### Кам’янець-Подільський національний університет імені Івана Огієнка

### КАФЕДРА КОМП’ЮТЕРНИХ НАУК

### Навчальна дисципліна «Об’єктно-орієнтоване програмування»

### ЛАБОРАТОРНА РОБОТА #04

## Тема:

Поліморфізм.

Варіант №9

**Виконав**:  
студент 2-го курсу  
групи KN1-B23  
Серебрянський Артем О.

**Прийняв**:  
доцент,  
Слободянюк О.В.

### Кам’янець-Подільський – 2025

1. **Короткі теоретичні відомості:**

#### **1. Поліморфізм**

Поліморфізм – наступний з трьох принципів ООП. Його застосування дозволяє створювати у похідних класах методи, які мають той самий інтерфейс, що й методи базового класу, проте відмінну функціональність.

Завдяки поліморфізму можна викликати однакові за іменем методи, які працюють по-різному в базовому та похідних класах.

1. **Повні умови завдань:**

**Задача 1.**

Створити об'єкт класу Птах, використовуючи класи Крила, Дзьоб. Методи: літати, сідати, харчуватися, атакувати.  
**Задача 2.**Створити клас TVector, який представляє вектор і місить методи для обчислення довжини вектора та скалярного добутку векторів. На основі цього класу створити класи, які представляють вектори з просторів R2 та R3 За допомогою цих класів обчислити значення виразу

1. **Послідовність виконання завдань:  
   Задача 1.**

* **Створення класів.  
  Реалізуйте класи Krula, dzob та Bird, додавши методи для поведінки птаха (летіти, сідати, харчуватися, атакувати).**
* **Перевизначення методів:   
  Перевизначте методи Equals(), GetHashCode() та ToString() для коректного порівняння об'єктів і виведення їхньої інформації.**
* **Тестування та перевірка:   
  Створіть об'єкти класів, перевірте їх взаємодію через методи, перевірте рівність об'єктів і виведення опису.  
  Задача 2.**
* **Розробка базового класу TVector:   
  Створити абстрактний клас TVector з методами Length() (довжина вектора) і DotProduct() (скалярний добуток).**
* **Створення похідних класів Vector2D і Vector3D  
  Реалізувати класи для 2D- та 3D-векторів з відповідними координатами.  
  Перевизначити методи Length() і DotProduct().  
  Перевизначити оператор + для додавання векторів.**
* **Розробка програми-клієнта (Main)  
  Створити масив векторів і виконати необхідні обчислення.  
  Обчислити вираз   
  Вивести результат у консоль.**

**4. Лістинги програм:**

**Задача 1.**using System;

using System.Text;

public class Krula

{

public override bool Equals(object obj)

{

if (obj is Krula)

{

return true;

}

return false;

}

public override int GetHashCode()

{

return base.GetHashCode();

}

public void Flap()

{

Console.WriteLine("Крила махають.");

}

}

public class dzob

{

public override bool Equals(object obj)

{

if (obj is dzob)

{

return true;

}

return false;

}

public override int GetHashCode()

{

return base.GetHashCode();

}

public void Peck()

{

Console.WriteLine("Дзьоб клює.");

}

}

public class Bird

{

public Krula Krula { get; set; }

public dzob dzob { get; set; }

public Bird()

{

Console.OutputEncoding = Encoding.UTF8;

Console.InputEncoding = Encoding.UTF8;

Krula = new Krula();

dzob = new dzob();

}

public void Fly()

{

Console.WriteLine("Птах летить.");

Krula.Flap();

}

public void sidatu()

{

Console.WriteLine("Птах сів.");

}

public void xarchyvatuca()

{

Console.WriteLine("Птах харчується.");

dzob.Peck();

}

public void Attack()

{

Console.WriteLine("Птах атакує.");

}

public override bool Equals(object obj)

{

if (obj is Bird)

{

Bird otherBird = (Bird)obj;

return this.Krula.Equals(otherBird.Krula) && this.dzob.Equals(otherBird.dzob);

}

return false;

}

public override int GetHashCode()

{

return (Krula?.GetHashCode() ?? 0) ^ (dzob?.GetHashCode() ?? 0);

}

public override string ToString()

{

return $"Птах з крильцями і дзьобом: {Krula.GetType().Name}, {dzob.GetType().Name}";

}

}

public class Program

{

public static void Main()

{

Bird bird = new Bird();

bird.Fly();

bird.sidatu();

bird.xarchyvatuca();

bird.Attack();

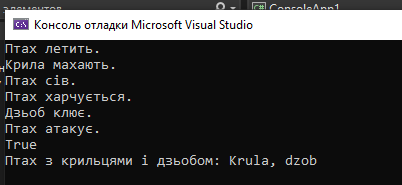
Bird anotherBird = new Bird();

Console.WriteLine(bird.Equals(anotherBird));

Console.WriteLine(bird.ToString());

}

}



**Задача 2.**using System;

abstract class TVector

{

public abstract double Length();

public abstract double DotProduct(TVector other);

}

class Vector2D : TVector

{

public double X { get; set; }

public double Y { get; set; }

public Vector2D(double x, double y)

{

X = x;

Y = y;

}

public override double Length()

{

return Math.Sqrt(X \* X + Y \* Y);

}

public override double DotProduct(TVector other)

{

if (other is Vector2D v)

return X \* v.X + Y \* v.Y;

throw new ArgumentException("Vector dimensions do not match.");

}

public static Vector2D operator +(Vector2D a, Vector2D b)

{

return new Vector2D(a.X + b.X, a.Y + b.Y);

}

}

class Vector3D : TVector

{

public double X { get; set; }

public double Y { get; set; }

public double Z { get; set; }

public Vector3D(double x, double y, double z)

{

X = x;

Y = y;

Z = z;

}

public override double Length()

{

return Math.Sqrt(X \* X + Y \* Y + Z \* Z);

}

public override double DotProduct(TVector other)

{

if (other is Vector3D v)

return X \* v.X + Y \* v.Y + Z \* v.Z;

throw new ArgumentException("Vector dimensions do not match.");

}

public static Vector3D operator +(Vector3D a, Vector3D b)

{

return new Vector3D(a.X + b.X, a.Y + b.Y, a.Z + b.Z);

}

}

class Program

{

static void Main()

{

Vector2D a = new Vector2D(1, 2);

Vector2D b = new Vector2D(3, 4);

Vector2D c = new Vector2D(5, 6);

Vector2D d = new Vector2D(7, 8);

Vector2D sum = (a + b) + (c + d);

double lengthA = a.Length();

double result = sum.Length() + lengthA;

Console.WriteLine($"S = {result}");

}

}

**5. Посилання на GitHub:**  
<https://github.com/Arhangell228/-/tree/main>