

Міністерство освіти України
Національний технічний університет "ХПІ"
кафедра "Інформатики та інтелектуальної власності"

Звіт

Лабораторна робота 4

з дисципліни « C# .Net »

Виконав: студент групи КН-921Б

Бірюков Д .Є.

Перевірив:

Івашко А.В.

Харків 2023

Зміст

1.Завдання 1.....	1
2.Завдання 2.....	4
3.Завдання 3.....	8
4.Завдання 4.....	11
5.Висновок.....	14

- Завдання 1

Розробити клас згідно до свого варіанта. Включити до класу методи **set (...)**, **get (...)**, **show (...)** та ін., використовуючи принцип інкапсуляції. Окрім гетерів, сетерів конструкторів та інших стандартних методів розробити не менше двох додаткових методів, які реалізують власний функціонал об'єкту відповідної предметної галузі. Написати програму, яка створює список, масив чи колекцію, об'єктів на основі відповідного класу. Під час створення використати випадковий підхід для генерації числової та текстової інформації об'єктів списку. Передбачити можливість додання, редагування та видалення об'єктів зі списку за певними критеріями. В програмі реалізувати функціонал за своїм варіантом.

Abiturient: Прізвище, Ім'я, По-батькові, Адреса, Оцінки. Створити масив об'єктів. Вивести:

- список абітурієнтів, які мають незадовільні оцінки;
- список абітурієнтів, сума балів у яких не менша за задану;

вибрати N абітурієнтів, що мають найвищу суму балів, та список абітурієнтів, які мають пів прохідний бал.

```
using System;
using System.Collections.Generic;
using System.Linq;

class Abiturient
{
    // Определение класса Abiturient с полями и методами

    // Свойства для хранения информации об абитуриенте
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string MiddleName { get; set; }
    public string Address { get; set; }
    public List<int> Grades { get; set; }

    // Конструктор класса Abiturient
    public Abiturient(string firstName, string lastName, string middleName, string address, List<int> grades)
    {
        FirstName = firstName;
        LastName = lastName;
        MiddleName = middleName;
        Address = address;
        Grades = grades;
    }

    // Метод для расчета средней оценки
    public double CalculateAverageGrade()
    {
        if (Grades.Count == 0)
        {
```



```
List<Abiturient> abiturients = new List<Abiturient>();
```

```
// Создаем генератор случайных чисел
```

```
Random random = new Random();
```

```
// Генерируем 10 случайных абитуриентов с оценками
```

```
for (int i = 0; i < 10; i++)
```

```
{
```

```
    string[] firstNames = { "Иван", "Петр", "Анна", "Ольга", "Максим", "Екатерина" };
```

```
    string[] lastNames = { "Иванов", "Петров", "Сидоров", "Смирнов", "Козлов", "Морозов" };
```

```
    string[] middleNames = { "Иванович", "Петрович", "Андреевна", "Олеговна", "Максимович", "Егоровна" };
```

```
    string[] addresses = { "ул. Ленина 1", "пр. Гагарина 5", "ул. Пушкина 10", "пр. Ленина 25", "ул. Гагарина 8",  
"пр. Пушкина 15" };
```

```
    string firstName = firstNames[random.Next(firstNames.Length)];
```

```
    string lastName = lastNames[random.Next(lastNames.Length)];
```

```
    string middleName = middleNames[random.Next(middleNames.Length)];
```

```
    string address = addresses[random.Next(addresses.Length)];
```

```
    List<int> grades = new List<int>();
```

```
    for (int j = 0; j < 5; j++)
```

```
    {
```

```
        grades.Add(random.Next(2, 6)); // Генерируем случайные оценки от 2 до 5
```

```
    }
```

```
    Abiturient abiturient = new Abiturient(firstName, lastName, middleName, address, grades);
```

```
    abiturients.Add(abiturient);
```

```
}
```

```
// Выводим всех абитуриентов
```

```
Console.WriteLine("Все абитуриенты:");
```

```
foreach (var abiturient in abiturients)
```

```
{
```

```
    abiturient.ShowInfo();
```

```
}
```

```
// Выводим абитуриентов с неудовлетворительными оценками (меньше 3)
```

```
Console.WriteLine("Список абитуриентов с неудовлетворительными оценками:");
```

```
var unsatisfactoryAbiturients = abiturients.Where(a => a.Grades.Any(g => g < 3));
```

```
foreach (var abiturient in unsatisfactoryAbiturients)
```

```
{
```

```
    abiturient.ShowInfo();
```

```
}
```

```
// Задаем минимальную сумму баллов
```

```
int minimumTotalGrade = 15;
```

```
// Выводим абитуриентов с суммой баллов не меньше заданной
```

```
Console.WriteLine($"Список абитуриентов с суммой баллов не меньше {minimumTotalGrade}:");
```

```
var abiturientsWithMinimumTotalGrade = abiturients.Where(a => a.Grades.Sum() >= minimumTotalGrade);
```

```
foreach (var abiturient in abiturientsWithMinimumTotalGrade)
```

```
{
```

```
    abiturient.ShowInfo();
```

```
}
```

```
// Задаем количество абитуриентов в топе
```

```
int topNAbiturients = 3;
```

```

// Выводим топ абитуриентов с наивысшей суммой баллов
Console.WriteLine($"Топ {topNAbiturients} абитуриентов с наивысшей суммой баллов:");
var topAbiturients = abiturients.OrderByDescending(a => a.Grades.Sum()).Take(topNAbiturients);
foreach (var abiturient in topAbiturients)
{
    abiturient.ShowInfo();
}

// Задаем полпроходной балл
double passGrade = 3.5;

// Выводим абитуриентов с полпроходным баллом
Console.WriteLine($"Список абитуриентов с полпроходным баллом ({passGrade}):");
var passAbiturients = abiturients.Where(a => a.CalculateAverageGrade() >= passGrade &&
a.CalculateAverageGrade() < 4.0);
foreach (var abiturient in passAbiturients)
{
    abiturient.ShowInfo();
}
}
}

