Министерство образования Республики Беларусь

Оршанский колледж ВГУ имени П.М. Машерова

Отчет

По модулю №2

«Объектно-ориентированное программирование (ООП)»

по учебной практике по программированию

Выполнил учащийся Войтеховская К.В.

группа 3ПОИС23 29.09.2025 г.

Проверил Алейников М.А.

30.09.2025 г.

Орша, 2025

**Ход работы.**

**Выполнение практических заданий.**

**Выполнение задания 1.** Создайте класс Person, представляющий человека. У этого класса должны быть поля для хранения имени, возраста и адреса. Добавьте методы для установки и получения значений этих полей. Затем создайте объекты этого класса и выведите информацию о них.

Для выполнения задания был создан класс Person который содержит данные о человеке. Класс включает три приватных поля для хранения персональных данных: имя возраст и адрес. Реализован конструктор класса с параметрами обеспечивающий инициализацию всех полей объекта при его создании. Конструктор принимает три аргумента имя возраст и адрес. Методы установки значений SetName SetAge SetAddress позволяют изменять соответствующие поля объекта после его создания. Методы получения значений GetName GetAge GetAddress предоставляют доступ для чтения к приватным полям класса что реализует принцип инкапсуляции. Дополнительно создан метод DisplayInfo который форматирует вывод информации о человеке в консоль в удобочитаемом виде с использованием интерполяции строк. Полученный код программы представлен в листинге 1.

Листинг 1. Код программы создания классов

using System;

class Person

{

// Поля класса

private string name;

private int age;

private string address;

// Конструктор

public Person(string name, int age, string address)

{

this.name = name;

this.age = age;

this.address = address;

}

// Методы для установки значений

public void SetName(string name) => this.name = name;

public void SetAge(int age) => this.age = age;

public void SetAddress(string address) => this.address = address;

// Методы для получения значений

public string GetName() => name;

public int GetAge() => age;

public string GetAddress() => address;

// Метод для вывода информации

public void DisplayInfo()

{

Console.WriteLine($"Имя: {name}");

Console.WriteLine($"Возраст: {age}");

Console.WriteLine($"Адрес: {address}");

Console.WriteLine();

}

}

class Program

{

static void Main()

{

// Создание объектов Person

Person person1 = new Person("Ксения", 17, "ул. Грицевца, д. 388");

Person person2 = new Person("Екатерина", 16, "пр. Мира, д. 25");

Person person3 = new Person("Алия", 18, "ул. Садовая, д. 15");

// Вывод информации о людях

Console.WriteLine("Информация о людях:");

person1.DisplayInfo();

person2.DisplayInfo();

person3.DisplayInfo();

// Изменение данных одного из объектов

person2.SetAge(17);

person2.SetAddress("ул. Новая, д. 5");

Console.WriteLine("После изменения данных:");

person2.DisplayInfo();

}

}

На рисунке 1 представлен результат выполнения программы.

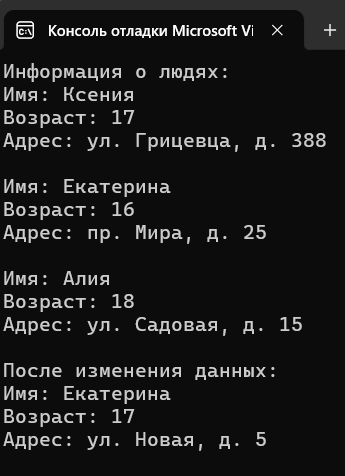


Рисунок 1 – Вывод классов

**Выполнение задания 2.** Создайте базовый класс Shape, представляющий геометрическую фигуру, и производные классы Circle и Rectangle. В базовом классе определите метод Area(), который возвращает площадь фигуры, и метод Perimeter(), который возвращает периметр фигуры. В производных классах переопределите эти методы для соответствующих фигур (круг и прямоугольник). Создайте объекты всех классов и выведите их площади и периметры.

По заданию был создан абстрактный базовый класс Shape представляющий геометрическую фигуру который определяет интерфейс для всех производных классов. В базовом классе объявлены два абстрактных метода Area и Perimeter предназначенных для вычисления площади и периметра фигуры соответственно а также виртуальный метод DisplayInfo для вывода информации о фигуре. Реализован производный класс Circle представляющий круг который включает свойство Radius для хранения радиуса и конструктор для инициализации этого свойства. В классе Circle переопределены методы Area и Perimeter где площадь вычисляется по формуле πr², а периметр по формуле 2πr. Также переопределен метод DisplayInfo для вывода специализированной информации о круге. Создан производный класс Rectangle представляющий прямоугольник с свойствами Width и Height для хранения ширины и высоты и конструктором для их инициализации. В классе Rectangle переопределены методы Area и Perimeter, где площадь вычисляется как произведение ширины на высоту а периметр по формуле 2(a+b). Метод DisplayInfo также переопределен для вывода информации о прямоугольнике. В основном методе Program Main продемонстрирована работа полиморфизма через создание объектов производных классов и сохранение их в переменных базового типа. Это представлено в листинге 2.

