        get
        {
            return fullname;
        }
        protected set
        {
            fullname = value;
        }
    }
```

```
public int Age
    {
        get
        {
            return age;
        }
        protected set
        {
            age = value;
        }
    }


public double Height
    {
        get
        {
            return height;
        }
        protected set
        {
            height = value;
        }
    }
```

Конструкторы:

```
public Human()  //конструктор без параметров, по умолчанию
    {
        fullname = "Без имени";
        age = 0;
        height = 0;
    }

public Human(string fullname, int age, double height) //конструктор с параметрами
    {
        if (age < 0 || height < 0)
        {
            throw new ArgumentException("Неправильные данные");
        }
        this.fullname = fullname;
        this.age = age;
        this.height = height;
    }
```

Если рост и возраст меньше нуля, то сгенерируется исключение с сообщением «Неправильные данные»

Виртуальный метод Инфо, который возвращает информацию о человеке, будет потом переопределяться в классах-наследниках

```
public virtual string Info()
    {
        string nameCorrected = fullname.Replace("_", " ");
        return "ФИО: " + nameCorrected + "  Возраст: " + age + "  Рост: " + height;
    }
```

Так как я введенный текст в консоле разделяю по пробелам, то фамилию имя и отчества нужно написать через нижнее подчеркивание, но когда мы захотим посмотреть информацию о человеке, то эти нижние подчеркивания я заменяю на пробелы.

class Student – представляет ученика.

Класс Студент наследуется от класса Человек.

Класс Student наследует все поля, свойства и методы класса Human, такие как Fullname, Age, Height, и Info()

Дополнительные поля:

```
private List<Grade> gradesList = new List<Grade>(); //лист типа оценка, который содержит элемент типа оценка.

private int classNumber;
```

И свойства к ним:

```
public List<Grade> GradesList
    {
        get
        {
            return gradesList;
        }
        private set
        {
            gradesList = value;
        }
    }

public int ClassNumber
    {
        get
        {
            return classNumber;
        }
        private set
        {
            classNumber = value;
        }
    }
```

Конструктор:

```
public Student(string fullname = "Без имени", int age = 0, double height = 0,
int classNumber = 0)
    {
        if (age < 0 || height < 0 || classNumber < 0 || classNumber > 12)
        {
            throw new ArgumentException("Неправильные данные");
        }

        Fullname = fullname;
        Age = age;
        Height = height;
        this.classNumber = classNumber;

    }
```

Это конструктор с параметрами, он инициализирует поля класса Human и номер класса студента. В нем также содержатся параметры по умолчанию.

При попытке ввести отрицательный рост, возраст и номер класса не из промежутка от 1 до 12 сгенерируется исключение.

Методы:

Метод, который возвращает среднюю оценку студента по типу работы и по предмету, которые мы укажем.

```
public double AverageGradeByType(WorkType workType, SubjectType subjectType)
    {
        double result = 0;
        int count = 0;
        foreach (Grade item in gradesList.Where(x => x.WorkTypeGrade == workType &&
x.Subject == subjectType)) //выбирает в списке оценок оценки по тем предметам и типу который
мы указали
        {
            result += item.GradeValue;
            count++;
        }

        if (count != 0)
        {
            result /= count;
        }

        return result;
    }
```

Метод подсчета финальной оценки ученика:

```
public int FinalGrade(SubjectType subjectType)
    {
```

```
        double result = 0;
        double[] coefficient = { 0.2, 0.2, 0.1, 0.2, 0.3 }; //коэффициенты к типам работы
        int workTypesCount = Enum.GetNames(typeof(WorkType)).Length; // находит длину енума
SubjectGradeType
        for (int i = 0; i < workTypesCount; i++)
        {
            double averageValue = AverageGradeByType((WorkType)i+1, subjectType); //Получить
значения енума по индексу
            averageValue *= coefficient[i];
            result += averageValue;
        }
        return (int)Math.Round(result);
    }
```

Метод добавления оценки ученику в студент лист

