Кам’янець-Подільський національний університет

 імені Івана Огієнка

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

Навчальна дисципліна «Об’єктно-орієнтоване програмування»

ЛАБОРАТОРНА РОБОТА #0104

Тема: Поліморфізм

Варіант №3

Виконав:

студент 1-го курсу

групи KNms1-B21

Волошин В.Ю.

Прийняла:

старший викладач,

Мястковська М.О.

Кам’янець-Подільський – 2023

1. **Хід роботи.**

Поліморфізм є ключовою концепцією в об'єктно-орієнтованому програмуванні, яка дозволяє об'єктам вести себе по-різному в залежності від контексту використання. Термін "поліморфізм" означає багато форм, вказуючи на здатність об'єктів приймати різні форми або використовуватися по-різному в різних ситуаціях.

1. У мові програмування C# існують три основні варіанти поліморфізму:

* Компіляційний поліморфізм (Статичний поліморфізм):
* Рішення приймається на етапі компіляції.
* Включає в себе перевантажені методи, оператори або властивості.

Приклад: перевантаження методу за його сигнатурою.

1. Ранньо-зв'язаний поліморфізм (Визначення типу під час компіляції):

* Тип об'єкта відомий на етапі компіляції.
* Включає використання віртуальних методів та перевантажених операторів.

1. Пізньо-зв'язаний поліморфізм (Визначення типу під час виконання):

* Тип об'єкта визначається під час виконання програми.
* Включає використання віртуальних методів та концепцію абстрактного класу.

Віртуальні методи та абстрактні класи є ключовими засобами реалізації раннього та пізнього зв'язування. Віртуальні методи можуть бути перевизначені в похідних класах, що надає гнучкість у визначенні функціональності. Абстрактні класи, з своєю чергою, визначають абстрактні методи, які повинні бути реалізовані в похідних класах.

Використання поліморфізму сприяє гнучкості коду, повторному використанню та сховищу деталей реалізації. Ця концепція дозволяє створювати більш чистий та розширюваний код, зменшуючи залежність від конкретних класів та підтримуючи принципи ООП.

1. **Завдання до виконання.**

**Завдання 1.**

Створити додаток, який задовольняє вимогам, наведеним в завданні. Успадкування застосовувати тільки в тих завданнях, в яких це логічно обґрунтоване. Аргументувати приналежність класу кожного створюваного методу і коректно перевизначити для кожного класу методи equals(), hashCode(), toString().

Створити об'єкт класу Літак, використовуючи класи Крило, Шасі, Двигун. Методи: літати, задавати маршрут, вивести на консоль маршрут.

**Завдання 2.**

У кожному варіанті завдання при описі класів самостійно визначити необхідні поля, властивості та методи вводу/виводу. Деякі методи класу-предка повинні бути віртуальними і абстрактними. У програмі-клієнті для збереження сукупності об’єктів використати масив.

Створити клас TBody, який представляє просторову геометричну фігуру з методами обчислення площі поверхні та об’єму. На основі цього класу створити класи нащадки TParallelepiped та TBall. Випадковим чином створити певну кількість паралелепіпедів та куль, щоб їх сумарна кількість дорівнювала *n* . Знайти сумарну площу поверхонь усіх геометричних тіл.

1. **Лістинг програми.**

class Wing

{

    // Представляє довжину крила

    public int Length { get; }

    // Конструктор для ініціалізації довжини крила

    public Wing(int length)

    {

        Length = length;

    }

    // Перевизначає метод Equals для власного порівняння за довжиною крила

    public override bool Equals(object obj)

    {

        if (this == obj)

            return true;

        if (obj == null || GetType() != obj.GetType())

            return false;

        Wing other = (Wing)obj;

        return Length == other.Length;

    }

    // Перевизначає метод GetHashCode для генерації хеш-коду об'єкта крила

    public override int GetHashCode()

    {

        return HashCode.Combine(Length);

    }

    // Перевизначає метод ToString для отримання рядкового представлення об'єкта крила

    public override string ToString()

    {

        return $"Крила: Довжина - {Length}";

    }

}

class Chassis

{

    // Представляє тип шасі (посадкового обладнання) літака

    public string Type { get; }

    // Конструктор для ініціалізації типу шасі

    public Chassis(string type)

    {

        Type = type;