Листинг 2. Код программы с наследованием и полиморфизмом

using System;

// Базовый класс

abstract class Shape

{

public abstract double Area();

public abstract double Perimeter();

public virtual void DisplayInfo()

{

Console.WriteLine($"Площадь: {Area():F2}, Периметр: {Perimeter():F2}");

}

}

// Производный класс Circle

class Circle : Shape

{

public double Radius { get; set; }

public Circle(double radius)

{

Radius = radius;

}

public override double Area()

{

return Math.PI \* Radius \* Radius;

}

public override double Perimeter()

{

return 2 \* Math.PI \* Radius;

}

public override void DisplayInfo()

{

Console.WriteLine($"Круг (радиус = {Radius}): Площадь = {Area():F2}, Периметр = {Perimeter():F2}");

}

}

// Производный класс Rectangle

class Rectangle : Shape

{

public double Width { get; set; }

public double Height { get; set; }

public Rectangle(double width, double height)

{

Width = width;

Height = height;

}

public override double Area()

{

return Width \* Height;

}

public override double Perimeter()

{

return 2 \* (Width + Height);

}

public override void DisplayInfo()

{

Console.WriteLine($"Прямоугольник ({Width} x {Height}): Площадь = {Area():F2}, Периметр = {Perimeter():F2}");

}

}

class Program

{

static void Main()

{

// Создание объектов разных фигур

Shape circle = new Circle(5);

Shape rectangle = new Rectangle(4, 6);

Shape circle2 = new Circle(3.5);

Shape rectangle2 = new Rectangle(2.5, 8);

// Массив фигур для демонстрации полиморфизма

Shape[] shapes = { circle, rectangle, circle2, rectangle2 };

Console.WriteLine("Информация о геометрических фигурах:");

foreach (Shape shape in shapes)

{

shape.DisplayInfo();

}

}

}

На рисунке 2 представлен результат выполнения программы.

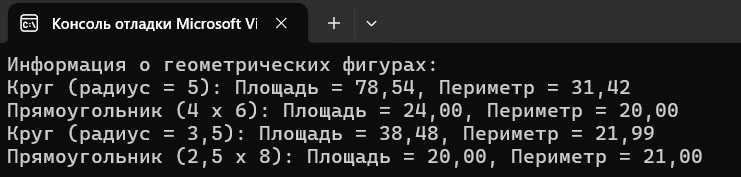


Рисунок 2 – Вывод информации о геометрических фигурах

**Выполнение задания 3.** Создайте классы Author и Book. Класс Author должен содержать информацию об авторе (имя и год рождения). Класс Book должен содержать информацию о книге (название, год выпуска и автора). Используйте композицию, чтобы связать объекты Author и Book. Создайте несколько объектов Author и Book и выведите информацию о них.

В ходе выполнения задания был создан класс Author содержащий информацию об авторе с двумя свойствами Name для хранения имени автора и BirthYear для хранения года рождения а также конструктор для инициализации этих свойств. Класс Author включает метод DisplayInfo который выводит информацию об авторе в консоль. Создан класс Book представляющий книгу который содержит свойства Title для хранения названия книги, PublicationYear для хранения года издания и свойство Author типа Author, что реализует композицию между классами. В классе Book реализован конструктор принимающий параметры для инициализации всех свойств включая объект автора а также метод DisplayInfo который выводит полную информацию о книге включая вызов метода DisplayInfo у связанного объекта Author. В основном методе Program Main продемонстрирована работа с композицией через создание трех объектов Author с различными данными об авторах затем на основе этих авторов создано четыре объекта Book где некоторые авторы связаны с несколькими книгами что показывает возможность повторного использования объектов Author. Это представлен в листинге 3.