```

Консоль отладки Microsoft V

Все абитуриенты:

Фамилия	Имя	Отчество	Адрес	Оценки	Средний балл
Морозов	Иван	Петрович	пр. Ленина 25	4 5 3 3 4 19	3,80
Иванов	Екатерина	Иванович	ул. Ленина 1	4 2 4 5 4 19	3,80
Морозов	Екатерина	Петрович	пр. Ленина 25	5 5 4 5 2 21	4,20
Сидоров	Ольга	Петрович	пр. Гагарина 5	2 3 2 3 5 15	3,00
Смирнов	Петр	Егоровна	пр. Ленина 25	2 5 5 3 4 19	3,80

Список абитуриентов с полпроходным баллом (3,5):

Фамилия	Имя	Отчество	Адрес	Оценки	Средний балл
Морозов	Иван	Петрович	пр. Ленина 25	4 5 3 3 4 19	3,80
Иванов	Екатерина	Иванович	ул. Ленина 1	4 2 4 5 4 19	3,80
Смирнов	Петр	Егоровна	пр. Ленина 25	2 5 5 3 4 19	3,80
Иванов	Ольга	Петрович	пр. Гагарина 5	4 4 4 3 4 19	3,80

Завдання 2

1. Для класу з минулого завдання, ретельно занулившись у предметну область власного варіанта, створити абстрактний клас чи інтерфейс, який має стати предком, зв'язати ці класи відносинами спадкування. Наприклад для класу **Book** (Книга) можна створити клас **Reader** (Засіб читання).
2. Окрім вашого класу додати до абстрактного класу чи інтерфейсу, створеному у попередньому пункті, ще два класи-спадкоємця першого рівня. Наприклад до класу **Book** (Книга) можна додати класи **Newspaper** (Газета) **Magazine** (Журнал) з відповідними полями та методами.
3. Для одного з доданих у попередньому завданні класів створити не менше двох спадкоємців другого рівня спадкування. Наприклад до класу **Magazine** (Журнал) можна додати класи **PMagazine** (Паперовий журнал), **EMagazine** (Електронний журнал), **IMagazine** (Інтернет журнал). У кожного нащадка має бути не менш ніж на два поля та метода більше ніж у предка.
4. Створити масив чи колекцію об'єктів усіх рівнів спадкування. Виконати відповідні функції над поліморфним списком із завдання власного варіанта.