```
public void AddGrade(Grade mark)
    {
        gradesList.Add(mark);
    }
```

Переписываем метод инфо теперь для студента с добавлением параметра номер класса

```
    public override string Info()
    {
        return base.Info() + "  Класс: " + classNumber;
    }
```

Посмотреть информацию о всех оценках студента

```
    public string AllGradesInfo()
    {
        string result = "";

        foreach (Grade grade in gradesList)
        {
            result += grade.Info() + "\n";
        }
        return result;
    }
```

Получаем оценки студента по определенному предмету

```
    public string GetGradesInfoBySubject(SubjectType subject)
    {
        string result = "";

        foreach (Grade grade in gradesList.Where(x => x.Subject == subject)) //перебираем
оценки в списке оценок(только те у которых предмет соответствует тому предмету, который мы
ввели)
        {
            result += grade.Info() + "\n";
        }

        return result;
    }
```

Получаем оценки студента по типу работы

```
    public string GetGradesInfoByWork(WorkType work)
    {
        string result = "";

        foreach (Grade grade in gradesList.Where(x => x.WorkTypeGrade == work))
        {
            result += grade.Info() + "\n";
        }

        return result;
    }
```

class Teacher

Он также наследуется от класса Человек.

Класс Teacher наследует все поля, свойства и методы класса Human, такие как Fullname, Age, Height, и Info()

При создании объекта класса Teacher можно передать значения для полей базового класса Human и предмет, преподаваемый учителем.

Поля:

```
private SubjectType subject; //предмет, который ведет учитель типа енум
private List<Student> studentList = new List<Student>(); //у учителя есть список учеников,
которые у него учатся
```

Свойство:

```
public List<Student> StudentList
    {
        get
        {
            return studentList;
        }
        private set
        {
            studentList = value;
        }
    }
```

Конструктор:

```
public Teacher(string fullname = "Без имени", int age = 0, double height = 0, SubjectType
subject = SubjectType.Spells)
    {
        if (age < 0 || height < 0 || (int)subject >
Enum.GetNames(typeof(SubjectType)).Length || (int)subject < 1)
        {
            throw new ArgumentException("Неправильные данные");
        }
        Fullname = fullname;
        Age = age;
        Height = height;
        this.subject = subject;
```

```
    }
```

Конструктор с параметрами, инициализирует поля класса Human и предмет, преподаваемый учителем.

Методы:

Подсчитывает среднее значение всех оценок, который выставлял учитель

```
public double GetAllMarksAverage()
    {
        double result = 0;
        int count = 0;
        //У Студента взять оценки по тому предмету который ведет нужный нам учитель
        foreach (Student student in studentList)
        {
            foreach (Grade grade in student.GradesList.Where(x => x.Subject == subject))
            {
                result += grade.GradeValue;
                count++;
            }

        if (count != 0)
        {
            result /= count;
        }
        return result;

   }

public void AddStudent(Student student) //добавляет одного студента в студентЛист учителя
    {
        studentList.Add(student);
    }

public void AddStudentsList(List<Student> studentList) // добавляет в студентЛист учителя
сразу список студентов
    {
        this.studentList.AddRange(studentList);
    }

    public void AssignStudentList(List<Student> studentList) // меняет старых студентов на
новых, переприсваивает студентЛист
    {
        this.studentList = studentList;
    }
```

Переписывает метод базового класса и добавляет предмет, который ведет учитель

```
    public override string Info()
    {
        return base.Info() + "  Предмет: " + subject;
    }
```

Получает информацию о всех оценках, которые выставлял учитель

```
public string GetAllTeacherMarksInfo()
{
    string result = "";

    foreach (Student student in studentList)
    {
        foreach (Grade grade in student.GradesList.Where(x => x.Subject == subject))
        {
            result += grade.Info() + "\n";
        }
    }

    if(result == "")
    {
        result = "Учитель не выставлял оценок\n";
    }

    return result;
}
```

class Grade:

Поля:

Перечисления: (енумы)

```
public enum WorkType
{ Homework = 1, Control, Test, Exam, FinalExam }

public enum SubjectType
{ Alchemy = 1, Transfiguration, Spells, Herbology, Numerology, Astronomy }
```

Я использовала енумы для типов предметов и типов работ. Енум - это перечисление каких-то констант, они здесь удобны, код лучше с ними читается. В школе практически не добавляются какие-то предметы, а если и добавятся, то с енумом это не сложно делать, поэтому можно просто задать список с выбором предметов.

Отсчет я начала с единицы, так удобнее для пользователя, по умолчанию они начинаются с нуля.