Листинг 3. Код программы с композицией

using System;

class Author

{

public string Name { get; set; }

public int BirthYear { get; set; }

public Author(string name, int birthYear)

{

Name = name;

BirthYear = birthYear;

}

public void DisplayInfo()

{

Console.WriteLine($"Автор: {Name} (год рождения: {BirthYear})");

}

}

class Book

{

public string Title { get; set; }

public int PublicationYear { get; set; }

public Author Author { get; set; } // Композиция

public Book(string title, int publicationYear, Author author)

{

Title = title;

PublicationYear = publicationYear;

Author = author;

}

public void DisplayInfo()

{

Console.WriteLine($"Книга: \"{Title}\"");

Console.WriteLine($"Год издания: {PublicationYear}");

Author.DisplayInfo();

Console.WriteLine();

}

}

class Program

{

static void Main()

{

// Создание авторов

Author author1 = new Author("Прист", 1988);

Author author2 = new Author("Шуи Цянчэн", 1986);

Author author3 = new Author("Митбан", 0000);

// Создание книг с авторами

Book book1 = new Book("Лю Яо: Возрождение клана Фу Яо", 2014, author1);

Book book2 = new Book("Кровавая корона", 2017, author2);

Book book3 = new Book("Седьмой лорд", 2010, author1);

Book book4 = new Book("Остатки грязи", 2018, author3);

Book[] books = { book1, book2, book3, book4 };

Console.WriteLine("Библиотека книг:");

foreach (Book book in books)

{

book.DisplayInfo();

}

}

}

На рисунке 3 представлен результат выполнения программы.

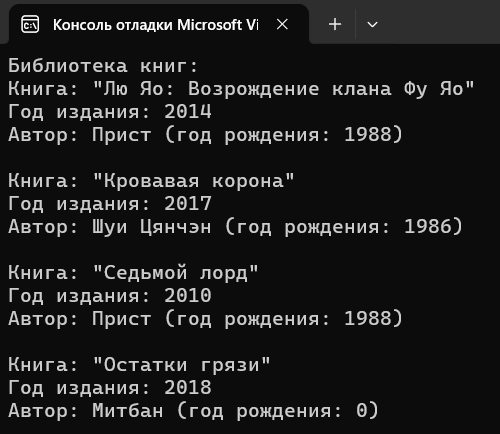


Рисунок 3 – Вывод массива с отсортированными строками

**Выполнение задания 4.** Создайте интерфейс IDrawable с методом Draw(), который выводит информацию о рисуемом объекте. Создайте классы Circle, Rectangle и Triangle, реализующие этот интерфейс. Создайте массив объектов, реализующих интерфейс IDrawable, и вызовите метод Draw() для каждого из них.

По заданию был создан интерфейс IDrawable, который определяет контракт в виде метода Draw, предназначенного для вывода информации о рисуемом объекте. Реализованы три класса геометрических фигур Circle Rectangle и Triangle каждый из которых имплементирует интерфейс IDrawable. Класс Circle содержит свойство Radius для хранения радиуса круга и конструктор для его инициализации а в методе Draw выводит информацию о радиусе круга. Класс Rectangle включает свойства Width и Height для хранения ширины и высоты прямоугольника конструктор с двумя параметрами и метод Draw выводящий размеры прямоугольника. Класс Triangle содержит три свойства SideA SideB и SideC для хранения длин сторон треугольника конструктор с тремя параметрами и метод Draw отображающий информацию о всех трех сторонах треугольника. Результат кода, представлен в листинге 4.

Листинг 4. Код программы с абстрактными классами

using System;

// Интерфейс

interface IDrawable

{

void Draw();

}

// Классы, реализующие интерфейс

class Circle : IDrawable

{

public double Radius { get; set; }

public Circle(double radius)

{

Radius = radius;

}

public void Draw()

{

Console.WriteLine($"Рисую круг с радиусом {Radius}");

}

}

class Rectangle : IDrawable

{

public double Width { get; set; }

public double Height { get; set; }

public Rectangle(double width, double height)

{

Width = width;

Height = height;

}

public void Draw()

{

Console.WriteLine($"Рисую прямоугольник {Width} x {Height}");

}

}

class Triangle : IDrawable

{

public double SideA { get; set; }

public double SideB { get; set; }

public double SideC { get; set; }

public Triangle(double a, double b, double c)

{

SideA = a;

SideB = b;

SideC = c;

}

public void Draw()

{

Console.WriteLine($"Рисую треугольник со сторонами {SideA}, {SideB}, {SideC}");

}

}

class Program

{

static void Main()

{

// Создание массива объектов, реализующих интерфейс IDrawable

IDrawable[] drawables = {

new Circle(5),

new Rectangle(4, 6),

new Triangle(3, 4, 5),

new Circle(2.5),

new Rectangle(10, 8),

new Triangle(5, 5, 5)

};

Console.WriteLine("Рисование объектов:");

// Вызов метода Draw() для каждого объекта

foreach (IDrawable drawable in drawables)

{

drawable.Draw();

}

}

}

На рисунке 4 представлен результат выполнения программы.

