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

### КАФЕДРА КОМП’ЮТЕРНИХ НАУК

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

### ЛАБОРАТОРНА РОБОТА #0103

## Тема:

**Основи об’єктно-орієнтованого програмування**

Варіант №1

**Виконав**:  
студент 1-го курсу  
групи KNms1-B24  
Білик Я. Ю.

**Прийняв**:  
старший викладач,  
Слободянюк О. В.

### Кам’янець-Подільський – 2025

1. **Короткі теоретичні відомості.**

Наслідування — це механізм, що дозволяє похідному класу отримати доступ до полів, методів і властивостей базового класу. Це основа повторного використання коду та реалізації поліморфізму.

Віртуальні методи і властивості — це такі члени базового класу, які дозволяють похідному класу змінювати їхню поведінку за допомогою перевизначення.

Перевизначення забезпечує поліморфізм — можливість об'єкта поводитись по-різному залежно від свого фактичного типу, навіть якщо він використовується через посилання на базовий тип.

Приховування — це створення нового члена в похідному класі з тією ж назвою, що в базовому, без перевизначення. У такому випадку новий член приховує член базового класу.

Відмінність полягає в тому, що перевизначення дає змогу викликати метод похідного класу навіть через посилання на базовий тип, а приховування — ні. Поведінка прихованого члена залежить від типу змінної, а не фактичного об'єкта.

Абстрактні класи — це класи, які не можна створити напряму. Вони слугують шаблоном для похідних класів і можуть містити як реалізовані, так і нереалізовані (абстрактні) методи.

System.Object — це базовий клас для всіх типів у C#. Він містить основні методи, такі як ToString, Equals, GetHashCode і GetType, які можна перевизначити для зміни стандартної поведінки.

Узагальнені типи (generics) дають змогу створювати класи, методи та інтерфейси з параметрами типів, що підвищує гнучкість і безпечність коду.

Обмеження задають умови для параметра типу: наприклад, щоб він був класом, структурою, мав конструктор без параметрів або успадковував певний тип. Узагальнені типи можуть наслідувати інші типи, як звичайні, так і узагальнені.

1. **Повні умови завдань.**

Задача 1.

Тварини та наслідування. Створіть базовий клас Animal із методом MakeSound(). Реалізуйте похідні класи Dog і Cat, кожен із власним перевизначенням методу.

Задача 2.

Віртуальні методи у спадкуванні. Реалізуйте базовий клас із віртуальним методом Describe(). Створіть похідні класи Car і Bike, які перевизначають цей метод.

Задача 3.

Сортування об'єктів. Реалізуйте клас Person з властивостями Name і Age. Реалізуйте інтерфейс IComparable для сортування за віком за допомогою методу sort().

1. **Лістинги програм.**

Задача 1

using System;

class Animal

{

public virtual void MakeSound()

{

Console.WriteLine("Звуки тварин");

}

}

class Dog : Animal

{

public override void MakeSound()

{

Console.WriteLine("Гав!");

}

}

class Cat : Animal

{

public override void MakeSound()

{

Console.WriteLine("Мяв!");

}

}

class Program

{

static void Main()

{

Animal dog = new Dog();

Animal cat = new Cat();

dog.MakeSound();

cat.MakeSound();

}

}

**Задача 2**

using System;

class Vehicle

{

public virtual void Describe()

{

Console.WriteLine("Це транспортний засіб.");

}

}

class Car : Vehicle

{

public override void Describe()

{

Console.WriteLine("Це автомобіль. Має чотири колеса та двигун.");

}

}

class Bike : Vehicle

{

public override void Describe()

{

Console.WriteLine("Це велосипед. Має два колеса та рухається за допомогою педалей.");

}

}

class Program

{

static void Main()

{

Vehicle genericVehicle = new Vehicle();

Vehicle myCar = new Car();

Vehicle myBike = new Bike();

genericVehicle.Describe();

myCar.Describe();

myBike.Describe();

}

}

**Задача 3**

using System;

using System.Collections.Generic;

class Person : IComparable<Person>

{

public string Name { get; set; }

public int Age { get; set; }

public Person(string name, int age)

{

Name = name;

Age = age;

}

public int CompareTo(Person other)

{

if (other == null) return 1;

return this.Age.CompareTo(other.Age);

}

public override string ToString()

{

return $"{Name}, {Age} років";

}

}

class Program

{

static void Main()

{

List<Person> people = new List<Person>

{

new Person("Андрій", 25),

new Person("Оксана", 30),

new Person("Ірина", 20)

};

people.Sort();

foreach (var person in people)

{

Console.WriteLine(person);

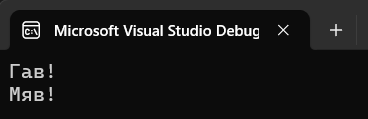
}

}

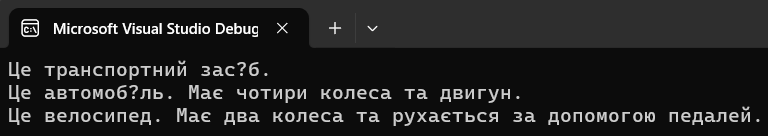
}

1. **Результати роботи програм.**

Задача 1



Задача 2



Задача 3