```
private SubjectType subject = SubjectType.Alchemy; //поле типа енум SubjectType

private int grade; //поле для оценки в баллах

private WorkType workType; //поле типа енум WorkType
```

Чтобы обратиться к элементу енума, нужно написать тип_енума.элемент, к которому мы хотим обратиться

Свойства:

```
public SubjectType Subject
    {
        get
        {
            return subject;
        }
        private set
        {
            subject = value;
        }
    }

public int GradeValue
    {
        get
        {
            return grade;
        }
        private set
        {
            grade = value;
        }
    }

public WorkType WorkTypeGrade
    {
        get
        {
            return workType;
        }
        private set
        {
            workType = value;
        }
    }
```

Конструктор:

```
public Grade(SubjectType subject = SubjectType.Spells, int grade = 0, WorkType workType =
WorkType.Homework)
    {
        if ( grade < 0 || grade > 12 || (int)subject >
Enum.GetNames(typeof(SubjectType)).Length || (int)subject < 1 || (int)workType >
Enum.GetNames(typeof(WorkType)).Length || (int)workType < 1)
        {
            throw new ArgumentException("Некоректная оценка");
        }
        this.subject = subject;
        this.grade = grade;
```

```
        this.workType = workType;
    }
```

Конструктор с параметрами, инициализирует поля класса Grade с заданными значениями предмета, оценки и типа работы.

При попытке ввести оценку больше 12 и меньше 0, а также если номер предмета и типа работы выйдут за границы енума, сгенерируется ошибка.

```
public string Info() //метод инфо для класса оценка
    {
        return $"Предмет: {subject}, Оценка {grade}, Вид работы: {workType}";
    }
```

class School:

Класс-контейнер, который хранит в себе списки учеников и учителей, чтобы, мы могли с ними как-то в программе взаимодействовать.

Поля:

```
private List<Student> studentList = new List<Student>();
private List<Teacher> teacherList = new List<Teacher>();
```

Свойства:

```
public List<Student> StudentList
    {
        get
        {
            return studentList;
        }
        private set
        {
            studentList = value;
        }
    }

    public List<Teacher> TeacherList
    {
        get
        {
            return teacherList;
        }
        private set
        {
            teacherList = value;
        }
    }
```

Конструктор:

```
public School() { }
```
//Конструктор без параметров, создает экземпляр класса School.

Методы:

Добавляет студента в список студентов школы

```
public void AddStudent(Student student)
    {
        studentList.Add(student);
    }
```

Добавляет учителя в список учителей школы

```
    public void AddTeacher(Teacher teacher)
    {
        teacherList.Add(teacher);
    }
```

Возвращает строку, содержащую информацию о всех студентах школы.

```
    public string GetAllStudentsInfo()
    {
        string result = "";

        foreach (var student in studentList)
        {
            result += student.Info() + "\n";
        }
        return result;
    }
```

Возвращает строку, содержащую информацию о всех учителях школы.

```
    public string GetAllTeachersInfo()
    {
        string result = "";

        foreach (var teacher in teacherList)
        {
            result += teacher.Info() + "\n";
        }
        return result;
    }
```

Возвращает студента по указанному полному имени. Если студента с таким именем не существует, возвращает null

```
    public Student TryGetStudentByName(string fullname)
    {
        if (string.IsNullOrEmpty(fullname))
        {
            return null;
        }
        return studentList.Where(student => student.Fullname ==
fullname).FirstOrDefault();//вернет студента где имя совпадает с тем что мы ввели
    }
```

Возвращает учителя по указанному полному имени. Если учителя с таким именем не существует, возвращает null