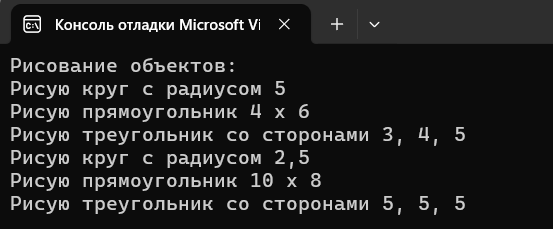


Рисунок 4 – Вывод массива с отсортированными строками

**Выполнение задания 5.** Создайте класс TemperatureSensor, который генерирует событие TemperatureChanged, когда измеренная температура меняется. Создайте класс Thermostat, который подписывается на событие TemperatureChanged и реагирует на изменение температуры, включая или выключая отопление.

Был создан пользовательский класс аргументов события TemperatureChangedEventArgs унаследованный от EventArgs который содержит два свойства OldTemperature и NewTemperature для хранения предыдущего и нового значения температуры а также конструктор для их инициализации. Реализован класс TemperatureSensor представляющий датчик температуры с приватным полем currentTemperature и свойством CurrentTemperature которое при установке нового значения проверяет изменилась ли температура и если да то вызывает защищенный виртуальный метод OnTemperatureChanged. В классе TemperatureSensor объявлено событие TemperatureChanged типа EventHandler<TemperatureChangedEventArgs> которое уведомляет подписчиков об изменении температуры. Метод OnTemperatureChanged безопасным способом вызывает событие передавая текущий объект датчика и созданный экземпляр TemperatureChangedEventArgs с данными о старой и новой температуре. Создан класс Thermostat представляющий термостат который содержит поле heatingOn для отслеживания состояния отопления и поле targetTemperature с целевой температурой 20 градусов. В конструкторе Thermostat происходит подписка на событие TemperatureChanged датчика температуры. Реализован метод-обработчик события OnTemperatureChanged который сравнивает новую температуру с целевой и принимает решение о включении или выключении отопления с гистерезисом в 1 градус чтобы избежать частых переключений. При снижении температуры ниже целевой и выключенном отоплении термостат включает обогрев а при превышении температуры на 1 градус выше целевой при включенном отоплении выключает его. В основном методе Program Main продемонстрирована работа системы через создание экземпляров TemperatureSensor и Thermostat с последующей симуляцией изменения температуры где установка новых значений в свойство CurrentTemperature приводит к генерации событий и соответствующей реакции термостата. Для наглядности процесса между изменениями температуры добавлены паузы с использованием Thread.Sleep. Код программы представлен в листинге 5.

Листинг 5. Код программы с реализацией события

using System;

// Класс для аргументов события

public class TemperatureChangedEventArgs : EventArgs

{

public double OldTemperature { get; }

public double NewTemperature { get; }

public TemperatureChangedEventArgs(double oldTemp, double newTemp)

{

OldTemperature = oldTemp;

NewTemperature = newTemp;

}

}

// Класс датчика температуры

class TemperatureSensor

{

private double currentTemperature;

// Событие

public event EventHandler<TemperatureChangedEventArgs> TemperatureChanged;

public double CurrentTemperature

{

get => currentTemperature;

set

{

if (value != currentTemperature)

{

double oldTemp = currentTemperature;

currentTemperature = value;

OnTemperatureChanged(oldTemp, currentTemperature);

}

}

}

// Метод для вызова события

protected virtual void OnTemperatureChanged(double oldTemp, double newTemp)

{

TemperatureChanged?.Invoke(this, new TemperatureChangedEventArgs(oldTemp, newTemp));

}

}

// Класс термостата

class Thermostat

{

private bool heatingOn = false;

private double targetTemperature = 20.0; // Целевая температура

public Thermostat(TemperatureSensor sensor)

{

// Подписка на событие

sensor.TemperatureChanged += OnTemperatureChanged;

}

// Обработчик события

private void OnTemperatureChanged(object sender, TemperatureChangedEventArgs e)

{

Console.WriteLine($"Температура изменилась: {e.OldTemperature:F1}°C -> {e.NewTemperature:F1}°C");

if (e.NewTemperature < targetTemperature && !heatingOn)

{

heatingOn = true;

Console.WriteLine("Включено отопление");

}

else if (e.NewTemperature >= targetTemperature + 1 && heatingOn)

{

heatingOn = false;

Console.WriteLine("Выключено отопление");