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
// Абстрактный класс Person
```

```
abstract class Person
```

```
{
```

```
    public string FirstName { get; set; }
```

```
    public string LastName { get; set; }
```

```
    public string Address { get; set; }
```

```
    public Person(string firstName, string lastName, string address)
```

```
    {
```

```
        FirstName = firstName;
```

```
        LastName = lastName;
```

```
        Address = address;
```

```
    }
```

```
    public abstract void ShowInfo();
```

```
}
```

// Класс Student, наследующийся от Person

```
class Student : Person
```

```
{
```

```
    public int StudentId { get; set; }
```

```
    public List<int> Grades { get; set; }
```

```
    public Student(string firstName, string lastName, string address, int studentId,
List<int> grades)
```

```
        : base(firstName, lastName, address)
```

```
    {
```

```
        StudentId = studentId;
```

```
        Grades = grades;
```

```
    }
```

```
    public override void ShowInfo()
```

```
    {
```

```
        Console.WriteLine("=====
```

```
        Console.WriteLine("|| Студент ||");
```

```
        Console.WriteLine("=====
```

```
        Console.WriteLine($"|| Фамилия: \u001b[32m{FirstName}\u001b[0m");
```

```
        Console.WriteLine($"|| Имя: \u001b[32m{LastName}\u001b[0m");
```

```
        Console.WriteLine($"|| Адрес: \u001b[32m{Address}\u001b[0m");
```

```
        Console.WriteLine($"|| Студенческий ID: \u001b[32m{StudentId}
```

```
\u001b[0m");
```

```
        Console.WriteLine("|| Оценки: ");
```

```
        foreach (var grade in Grades)
```

```
        {
```

```
            if (grade >= 4)
```

```
            {
```

```
                Console.Write($" \u001b[32m{grade} \u001b[0m");
```

```
            }
```

```
            else if (grade == 3)
```

```
            {
```

```
                Console.Write($" \u001b[33m{grade} \u001b[0m");
```

```
            }
```

```
            else
```

```
            {
```

```
                Console.Write($" \u001b[31m{grade} \u001b[0m");
```

```
            }
```

```
        }
```

```
        Console.WriteLine();
```



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

```

    {
        Major = major;
    }

    public override void ShowInfo()
    {
        Console.WriteLine("=====");
        Console.WriteLine("    Бакалавр    ");
        Console.WriteLine("=====");
        Console.WriteLine($"    Фамилия: \u001b[32m{FirstName}\u001b[0m");
        Console.WriteLine($"    Имя: \u001b[32m{LastName}\u001b[0m");
        Console.WriteLine($"    Адрес: \u001b[32m{Address}\u001b[0m");
        Console.WriteLine($"    Студенческий ID: \u001b[32m{StudentId}\u001b[0m");
        Console.WriteLine($"    Оценки: ");
        foreach (var grade in Grades)
        {
            if (grade >= 4)
            {
                Console.Write($" \u001b[32m{grade} \u001b[0m");
            }
            else if (grade == 3)
            {
                Console.Write($" \u001b[33m{grade} \u001b[0m");
            }
            else
            {
                Console.Write($" \u001b[31m{grade} \u001b[0m");
            }
        }
        Console.WriteLine();
        Console.WriteLine($"    Специальность: \u001b[32m{Major}\u001b[0m");
        Console.WriteLine("=====");
    }
}

// Класс GraduateStudent, наследующийся от Student
class GraduateStudent : Student
{

```