```
    public Teacher TryGetTeacherByName(string fullname)
    {
        if (string.IsNullOrEmpty(fullname))
        {
            return null;
        }
        return teacherList.Where(teacher => teacher.Fullname == fullname).FirstOrDefault();}
```

class Programm

Методы:

Проверяет все ли параметры ввел пользователь и только тогда выполняется запрошенная команда

```
public static bool IsArrayMinSize(string[] inputArray, int minSize)
    {
        return inputArray.Length >= minSize;
    }
```

Частые повторения я вынесла в отдельные методы

```
    public static void LogErrorData()
    {
        Console.WriteLine("Вы ввели некорректные данные.\n");
    }

    public static void LogStudentNotFound()
    {
        Console.WriteLine("Такого студента не существует\n");
    }

    public static void LogTeacherNotFound()
    {
        Console.WriteLine("Такого учителя не существует\n");
    }


static void Main(string[] args)
    {
        School school = new School(); //
```

Вывожу список команд и инструкции к ним

```
        Console.ForegroundColor = ConsoleColor.Red;

        Console.WriteLine("Введите AddStudent Ф_И_О возраст рост номер_класса, чтобы
добавить ученика");
        Console.WriteLine("Введите StudentInfo Ф_И_О, чтобы посмотреть информацию о нужном
ученике");
        Console.WriteLine("Введите ShowAllStudents, чтобы посмотреть информацию о всех
учениках");
        Console.WriteLine("Введите ShowListOfSubject, чтобы посмотреть список предметов");
        Console.WriteLine("Введите ShowListOfWorkType, чтобы посмотреть список типов
работ");
        Console.WriteLine("Введите AddTeacher Ф_И_О возраст рост предмет(от 1 до 6), чтобы
добавить учителя");
```

```
        Console.WriteLine("Введите TeacherInfo Ф_И_О, чтобы посмотреть информацию о нужном
учителе");
        Console.WriteLine("Введите AddTeacherToStudent Ф_И_О учителя Ф_И_О ученика, чтобы
добавить добавить учителю ученика");
        Console.WriteLine("Введите ShowAllTeachers, чтобы посмотреть информацию о всех
учителях");
        Console.WriteLine("Введите AddGrade Ф_И_О ученика предмет(от 1 до 6) балл
тип_работы(от 1 до 5), чтобы добавить оценку ученику");
        Console.WriteLine("Введите ShowAllStudentMarks Ф_И_О ученика, чтобы посмотреть все
оценки ученика");
        Console.WriteLine("Введите ShowGradeBySubject Ф_И_О ученика предмет(от 1 до 6),
чтобы посмотреть оценки ученика по определенному предмету");
        Console.WriteLine("Введите ShowGradeByWorkType Ф_И_О ученика тип_работы(от 1 до 5),
чтобы посмотреть оценки ученика по типу работы");
        Console.WriteLine("Введите FinalGrade Ф_И_О ученика предмет(от 1 до 6), чтобы
посмотреть итоговую оценку ученика");
        Console.WriteLine("Введите AverageTeacherMark Ф_И_О учителя, чтобы найти среднюю
оценку из тех, что выставлял этот учитель");
        Console.WriteLine("Введите ShowAllTeacherMarks Ф_И_О учителя, чтобы посмотреть все
оценки, которые выставлял этот учитель");
        Console.WriteLine("Введите exit, чтобы выйти\n");

        Console.ResetColor();


while (true)
        {
            string userInput = Console.ReadLine(); //то что введет пользователь
            string[] inputArray = userInput.Split(' '); //делю по пробелам
            switch (inputArray[0])
            {
                case "AddStudent": //команда добавления студента
                    //Console.WriteLine("Напишите Ф_И_О возраст рост номер_класса");
                    if(!IsArrayMinSize(inputArray, 4))
                    {
                        LogErrorData();
                        break;
                    }

                    string name = inputArray[1];
```

С помощью TryParse я конвертирую строку в тип int и double, если это возможно сделать.