}

Console.WriteLine($"Состояние отопления: {(heatingOn ? "ВКЛ" : "ВЫКЛ")}");

Console.WriteLine();

}

}

class Program

{

static void Main()

{

TemperatureSensor sensor = new TemperatureSensor();

Thermostat thermostat = new Thermostat(sensor);

Console.WriteLine("Симуляция изменения температуры:");

// Имитация изменения температуры

sensor.CurrentTemperature = 18.0;

System.Threading.Thread.Sleep(1000);

sensor.CurrentTemperature = 19.5;

System.Threading.Thread.Sleep(1000);

sensor.CurrentTemperature = 21.0;

System.Threading.Thread.Sleep(1000);

sensor.CurrentTemperature = 19.0;

System.Threading.Thread.Sleep(1000);

sensor.CurrentTemperature = 22.5;

}

}

На рисунке 5 представлен результат выполнения программы.

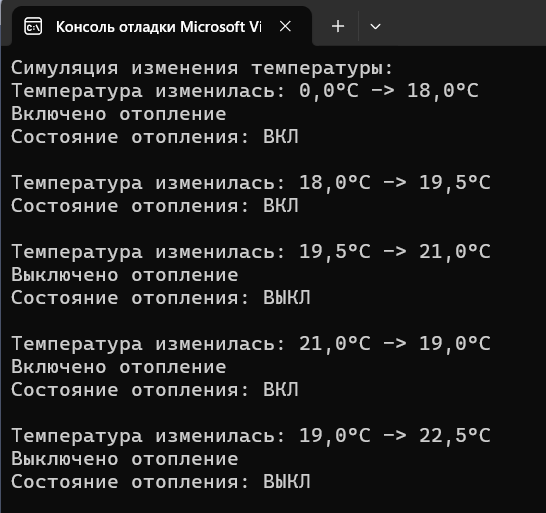


Рисунок 5 – Вывод массива с отсортированными строками

**Задания по вариантам.**

**Вариант 2.**

**Выполнение задания 1.** Создайте класс "Автомобиль" с полями: марка, модель, год выпуска и цена. Реализуйте методы для расчёта стоимости автомобиля с учётом скидки и налога на добавленную стоимость.

По заданию создался класс Car содержащий четыре свойства для хранения информации об автомобиле: Brand для марки, Model для модели, Year для года выпуска и Price для базовой стоимости. Реализован конструктор класса принимающий параметры для инициализации всех свойств объекта. Разработан метод CalculatePriceWithDiscount который вычисляет стоимость автомобиля с учётом процентной скидки при этом метод включает проверку входных данных и генерирует исключение ArgumentException если значение скидки выходит за допустимый диапазон от 0 до 100 процентов. Создан метод CalculatePriceWithVAT рассчитывающий стоимость с учётом налога на добавленную стоимость с значением НДС по умолчанию 20 процентов что позволяет использовать метод как с указанием конкретной ставки налога так и без него. Реализован комплексный метод CalculateFinalPrice который последовательно применяет скидку и затем добавляет НДС к стоимости автомобиля демонстрируя возможность комбинирования ранее созданных методов. Для вывода основной информации об автомобиле создан метод DisplayInfo форматирующий данные в удобочитаемом виде с использованием специального формата валюты для отображения цен. Код полученной программы представлен в листинге 6.

Листинг 6. Код программы создания классов

using System;

class Car

{

public string Brand { get; set; }

public string Model { get; set; }

public int Year { get; set; }

public decimal Price { get; set; }

public Car(string brand, string model, int year, decimal price)

{

Brand = brand;

Model = model;

Year = year;

Price = price;

}

// Метод для расчета стоимости со скидкой

public decimal CalculatePriceWithDiscount(decimal discountPercent)

{

if (discountPercent < 0 || discountPercent > 100)

throw new ArgumentException("Скидка должна быть от 0 до 100%");

return Price \* (1 - discountPercent / 100);

}

// Метод для расчета стоимости с НДС

public decimal CalculatePriceWithVAT(decimal vatPercent = 20)

{

return Price \* (1 + vatPercent / 100);

}

// Метод для расчета стоимости со скидкой и НДС

public decimal CalculateFinalPrice(decimal discountPercent, decimal vatPercent = 20)

{

decimal priceAfterDiscount = CalculatePriceWithDiscount(discountPercent);

return priceAfterDiscount \* (1 + vatPercent / 100);

}

public void DisplayInfo()

{

Console.WriteLine($"Автомобиль: {Brand} {Model}");

Console.WriteLine($"Год выпуска: {Year}");

Console.WriteLine($"Базовая цена: {Price:C}");

}

}

class Program

{

static void Main()