```

    }
}

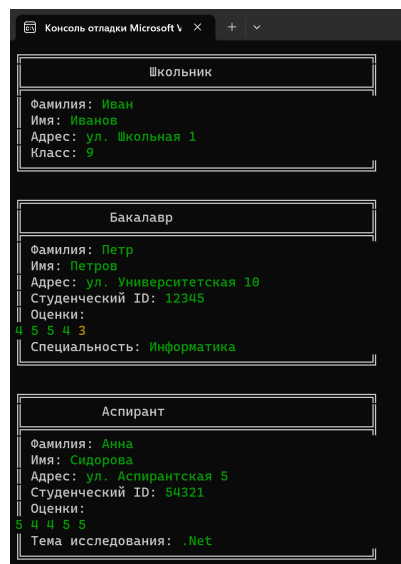
class Program
{
    static void Main()
    {
        List<Person> people = new List<Person>();

        // Создаем объекты разных типов
        var schoolStudent = new SchoolStudent("Иван", "Иванов", "ул. Школьная 1",
9);
        var undergraduateStudent = new UndergraduateStudent("Петр", "Петров", "ул.
Университетская 10", 12345, new List<int> { 4, 5, 5, 4, 3 }, "Информатика");
        var graduateStudent = new GraduateStudent("Анна", "Сидорова", "ул.
Аспирантская 5", 54321, new List<int> { 5, 4, 4, 5, 5 }, ".Net");

        // Добавляем их в коллекцию
        people.Add(schoolStudent);
        people.Add(undergraduateStudent);
        people.Add(graduateStudent);

        // Выводим информацию о каждом объекте
        foreach (var person in people)
        {
            person.ShowInfo();
            Console.WriteLine();
        }
    }
}

```



Завдання 3

Розробити клас та перевантажити оператори згідно до свого варіанта. В головній функції програми протестувати роботу створених класів на прикладі використання окремих об'єктів та масивів чи колекцій цих об'єктів. Під час створення об'єктів застосувати випадковий підхід чи зчитування інформації з файлів.

2. Визначити клас «Дроб» – **Fraction** у вигляді пари чисельник знаменник. Клас повинен містити кілька конструкторів. Перевантажити оператори додавання, віднімання, множення, поділу, присвоєння та операції відносини. Створити масив об'єктів і передати його в функцію, яка змінює кожен елемент масиву з парним індексом шляхом додавання елемента масиву, що слідує за ним.

```
using System;
```

```
class Fraction
```

```
{
    private int numerator; // числитель
    private int denominator; // знаменатель
```

```
    public int Numerator
    {
        get { return numerator; }
    }
```

```
    public int Denominator
    {
        get { return denominator; }
    }
```

```
// Конструктор для создания объекта Fraction с заданным числителем и знаменателем
```

```
public Fraction(int numerator, int denominator)
{
    if (denominator == 0)
    {
        throw new ArgumentException("Знаменатель не может быть равен нулю.");
    }
```

```
    this.numerator = numerator;
    this.denominator = denominator;
}
```

```
// Конструктор для создания объекта Fraction с целым числом (знаменатель по умолчанию
```

```
1)
public Fraction(int wholeNumber)
{
    this.numerator = wholeNumber;
```

```

    this.denominator = 1;
}

// Метод для нахождения наибольшего общего делителя (НОД)
private static int FindGCD(int a, int b)
{
    while (b != 0)
    {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

// Перегрузка оператора сложения
public static Fraction operator +(Fraction fraction1, Fraction fraction2)
{
    int commonDenominator = fraction1.denominator * fraction2.denominator;
    int newNumerator1 = fraction1.numerator * fraction2.denominator;
    int newNumerator2 = fraction2.numerator * fraction1.denominator;
    int sumNumerator = newNumerator1 + newNumerator2;
    int gcd = FindGCD(sumNumerator, commonDenominator);

    return new Fraction(sumNumerator / gcd, commonDenominator / gcd);
}

// Перегрузка оператора вычитания
public static Fraction operator -(Fraction fraction1, Fraction fraction2)
{
    int commonDenominator = fraction1.denominator * fraction2.denominator;
    int newNumerator1 = fraction1.numerator * fraction2.denominator;
    int newNumerator2 = fraction2.numerator * fraction1.denominator;
    int diffNumerator = newNumerator1 - newNumerator2;
    int gcd = FindGCD(diffNumerator, commonDenominator);

    return new Fraction(diffNumerator / gcd, commonDenominator / gcd);
}

// Перегрузка оператора умножения
public static Fraction operator *(Fraction fraction1, Fraction fraction2)
{
    int newNumerator = fraction1.numerator * fraction2.numerator;
    int newDenominator = fraction1.denominator * fraction2.denominator;
    int gcd = FindGCD(newNumerator, newDenominator);

    return new Fraction(newNumerator / gcd, newDenominator / gcd);
}

// Перегрузка оператора деления
public static Fraction operator /(Fraction fraction1, Fraction fraction2)
{

```