```
                    bool ageSuccess = int.TryParse(inputArray[2], out int age);
                    bool heightSuccess = double.TryParse(inputArray[3], out double height);
                    bool classNumberSuccess = int.TryParse(inputArray[4], out  int
classNumber);

                    if (ageSuccess && heightSuccess && classNumberSuccess)
                    {
                        Student student = new Student(name, age, height, classNumber);
                        school.AddStudent(student);
                        Console.WriteLine($"Вы добавили ученика\n");
                    }
                    else
                    {
                        LogErrorData();
                    }
```

```
                    break;
```

Метод для получения информации о студенте по его ФИО

```
case "StudentInfo":
                    // Console.WriteLine("Напишите Ф_И_О студента");
                    Student studentByName = school.TryGetStudentByName(inputArray[1]);
                    if(studentByName != null)
                    {
                        Console.WriteLine($"{studentByName.Info()}\n");
                    }
                    else
                    {
                        LogStudentNotFound();
                    }

                    break;
```

Метод, который показывает информацию о всех студентов

```
case "ShowAllStudents":
                    Console.WriteLine(school.GetAllStudentsInfo());
                    break;
```

Метод, добавляет учителя

```
                case "AddTeacher":
                    if (!IsArrayMinSize(inputArray, 4))
                    {
                        LogErrorData();
                        break;
                    }

                    string nameTeacher = inputArray[1];

                    bool ageTeacherSuccess = int.TryParse(inputArray[2], out int
ageTeacher);
                    bool heightTeacherSuccess = double.TryParse(inputArray[3], out double
heightTeacher);
                    bool subjectSuccess = int.TryParse(inputArray[4], out int subject);

                    if (ageTeacherSuccess && heightTeacherSuccess && subjectSuccess)
                    {
                        Teacher teacher = new Teacher(nameTeacher, ageTeacher,
heightTeacher, (SubjectType)subject);
                        school.AddTeacher(teacher);
                        Console.WriteLine($"Вы добавили учителя\n");
                    }
                    else
                    {
                        LogErrorData();
                    }
                    break;
```

Метод, который выводит информацию о нужном учителе

```
case "TeacherInfo":
                    Teacher teacherByName = school.TryGetTeacherByName(inputArray[1]);
                    if (teacherByName != null)
                    {
                        Console.WriteLine($"{teacherByName.Info()}\n");
                    }
                    else
                    {
                        LogTeacherNotFound();
                    }
                    break;
```

Метод, который добавляет учителю ученика, если найдется такой учитель и ученик

```
                case "AddTeacherToStudent":
                    //Console.WriteLine(Введите учитель ученик);
                    Teacher teacherToAddStudent = school.TryGetTeacherByName(inputArray[1]);
                    if (teacherToAddStudent == null)
                    {
                        LogTeacherNotFound();
                        break;
                    }

                    Student studentForTeacher = school.TryGetStudentByName(inputArray[2]);
                    if (studentForTeacher == null)
                    {
                        LogStudentNotFound();
                        break;
                    }

                    teacherToAddStudent.AddStudent(studentForTeacher);
                    Console.WriteLine("Вы успешно присвоили ученика учителю\n");
                    break;
```

Метод, который показывает информацию о всех учителях

```
case "ShowAllTeachers":
                    Console.WriteLine(school.GetAllTeachersInfo());
                    break;
```

Метод добавления оценки

```
                case "AddGrade":
                    //Console.WriteLine("Напишите Студент предмет бал тип_работы");

                    if (!IsArrayMinSize(inputArray, 4))
                    {
                        LogErrorData();
                        break;
                    }

                    Student studentForGrade = school.TryGetStudentByName(inputArray[1]);
                    if (studentForGrade == null)
                    {
                        LogStudentNotFound();
```