{

Car car1 = new Car("Toyota", "Camry", 2022, 2500000);

Car car2 = new Car("BMW", "X5", 2023, 5000000);

Console.WriteLine("Расчет стоимости автомобилей:");

// Первый автомобиль

car1.DisplayInfo();

Console.WriteLine($"Цена со скидкой 10%: {car1.CalculatePriceWithDiscount(10):C}");

Console.WriteLine($"Цена с НДС: {car1.CalculatePriceWithVAT():C}");

Console.WriteLine($"Итоговая цена (скидка 10% + НДС): {car1.CalculateFinalPrice(10):C}");

Console.WriteLine();

// Второй автомобиль

car2.DisplayInfo();

Console.WriteLine($"Цена со скидкой 15%: {car2.CalculatePriceWithDiscount(15):C}");

Console.WriteLine($"Цена с НДС: {car2.CalculatePriceWithVAT():C}");

Console.WriteLine($"Итоговая цена (скидка 15% + НДС): {car2.CalculateFinalPrice(15):C}");

}

}

На рисунке 6 представлен результат выполнения программы.

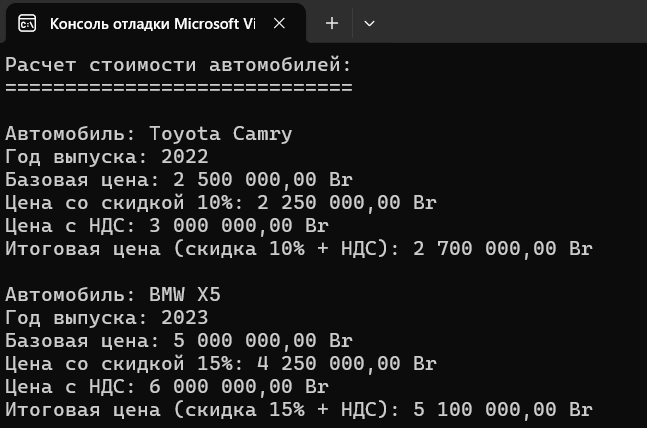


Рисунок 6 – Вывод классов

**Выполнение задания 2.** Создайте структуру с именем student, содержащую поля: фамилия и инициалы, номер группы, успеваемость (массив из пяти элементов). Создать массив из десяти элементов такого типа, упорядочить записи по возрастанию среднего балла. Добавить возможность вывода фамилий и номеров групп студентов, имеющих оценки, равные только 4 или 5.

Была создана структура Student содержащая три поля: LastNameAndInitials для хранения фамилии и инициалов студента, GroupNumber для номера учебной группы и Grades представляющий массив из пяти целых чисел для хранения академических оценок. Реализован конструктор структуры обеспечивающий инициализацию всех полей. Разработан метод GetAverageGrade вычисляющий средний балл студента с использованием метода Average из пространства имён System.Linq что демонстрирует интеграцию возможностей LINQ в пользовательские структуры. Создан метод HasOnlyGoodGrades проверяющий соответствие всех оценок студента критерию высокой успеваемости то есть наличию только оценок 4 и 5 с применением метода All и лямбда-выражения для компактной и эффективной проверки условия. Метод DisplayInfo обеспечивает форматированный вывод информации о студенте включая все оценки и вычисленный средний балл с выравниванием столбцов для улучшенной читаемости. В основном методе Program Main создан и инициализирован массив из десяти элементов типа Student с различными данными об успеваемости. Для сортировки массива студентов по возрастанию среднего балла применён алгоритм пузырьковой сортировки с вложенными циклами и временной переменной для обмена элементов что демонстрирует работу с ценными типами и необходимость прямого присваивания при перестановке элементов массива. После сортировки выполнен вывод всего отсортированного списка студентов с подробной информацией об успеваемости каждого. Для фильтрации студентов с высокой успеваемостью использован метод расширения Where в сочетании с ранее созданным методом HasOnlyGoodGrades а затем выводятся только фамилии и номера групп соответствующих студентов. В заключение программа выводит статистическую информацию об общем количестве студентов и количестве учащихся с высокой успеваемостью используя свойства Length и Count. Код программы представлен в листинге 7.