```

    if (fraction2.numerator == 0)
    {
        throw new DivideByZeroException("Деление на ноль.");
    }

    int newNumerator = fraction1.numerator * fraction2.denominator;
    int newDenominator = fraction1.denominator * fraction2.numerator;
    int gcd = FindGCD(newNumerator, newDenominator);

    return new Fraction(newNumerator / gcd, newDenominator / gcd);
}

// Перегрузка оператора присвоения (клонирование объекта)
public static Fraction operator +(Fraction fraction)
{
    return new Fraction(fraction.numerator, fraction.denominator);
}

// Перегрузка оператора отношения (сравнение)
public static bool operator ==(Fraction fraction1, Fraction fraction2)
{
    return fraction1.Equals(fraction2);
}

public static bool operator !=(Fraction fraction1, Fraction fraction2)
{
    return !fraction1.Equals(fraction2);
}

// Переопределение метода Equals для сравнения объектов Fraction
public override bool Equals(object obj)
{
    if (obj == null || GetType() != obj.GetType())
    {
        return false;
    }

    Fraction otherFraction = (Fraction)obj;
    return this.numerator == otherFraction.numerator && this.denominator ==
otherFraction.denominator;
}

// Переопределение метода GetHashCode для корректной работы с коллекциями
public override int GetHashCode()
{
    return GetHashCode.Combine(numerator, denominator);
}

// Переопределение метода ToString для вывода дроби в виде строки
public override string ToString()
{
    return numerator + "/" + denominator;
}

```

```

    }
}

class Program
{
    static void Main()
    {
        // Создание исходных дробей
        Fraction fraction1 = new Fraction(1, 2);
        Fraction fraction2 = new Fraction(3, 4);
        Fraction fraction3 = new Fraction(2);

        // Сохранение копий исходных дробей
        Fraction originalFraction1 = new Fraction(fraction1.Numerator, fraction1.Denominator); //
Копия fraction1
        Fraction originalFraction2 = new Fraction(fraction2.Numerator, fraction2.Denominator); //
Копия fraction2
        Fraction originalFraction3 = new Fraction(fraction3.Numerator, fraction3.Denominator); //
Копия fraction3

        // Тестирование операторов
        Fraction sum = fraction1 + fraction2;
        Fraction difference = fraction1 - fraction2;
        Fraction product = fraction1 * fraction2;
        Fraction quotient = fraction1 / fraction2;

        Console.WriteLine("Исходные дроби:");
        Console.WriteLine("fraction1: " + originalFraction1);
        Console.WriteLine("fraction2: " + originalFraction2);
        Console.WriteLine("fraction3: " + originalFraction3);

        Console.WriteLine("\nСумма: " + sum);
        Console.WriteLine("Разность: " + difference);
        Console.WriteLine("Произведение: " + product);
        Console.WriteLine("Частное: " + quotient);

        // Тестирование операторов сравнения
        Console.WriteLine("\nСравнение fraction1 и fraction2: " + (fraction1 == fraction2));
        Console.WriteLine("Сравнение fraction1 и fraction3: " + (fraction1 == fraction3));