    }

    // Перевизначає метод Equals для власного порівняння за типом шасі

    public override bool Equals(object obj)

    {

        if (this == obj)

            return true;

        if (obj == null || GetType() != obj.GetType())

            return false;

        Chassis other = (Chassis)obj;

        return Type == other.Type;

    }

    // Перевизначає метод GetHashCode для генерації хеш-коду об'єкта шасі

    public override int GetHashCode()

    {

        return HashCode.Combine(Type);

    }

    // Перевизначає метод ToString для отримання рядкового представлення об'єкта шасі

    public override string ToString()

    {

        return $"Двигун: Тип Ходова: Тип - {Type}";

    }

}

class Engine

{

    // Представляє тип двигуна літака

    public string Type { get; }

    // Конструктор для ініціалізації типу двигуна

    public Engine(string type)

    {

        Type = type;

    }

    // Перевизначає метод Equals для власного порівняння за типом двигуна

    public override bool Equals(object obj)

    {

        if (this == obj)

            return true;

        if (obj == null || GetType() != obj.GetType())

            return false;

        Engine other = (Engine)obj;

        return Type == other.Type;

    }

    // Перевизначає метод GetHashCode для генерації хеш-коду об'єкта двигуна

    public override int GetHashCode()

    {

        return HashCode.Combine(Type);

    }

    // Перевизначає метод ToString для отримання рядкового представлення об'єкта двигуна

    public override string ToString()

    {

        return $"Двигун: Тип - {Type}";

    }

}

class Airplane : Wing

{

    private readonly Chassis \_chassis;

    private readonly Engine \_engine;

    // Конструктор для ініціалізації об'єкта літака зазначеною довжиною крила, шасі та двигуном

    public Airplane(int wingLength, Chassis chassis, Engine engine)

        : base(wingLength)

    {

        \_chassis = chassis;

        \_engine = engine;

    }

    // Метод, що виводить повідомлення про польот

    public void Fly()

    {

        Console.WriteLine("Літак летить!");

    }

    // Метод для встановлення маршруту

    public void SetRoute(string route)

    {

        Console.WriteLine($"Набір маршруту: {route}");

    }

    // Метод для відображення маршруту на консолі

    public void DisplayRoute()

    {

        Console.WriteLine("Маршрут, що відображається на консолі.");

    }

    // Перевизначає метод Equals для власного порівняння об'єктів літака

    public override bool Equals(object obj)

    {

        if (this == obj)

            return true;

        if (obj == null || GetType() != obj.GetType())

            return false;

        if (!base.Equals(obj))

            return false;

        Airplane other = (Airplane)obj;

        return \_chassis.Equals(other.\_chassis) && \_engine.Equals(other.\_engine);

    }

    // Перевизначає метод GetHashCode для генерації хеш-коду об'єкта літака

    public override int GetHashCode()

    {

        return HashCode.Combine(base.GetHashCode(), \_chassis, \_engine);

    }

    // Перевизначає метод ToString для отримання рядкового представлення об'єкта літака

    public override string ToString()

    {

        return $"Літак: {base.ToString()}, {\_chassis}, {\_engine}";

    }

}

abstract class TBody

{

    // Абстрактний клас, який представляє тіло

    public abstract double CalculateSurfaceArea();  // Абстрактний метод для розрахунку площі поверхні

    public abstract double CalculateVolume();       // Абстрактний метод для розрахунку об'єму

    // Віртуальний метод для відображення інформації про тіло

    public virtual void DisplayInfo()

    {

        Console.WriteLine($"Область поверхні: {CalculateSurfaceArea()}");

        Console.WriteLine($"Об'єм: {CalculateVolume()}");

    }

}

class TParallelepiped : TBody

{

    // Клас, який представляє паралелепіпед

    public double Length { get; }

    public double Width { get; }

    public double Height { get; }

    // Конструктор для ініціалізації довжини, ширини та висоти паралелепіпеда

    public TParallelepiped(double length, double width, double height)

    {

        Length = length;

        Width = width;

        Height = height;

    }

    // Реалізація абстрактного методу для розрахунку площі поверхні паралелепіпеда

    public override double CalculateSurfaceArea()

    {

        return 2 \* (Length \* Width + Width \* Height + Height \* Length);

    }

    // Реалізація абстрактного методу для розрахунку об'єму паралелепіпеда

    public override double CalculateVolume()