```
                    break;
                }

                bool studentSubjectSuccess = int.TryParse(inputArray[2], out int
studentSubject);
                bool scoreSuccess = int.TryParse(inputArray[3], out int score);
                bool studentSubjectTypeSuccess = int.TryParse(inputArray[4], out int
studentSubjectType);

                if(studentSubjectSuccess && scoreSuccess && studentSubjectTypeSuccess)
                {
                    Grade mark = new Grade((SubjectType)studentSubject, score,
(WorkType)studentSubjectType);

                    studentForGrade.AddGrade(mark);
                    Console.WriteLine("Вы добавили оценку ученику\n");
                }
                else
                {
                    LogErrorData();
                }

                break;
```

Метод, который показывает все оценки ученика

```
case "ShowAllStudentMarks":
                //команда студент
                Student studentMarks = school.TryGetStudentByName(inputArray[1]);
                if (studentMarks == null)
                {
                    LogStudentNotFound();
                    break;
                }
                Console.WriteLine(studentMarks.AllGradesInfo());
                break;
```

Метод который показывает оценки ученика по определенному предмету

```
            case "ShowGradeBySubject":
                //команда студент предмет
                if (!IsArrayMinSize(inputArray, 2))
                {
                    LogErrorData();
                    break;
                }

                Student studentMarksBySubject =
school.TryGetStudentByName(inputArray[1]);
                if (studentMarksBySubject == null)
                {
                    LogStudentNotFound();
                    break;
                }

                bool subjectTypeSuccess = int.TryParse(inputArray[2], out int
subjectType);
                if (subjectTypeSuccess)
                {
```

```
Console.WriteLine(studentMarksBySubject.GetGradesInfoBySubject((SubjectType)subjectType));
                    }
                    break;
```

Метод, который показывает оценки ученика по определенному типу работы

```
case "ShowGradeByWorkType":
                    if (!IsArrayMinSize(inputArray, 2))
                    {
                        LogErrorData();
                        break;
                    }

                    Student studentMarksByWork = school.TryGetStudentByName(inputArray[1]);
                    if (studentMarksByWork == null)
                    {
                        LogStudentNotFound();
                        break;
                    }

                    bool workTypeSuccess = int.TryParse(inputArray[2], out int workType);
                    if (workTypeSuccess)
                    {

Console.WriteLine(studentMarksByWork.GetGradesInfoByWork((WorkType)workType));
                    }
                    break;
```

Метод для подсчета финальной оценки ученика

```
                case "FinalGrade":
                    if (!IsArrayMinSize(inputArray, 2))
                    {
                        LogErrorData();
                        break;
                    }

                    Student studentFinalGrade = school.TryGetStudentByName(inputArray[1]);
                    if (studentFinalGrade == null)
                    {
                        LogStudentNotFound();
                        break;
                    }

                    bool subjectFinalSuccess = int.TryParse(inputArray[2], out int
subjectFinal);
                    if (subjectFinalSuccess)
                    {
                        Console.WriteLine($"Итоговая оценка:
{studentFinalGrade.FinalGrade((SubjectType)subjectFinal)}\n");
                    }
                    break;
```

Метод, который считает среднюю оценку из всех, что выставлял учитель

```
case "AverageTeacherMark":
                    Teacher teacherAllMarksAverage =
school.TryGetTeacherByName(inputArray[1]);
                    if (teacherAllMarksAverage == null)
                    {
                        LogTeacherNotFound();
                        break;
                    }
                    Console.WriteLine($"Средняя оценка выставленная учителем:
\n{teacherAllMarksAverage.GetAllMarksAverage()}\n");
                    break;
```

Метод, который показывает все оценки, которые выставлял учитель