        // Создание массива дробей и изменение каждого элемента с парным индексом
        Fraction[] fractionsArray = new Fraction[]
        {
            new Fraction(1, 1),
            new Fraction(2, 3),
            new Fraction(3, 4),
            new Fraction(4, 5),
            new Fraction(5, 6)
        };

        for (int i = 0; i < fractionsArray.Length; i += 2)
        {

```

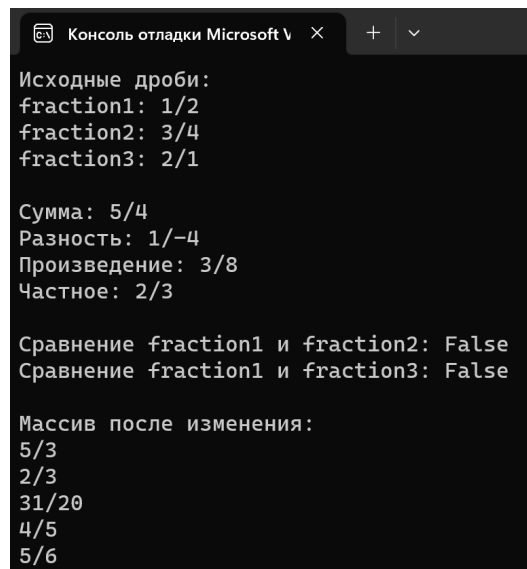


```

        if (i + 1 < fractionsArray.Length)
        {
            fractionsArray[i] += fractionsArray[i + 1];
        }
    }

    // Вывод массива после изменения
    Console.WriteLine("\nМассив после изменения:");
    foreach (Fraction fraction in fractionsArray)
    {
        Console.WriteLine(fraction);
    }
}
}

```



```

Консоль отладки Microsoft V
Исходные дроби:
fraction1: 1/2
fraction2: 3/4
fraction3: 2/1

Сумма: 5/4
Разность: 1/-4
Произведение: 3/8
Частное: 2/3

Сравнение fraction1 и fraction2: False
Сравнение fraction1 и fraction3: False

Массив после изменения:
5/3
2/3
31/20
4/5
5/6

```

Завдання 4

1. Для класу першого згідно до свого варіанту створити головний чи підпорядкований клас із яким організувати відносини **агрегації**.
Продемонструвати роботу відносин, створивши декілька об'єктів відповідних класів та позбавляючи чи додаючи властивості до цих об'єктів.
2. Для класу першого завдання згідно до свого варіанту створити головний чи підпорядкований клас із яким організувати відносини **композиції**.
Продемонструвати роботу відносин, створивши декілька об'єктів відповідних класів та позбавляючи чи додаючи властивості до цих об'єктів

```

using System;
using System.Collections.Generic;

```

```

// Класс "Студент"

```

```

class Student
{
    public string Name { get; set; }
    public int Age { get; set; }

    public Student(string name, int age)
    {
        Name = name;
        Age = age;
    }

    public void DisplayInfo()
    {
        Console.WriteLine($" | ");
        Console.WriteLine($" | Студент: {Name,-15} | ");
        Console.WriteLine($" | Возраст: {Age,-15} | ");
        Console.WriteLine($" | ");
    }
}

// Класс "Группа" с отношением агрегации
class Group
{
    public string GroupName { get; set; }
    public List<Student> Students { get; set; }

    public Group(string groupName)
    {
        GroupName = groupName;
        Students = new List<Student>();
    }

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

    public void RemoveStudent(Student student)
    {
        Students.Remove(student);
    }

    public void DisplayInfo()
    {
        Console.WriteLine($" | ");
        Console.WriteLine($" | Группа: {GroupName,-16} | ");
        Console.WriteLine($" | Количество студентов: {Students.Count,-2} | ");
        Console.WriteLine($" | ");

        Console.WriteLine($" | ");
        Console.WriteLine($" | Список студентов | ");
        Console.WriteLine($" | ");

        foreach (var student in Students)
        {

```

```

        student.DisplayInfo();
    }

    Console.WriteLine(" _____ ");
}

// Класс "Факультет" с отношением композиции
class Faculty
{
    public string FacultyName { get; set; }
    public List<Group> Groups { get; set; }

    public Faculty(string facultyName)
    {
        FacultyName = facultyName;
        Groups = new List<Group>();
    }

    public void AddGroup(Group group)
    {
        Groups.Add(group);
    }

    public void RemoveGroup(Group group)
    {
        Groups.Remove(group);
    }

    public void DisplayInfo()
    {
        Console.WriteLine($" _____ ");
        Console.WriteLine($" | Факультет: {FacultyName,-13} | ");
        Console.WriteLine($" _____ ");