Листинг 7. Код программы создания классов

using System;

using System.Linq;

struct Student

{

public string LastNameAndInitials;

public string GroupNumber;

public int[] Grades; // массив из 5 оценок

public Student(string lastName, string group, int[] grades)

{

LastNameAndInitials = lastName;

GroupNumber = group;

Grades = grades;

}

// Метод для вычисления среднего балла

public double GetAverageGrade()

{

return Grades.Average();

}

// Метод для проверки, все ли оценки 4 или 5

public bool HasOnlyGoodGrades()

{

return Grades.All(grade => grade == 4 || grade == 5);

}

public void DisplayInfo()

{

Console.Write($"{LastNameAndInitials,-20} {GroupNumber,-10} ");

foreach (int grade in Grades)

{

Console.Write($"{grade} ");

}

Console.WriteLine($"| Средний: {GetAverageGrade():F2}");

}

}

class Program

{

static void Main()

{

// Создание массива студентов

Student[] students = new Student[10];

// Инициализация студентов

students[0] = new Student("Овечкин А.И.", "ПОИС62", new int[] { 5, 4, 5, 5, 4 });

students[1] = new Student("Быков Б.С.", "ПОИС62", new int[] { 3, 4, 3, 5, 4 });

students[2] = new Student("Козий В.К.", "ПОИС62", new int[] { 5, 5, 5, 5, 5 });

students[3] = new Student("Петухова Д.М.", "ПОИС62", new int[] { 4, 4, 4, 4, 4 });

students[4] = new Student("Курицев Е.П.", "ПОИС62", new int[] { 3, 3, 4, 3, 4 });

students[5] = new Student("Собычкин Ж.Р.", "ПОИС62", new int[] { 5, 4, 5, 4, 5 });

students[6] = new Student("Петров З.Т.", "ПОИС62", new int[] { 4, 5, 4, 5, 4 });

students[7] = new Student("Котов Ю.У.", "ПОИС62", new int[] { 3, 4, 3, 3, 4 });

students[8] = new Student("Лошаднева К.Ч.", "ПОИС62", new int[] { 5, 5, 4, 5, 5 });

students[9] = new Student("Свинев Л.Я.", "ПОИС62", new int[] { 4, 4, 5, 4, 4 });

// Сортировка по возрастанию среднего балла

for (int i = 0; i < students.Length - 1; i++)

{

for (int j = i + 1; j < students.Length; j++)

{

if (students[i].GetAverageGrade() > students[j].GetAverageGrade())

{

Student temp = students[i];

students[i] = students[j];

students[j] = temp;

}

}

}

// Вывод отсортированного списка

Console.WriteLine("Список студентов (отсортирован по среднему баллу):"); Console.WriteLine("Фамилия и инициалы Группа Оценки Средний балл");

foreach (Student student in students)

{

student.DisplayInfo();

}

// Вывод студентов с оценками только 4 и 5

Console.WriteLine("\nСтуденты с оценками только 4 и 5:");

var excellentStudents = students.Where(s => s.HasOnlyGoodGrades());

if (excellentStudents.Any())

{

foreach (Student student in excellentStudents)

{

Console.WriteLine($"{student.LastNameAndInitials} - {student.GroupNumber}");

}

}

else

{

Console.WriteLine("Таких студентов нет");

}

Console.WriteLine($"\nВсего студентов: {students.Length}");

Console.WriteLine($"Из них с оценками только 4 и 5: {excellentStudents.Count()}");

}

}

На рисунке 7 представлен результат выполнения программы.

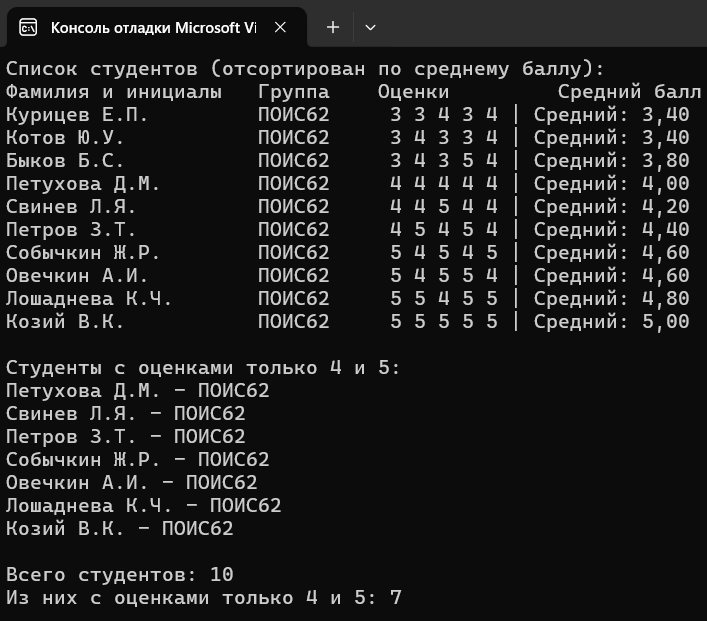


Рисунок 7 – Вывод классов

**Выполнение задания 3.** Создание класса "Геометрическая фигура" (Shape) и его производных: Создайте абстрактный класс "Геометрическая фигура" с методом для вычисления площади. Создайте производные классы, например, "Круг" (Circle), "Прямоугольник" (Rectangle), "Треугольник" (Triangle) и т. д. Реализуйте методы для вычисления площади у каждого класса. Создайте объекты различных классов и протестируйте их функциональность.