```
                case "ShowAllTeacherMarks":
                    Teacher teacherAllMarks = school.TryGetTeacherByName(inputArray[1]);
                    if (teacherAllMarks == null)
                    {
                        LogTeacherNotFound();
                        break;
                    }
                    Console.WriteLine($"Все оценки выставленные учителем:
\n{teacherAllMarks.GetAllTeacherMarksInfo()}");
                    break;
```

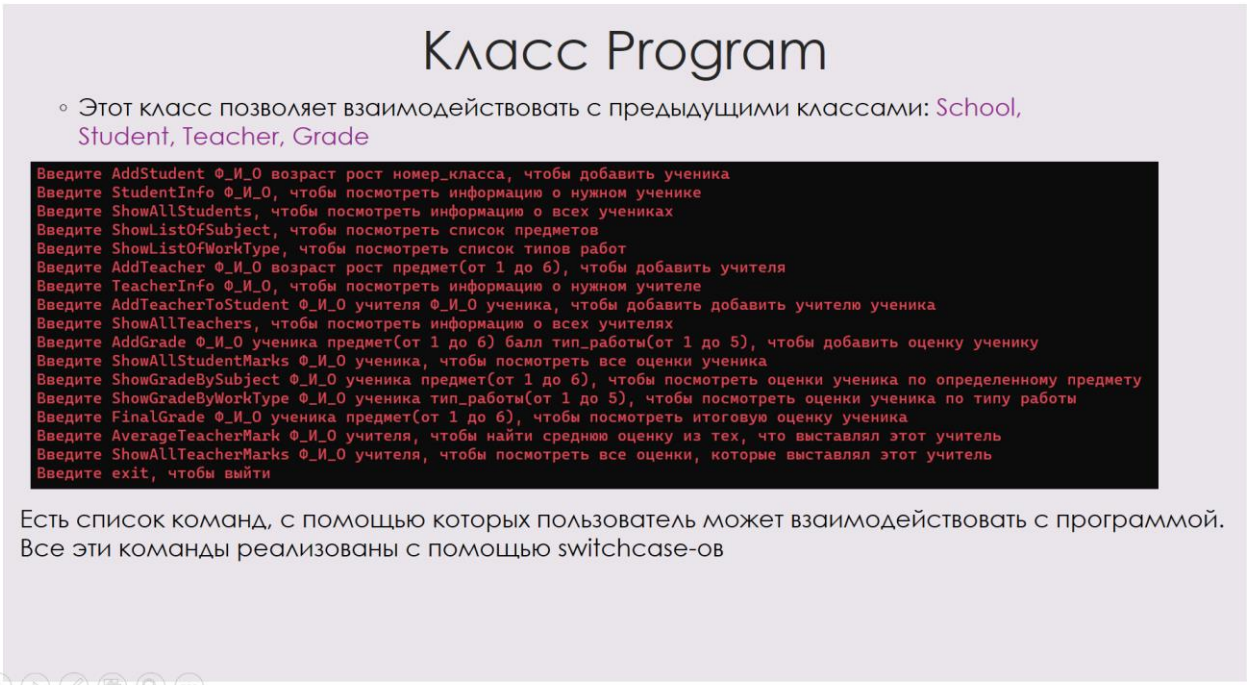
Методы для вывода списка доступных предметов и типов работ

```
case "ShowListOfSubject":
                    Console.WriteLine(" 1: Алхимия\n 2: Трансфигурация\n 3: Заклинания\n 4:
Гербология\n 5: Нумерология\n 6: Астрономия\n");
                    break;

case "ShowListOfWorkType":
                    Console.WriteLine(" 1: Домашняя работа\n 2: Контрольная\n 3: Тест\n 4:
Экзамен\n 5: Финальный экзамен\n");
                    break;
```

Метод для выхода из программы

```
                case "exit":
                    return;
            }

        }
    }
```

Пример работы программы:



Пример работы с интерактивной консолью

```
ShowListOfSubject
 1: Алхимия
 2: Трансфигурация
 3: Заклинания
 4: Гербология
 5: Нумерология
 6: Астрономия

ShowListOfWorkType
 1: Домашняя работа
 2: Контрольная
 3: Тест
 4: Экзамен
 5: Финальный экзамен

AddTeacher Северус_Снейп 38 176 1
Вы добавили учителя

AddTeacher Минерва_Макгонагалл 55 168 2
Вы добавили учителя
```

```
AddStudent Гермиона_Грейнджер 11 165 1
Вы добавили ученика

AddStudent Оливер_Вуд 15 184 5
Вы добавили ученика

AddStudent Седрик_Диггори 14 180,5 4
Вы добавили ученика

StudentInfo Гермиона_Грейнджер
ФИО: Гермиона Грейнджер  Возраст: 11  Рост: 165  Класс: 1

ShowAllStudents
ФИО: Гермиона Грейнджер  Возраст: 11  Рост: 165  Класс: 1
ФИО: Оливер Вуд  Возраст: 15  Рост: 184  Класс: 5
ФИО: Седрик Диггори  Возраст: 14  Рост: 180,5  Класс: 4
```

Мы можем написать команду и параметры к ней, если мы введем все нужные параметры, то команда обработается успешно, если нет, в консоль выведется ошибка.