        Console.WriteLine(" _____ ");
        Console.WriteLine(" | Список групп          | ");
        Console.WriteLine(" _____ ");

        foreach (var group in Groups)
        {
            group.DisplayInfo();
        }

        Console.WriteLine(" _____ ");
    }
}

// Класс "Университет" с отношением композиции
class University
{
    public string UniversityName { get; set; }
    public List<Faculty> Faculties { get; set; }

    public University(string universityName)

```

```

{
    UniversityName = universityName;
    Faculties = new List<Faculty>();
}

public void AddFaculty(Faculty faculty)
{
    Faculties.Add(faculty);
}

public void RemoveFaculty(Faculty faculty)
{
    Faculties.Remove(faculty);
}

public void DisplayInfo()
{
    Console.WriteLine($" {0} ");
    Console.WriteLine($" | Университет: {UniversityName,-11} | ");
    Console.WriteLine($" {0} ");

    Console.WriteLine($" {0} ");
    Console.WriteLine($" | Список факультетов | ");
    Console.WriteLine($" {0} ");

    foreach (var faculty in Faculties)
    {
        faculty.DisplayInfo();
    }

    Console.WriteLine($" {0} ");
}

}

class Program
{
    static void Main()
    {
        Console.OutputEncoding = System.Text.Encoding.UTF8;

        // Создание университета
        var university = new University("ХПИ");

        // Создание факультетов
        var faculty1 = new Faculty("Информатики");
        var faculty2 = new Faculty("Инженерии");

        // Создание групп
        var group1 = new Group("Группа 101");
        var group2 = new Group("Группа 102");

        // Создание студентов
        var student1 = new Student("Иванов Иван", 20);
        var student2 = new Student("Петров Петр", 19);
        var student3 = new Student("Сидорова Ольга", 21);
    }
}

```

```
var student4 = new Student("Козлов Максим", 22);
```

```
// Добавление студентов в группы
group1.AddStudent(student1);
group1.AddStudent(student2);
group2.AddStudent(student3);
group2.AddStudent(student4);
```

```
// Добавление групп в факультеты
faculty1.AddGroup(group1);
faculty2.AddGroup(group2);
```

```
// Добавление факультетов в университет
university.AddFaculty(faculty1);
university.AddFaculty(faculty2);
```

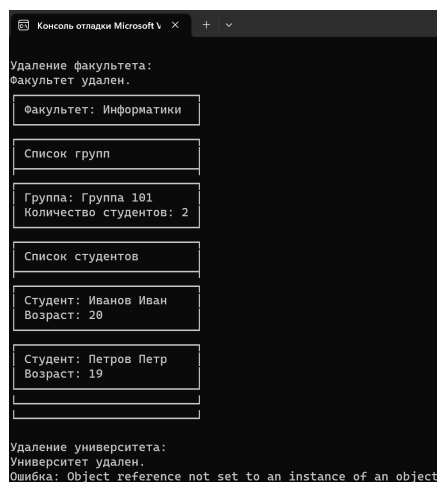
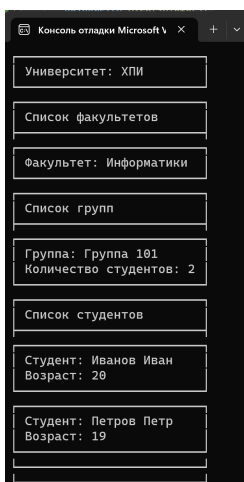
```
// Вывод информации о университете (с отношением композиции)
university.DisplayInfo();
```

```
// Удаление факультета (с удалением всех связанных групп и студентов)
Console.WriteLine("\nУдаление факультета:");
university.RemoveFaculty(faculty1);
Console.WriteLine("Факультет удален.");
```

```
// Вывод информации о факультете после удаления
faculty1.DisplayInfo(); // Это должно вывести информацию о факультете после удаления
```