    {

        return Length \* Width \* Height;

    }

}

class TBall : TBody

{

    // Клас, який представляє кулю

    public double Radius { get; }

    // Конструктор для ініціалізації радіусу кулі

    public TBall(double radius)

    {

        Radius = radius;

    }

    // Реалізація абстрактного методу для розрахунку площі поверхні кулі

    public override double CalculateSurfaceArea()

    {

        return 4 \* Math.PI \* Radius \* Radius;

    }

    // Реалізація абстрактного методу для розрахунку об'єму кулі

    public override double CalculateVolume()

    {

        return (4.0 / 3.0) \* Math.PI \* Math.Pow(Radius, 3);

    }

}

class Program

{

    static void Main()

    {

        // Завдання 1

        // Створення об'єктів для кожного компонента літака

        Wing wing = new Wing(50);

        Chassis chassis = new Chassis("Пасажирський");

        Engine engine = new Engine("Турбоджет");

        // Створення об'єкта літака та ініціалізація його компонентами

        Airplane airplane = new Airplane(50, chassis, engine);

        // Виклик методів для демонстрації роботи літака

        airplane.Fly();

        airplane.SetRoute("Київ - Париж");

        airplane.DisplayRoute();

        // Виведення інформації про кожен компонент

        Console.WriteLine(wing.ToString());

        Console.WriteLine(chassis.ToString());

        Console.WriteLine(engine.ToString());

        Console.WriteLine(airplane.ToString());

        // Порівняння об'єктів за допомогою Equals

        Console.WriteLine($"Чи рівні крила? {wing.Equals(new Wing(50))}");

        Console.WriteLine($"Чи рівні літаки? {airplane.Equals(new Airplane(50, chassis, engine))}");

        Console.WriteLine();

        // Завдання 2

        // Масив об'єктів типу TBody

        TBody[] bodies = new TBody[5];

        // Ініціалізація об'єктів масиву з випадковими значеннями

        Random random = new Random();

        for (int i = 0; i < bodies.Length; i++)

        {

            if (random.Next(2) == 0)

            {

                // Якщо випадкове число дорівнює 0, створюється об'єкт паралелепіпеда

                double length = random.Next(1, 10);

                double width = random.Next(1, 10);

                double height = random.Next(1, 10);

                bodies[i] = new TParallelepiped(length, width, height);

            }

            else

            {

                // Якщо випадкове число дорівнює 1, створюється об'єкт кулі

                double radius = random.Next(1, 10);

                bodies[i] = new TBall(radius);

            }

        }

        // Підрахунок загальної площі поверхні

        double totalSurfaceArea = 0;

        // Виведення інформації про кожен об'єкт масиву

        foreach (var body in bodies)

        {

            Console.WriteLine($"Тип фігури: {body.GetType()}");

            body.DisplayInfo();

            totalSurfaceArea += body.CalculateSurfaceArea();

            Console.WriteLine("--------------------");

        }

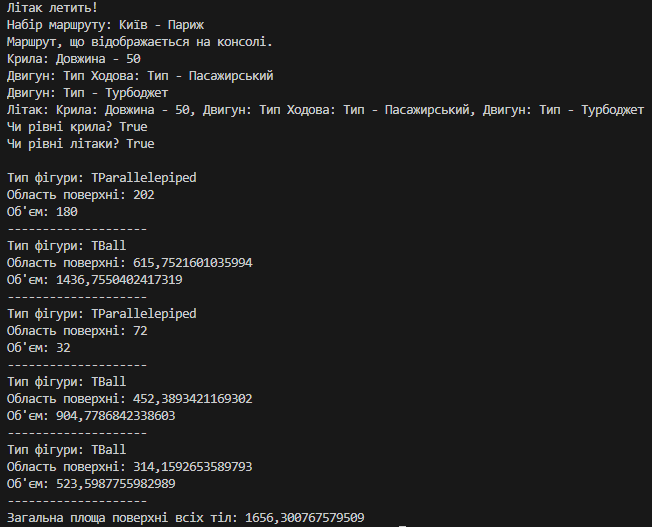
        // Виведення загальної площі поверхні всіх об'єктів

        Console.WriteLine($"Загальна площа поверхні всіх тіл: {totalSurfaceArea}");

    }

}

1. **Результат роботи.**



1. **Посилання на проект.**