```
AddGrade Седрик_Диггори 1 10 1
Вы добавили оценку ученику

ShowAllStudentMarks Гермиона_Грейнджер
Предмет: Alchemy, Оценка 12, Вид работы: Exam
Предмет: Alchemy, Оценка 12, Вид работы: FinalExam

ShowGradeBySubject Гермиона_Грейнжер 1
Такого студента не существует

ShowGradeBySubject Гермиона_Грейнджер 1
Предмет: Alchemy, Оценка 12, Вид работы: Exam
Предмет: Alchemy, Оценка 12, Вид работы: FinalExam

ShowGradeByWorkType Седрик_Диггори 1
Предмет: Alchemy, Оценка 10, Вид работы: Homework

FinalGrade Гермиона_Грейнджер 1
Итоговая оценка: 6

AverageTeacherMark Северус_Снейп
Средняя оценка выставленная учителем:
11,333333333333334

ShowAllTeacherMarks Северус_Снейп
Все оценки выставленные учителем:
Предмет: Alchemy, Оценка 12, Вид работы: Exam
Предмет: Alchemy, Оценка 12, Вид работы: FinalExam
```

Еще несколько команд

- В конце пишем exit и программа завершится.

```
exit

C:\Универ\Программирование\Курсовая\School\Scho
ol\bin\Debug\netcoreapp3.1\Exam.exe (процесс 48
12) завершил работу с кодом 0.
```

Выводы:

Была создана программа, которая реализует школу. В школу можно добавлять студентов и учителей, так же ставить оценки ученикам и присваивать учеников учителю.

Были созданы такие команды для взаимодействия с программой:

```
Введите AddStudent Ф_И_О возраст рост номер_класса, чтобы добавить ученика
Введите StudentInfo Ф_И_О, чтобы посмотреть информацию о нужном ученике
Введите ShowAllStudents, чтобы посмотреть информацию о всех учениках
Введите ShowListOfSubject, чтобы посмотреть список предметов
Введите ShowListOfWorkType, чтобы посмотреть список типов работ
Введите AddTeacher Ф_И_О возраст рост предмет(от 1 до 6), чтобы добавить учителя
Введите TeacherInfo Ф_И_О, чтобы посмотреть информацию о нужном учителе
Введите AddTeacherToStudent Ф_И_О учителя Ф_И_О ученика, чтобы добавить добавить учителю ученика
Введите ShowAllTeachers, чтобы посмотреть информацию о всех учителях
Введите AddGrade Ф_И_О ученика предмет(от 1 до 6) балл тип_работы(от 1 до 5), чтобы добавить оценку ученику
Введите ShowAllStudentMarks Ф_И_О ученика, чтобы посмотреть все оценки ученика
Введите ShowGradeBySubject Ф_И_О ученика предмет(от 1 до 6), чтобы посмотреть оценки ученика по определенному предмету
Введите ShowGradeByWorkType Ф_И_О ученика тип_работы(от 1 до 5), чтобы посмотреть оценки ученика по типу работы
Введите FinalGrade Ф_И_О ученика предмет(от 1 до 6), чтобы посмотреть итоговую оценку ученика
Введите AverageTeacherMark Ф_И_О учителя, чтобы найти среднюю оценку из тех, что выставлял этот учитель
Введите ShowAllTeacherMarks Ф_И_О учителя, чтобы посмотреть все оценки, которые выставлял этот учитель
Введите exit, чтобы выйти
```