Создан абстрактный класс Shape определяющий общий интерфейс для всех геометрических фигур. Класс содержит абстрактный метод CalculateArea предназначенный для вычисления площади фигуры а также виртуальный метод DisplayInfo обеспечивающий вывод информации о фигуре и ее площади. Реализован класс Circle представляющий геометрическую фигуру круг с свойством Radius для хранения радиуса и конструктором для его инициализации. В классе переопределен метод CalculateArea вычисляющий площадь круга по стандартной математической формуле πr² с использованием константы Math.PI. Также переопределен метод DisplayInfo выводящий специализированную информацию о круге включая значение радиуса. Создан класс Rectangle представляющий прямоугольник с свойствами Width и Height для хранения ширины и высоты соответственно и конструктором с двумя параметрами. В классе переопределен метод CalculateArea вычисляющий площадь прямоугольника как произведение ширины на высоту а метод DisplayInfo выводит информацию о размерах прямоугольника и вычисленной площади. Реализован класс Triangle представляющий треугольник с свойствами Base для основания и Height для высоты и соответствующим конструктором. В классе переопределен метод CalculateArea вычисляющий площадь треугольника по стандартной формуле ½ основания на высоту а метод DisplayInfo выводит детальную информацию о параметрах треугольника и его площади. В основном методе Program Main продемонстрирован принцип полиморфизма через создание массива объектов типа Shape содержащего экземпляры всех трех производных классов с различными параметрами. Код программы представлен в листинге 8.

Листинг 8. Код программы создания классов

using System;

// Абстрактный класс

abstract class Shape

{

public abstract double CalculateArea();

public virtual void DisplayInfo()

{

Console.WriteLine($"Площадь: {CalculateArea():F2}");

}

}

// Класс Circle

class Circle : Shape

{

public double Radius { get; set; }

public Circle(double radius)

{

Radius = radius;

}

public override double CalculateArea()

{

return Math.PI \* Radius \* Radius;

}

public override void DisplayInfo()

{

Console.WriteLine($"Круг (радиус = {Radius}): Площадь = {CalculateArea():F2}");

}

}

// Класс Rectangle

class Rectangle : Shape

{

public double Width { get; set; }

public double Height { get; set; }

public Rectangle(double width, double height)

{

Width = width;

Height = height;

}

public override double CalculateArea()

{

return Width \* Height;

}

public override void DisplayInfo()

{

Console.WriteLine($"Прямоугольник ({Width} x {Height}): Площадь = {CalculateArea():F2}");

}

}

// Класс Triangle

class Triangle : Shape

{

public double Base { get; set; }

public double Height { get; set; }

public Triangle(double triangleBase, double height)

{

Base = triangleBase;

Height = height;

}

public override double CalculateArea()

{

return 0.5 \* Base \* Height;

}

public override void DisplayInfo()

{

Console.WriteLine($"Треугольник (основание = {Base}, высота = {Height}): Площадь = {CalculateArea():F2}");

}

}

class Program

{

static void Main()

{

// Создание объектов разных фигур

Shape[] shapes = {

new Circle(5),

new Rectangle(4, 6),

new Triangle(3, 4),

new Circle(2.5),

new Rectangle(10, 8),

new Triangle(5, 7)

};

Console.WriteLine("Площади геометрических фигур:");

// Вывод информации о всех фигурах

foreach (Shape shape in shapes)

{

shape.DisplayInfo();

}

}

}

На рисунке 8 представлен результат выполнения программы.

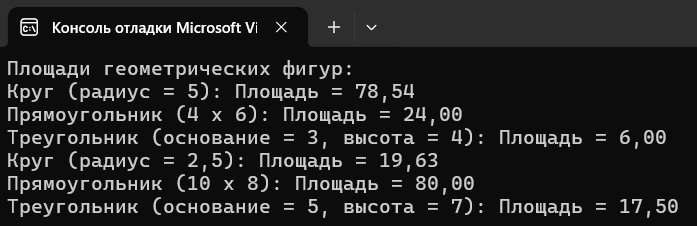


Рисунок 8 – Вывод классов