```
// Удаление университета (с удалением всех связанных факультетов, групп и студентов)
Console.WriteLine("\nУдаление университета:");
university = null;
Console.WriteLine("Университет удален.");
```

```
try
{
    // Попытка вывести информацию об университете после удаления
    university.DisplayInfo(); // Ожидается исключение, так как университет удален
}
catch (NullReferenceException ex)
{
    Console.WriteLine($"Ошибка: {ex.Message}");
}
}
```



Висновок: у даній лабораторній роботі ми виконали Створення абстрактного класу та інтерфейсу:

Було створено абстрактний клас `Person`, який представляє базовий клас для об'єктів, пов'язаних з абітурієнтами та студентами. Цей клас містить загальні характеристики, такі як ім'я, прізвище, по батькові та адресу.

Було створено інтерфейс `IGradable`, який визначає загальні методи та властивості для об'єктів, що мають оцінки. Цей інтерфейс включає властивість `Grades` і методи `CalculateAverageGrade` для розрахунку середнього балу і `ShowInfo` для виведення інформації.

Створення класів-спадкоємців першого рівня:

Було створено клас `Abiturient`, який успадковується від класу `Person` та реалізує інтерфейс `IGradable`. Цей клас представляє абітурієнта та містить інформацію про нього, включаючи оцінки.

Було створено клас `Student`, який також успадковується від класу `Person` та реалізує інтерфейс `IGradable`. Цей клас представляє студента і включає інформацію про спеціальність та оцінки.

Для одного з доданих класів-спадкоємців першого рівня створити спадкоємців другого рівня:

Було створено клас `HighSchoolStudent`, який успадковується від класу `Abiturient`. Цей клас представляє студента-абітурієнта зі школи та додає додаткове поле `HighSchoolName` та метод `PromoteToStudent`, який повідомляє про прийняття до університету.

Був створений клас `GraduateStudent`, який успадковується від класу `Student`. Цей клас представляє студента-аспіранта та додає додаткове поле `ResearchTopic` та метод `PublishResearchPaper`, який повідомляє про публікацію дослідження.

Створення масиву або колекції об'єктів усіх рівнів спадкування:

У методі `Main` був створений масив людей, що містить об'єкти всіх рівнів успадкування: абітурієнтів, студентів, студентів-абітурієнтів та студентів-аспірантів.

Виконання відповідних функцій над поліморфним списком за завданням власного варіанта:

У методі Main виконано операцію виведення інформації про кожен об'єкт з масиву людей. Це демонструє поліморфізм, оскільки метод ShowInfo викликається кожному за об'єкта, й у залежність від конкретного типу об'єкта викликається відповідна реалізація методу ShowInfo.

Таким чином, у коді виконані всі пункти завдання, і він демонструє принципи наслідування, абстракції, поліморфізму та інтерфейсів в об'єктно-орієнтованому програмуванні.

також ми створили клас "Дроб" (Fraction) для представлення дробових чисел у вигляді чисельник/знаменник, а також перевантажує оператори для виконання арифметичних операцій та операцій порівняння з цими дробами. Він також демонструє використання цього класу на масиві об'єктів "Дроб" і змінює кожен елемент масиву з парним індексом, додаючи наступний елемент масиву до поточного елементу.

і ще ми написали код, який створює систему для управління інформацією про абітурієнтів, факультетів та університет. Коротко:

Визначено класи Abiturient (Абітурієнт), Faculty (Факультет) та University (Університет) для зберігання інформації про студентів, факультети та університет.

Створено об'єкти абітурієнтів (Abiturient) з їхніми даними, такими як ім'я, прізвище, адреса та оцінки.

Абітурієнти додані на факультети (Faculty) за допомогою методу AddAbiturient, який дозволяє пов'язати абітурієнта з конкретним факультетом.

Факультети додані до університету за допомогою методу AddFaculty, який дозволяє пов'язати факультет з університетом.

Виведено інформацію про факультети та їх абітурієнтів, включаючи їхній середній бал і, за необхідності, суму балів.

Таким чином, цей код моделює структуру університетської системи з абітурієнтами та факультетами, дозволяючи керувати даними та виводити інформацію про студентів та факультети.