Кам’янець-Подільський національний університет

 імені Івана Огієнка

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

Навчальна дисципліна «Об’єктно-орієнтоване програмування»

ЛАБОРАТОРНА РОБОТА #0103

Тема: Спадкування

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.**

Спадкування в мові програмування C# є ключовим механізмом об'єктно-орієнтованого програмування, що дозволяє створювати нові класи на основі існуючих. Ця концепція дозволяє використовувати та розширювати функціональність вже існуючих класів, що сприяє впорядкованому та повторному використанню коду.

Спадкування включає два основні класи: базовий клас і похідний клас. Базовий клас містить загальні атрибути та методи, які можуть бути спільно використані іншими класами. Похідний клас успадковує ці атрибути та методи і може додавати свою власну функціональність або перевизначати існуючі.

Однією з важливих переваг спадкування є можливість створювати ієрархії класів, де багато класів можуть успадковувати властивості та методи одного базового класу. Це спрощує управління кодом і забезпечує зручний спосіб розширення функціональності програми.

Крім того, спадкування дозволяє використовувати поліморфізм, що означає, що об'єкти похідних класів можуть використовуватися там, де очікується об'єкт базового класу, що покращує гнучкість коду.

Важливо враховувати обмеження спадкування, такі як неможливість успадкування від більше ніж одного класу (одинарне спадкування в C#), а також ризик зайвого зв'язку між класами, який може виникнути при надмірному використанні спадкування.

Отже, спадкування в C# є важливою концепцією об'єктно-орієнтованого програмування, яка дозволяє створювати логічні та структуровані ієрархії класів, сприяючи повторному використанню коду та забезпечуючи гнучкість в розробці програм.

1. **Завдання до виконання.**

### Задача 1.

Створити новий проект із трьома класами та ще одним класом, щоб перевірити логіку вашого коду. Основними класами програми є такі класи:

* Person
* Student
* Professor

Класи Student і Professor є нащадками класу Person.

Клас Student буде включати відкритий Study() метод, який буде писати «I'm studying» на екрані.

Клас Person повинен мати два загальних методів Greet() та SetAge(int age), який встановлює вік людини.

Клас Professor включатиме відкритий Explain() метод, який буде писати «I'm explaining» на екрані.

Також створіть відкритий метод ShowAge() у класі Student, який пише «My age is: x years old» на екрані.

Потрібно створити інший тестовий клас під назвою StudentProfessorTest з методом Main, щоб зробити наступне:

1. Створіть нову людину і змусьте її привітатися.
2. Створіть нового студента, встановіть вік, привітайтеся та покажіть його вік на екрані.
3. Створіть нового професора, встановіть вік, привітайтеся і почніть пояснення.

### Задача 2.

Створіть клас User, в якому будуть наступні protected поля - name (ім'я), age (вік), public методи setName, getName, setAge, getAge.

Створіть клас Worker, який успадковує від класу User і вносить додаткове private поле salary (зарплата), а також методи public getSalary і setSalary.

Створіть об'єкт цього класу 'Іван', вік 25, зарплата 1000. Створіть другий об'єкт цього класу 'Вася', вік 26, зарплата 2000. Знайдіть суму зарплата Івана і Васі.

Створіть клас Student, який успадковує від класу User і вносить додаткові private поля стипендія, курс, а також геттери і сеттери для них.

Створіть клас Driver (Водій), який буде успадковуватися від класу Worker з попередньої задачі. Цей метод повинен вносити такі private поля: водійський стаж, категорія водіння (A, B, C).

### Задача 3 (з індивідуальними варіантами).

Створити клас з полями, відповідно до свого варіанту.

Реалізувати в класі методи:

- конструктор за замовчуванням;

- функції обробки даних (1 і 2), зазначені в індивідуальному завданні;

- функцію формування рядка інформації про об'єкт.

Для створеного класу створити клас-нащадок з додатковим полем, зазначеним в індивідуальному завданні.

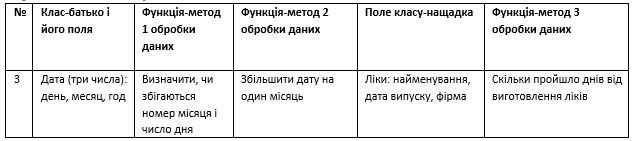
Реалізувати в класі-нащадку методи:

- конструктор;

- функцію обробки даних, зазначену в індивідуальному завданні.

- функцію формування рядки інформації про об'єкт.

Створити проект для демонстрації роботи: введення і виведення інформації про об'єкти: класі-батьку і класі-нащадку.



1. **Лістинг програми.**

class Person

{

    protected int age;

    public void Greet()

    {

        Console.WriteLine("Hello, I'm a person.");

    }

    public void SetAge(int age)

    {

        this.age = age;

    }

}

class Student : Person

{

    public void Study()

    {

        Console.WriteLine("I'm studying.");

    }

    public void ShowAge()

    {

        Console.WriteLine($"My age is: {age} years old");

    }

}

class Professor : Person

{

    public void Explain()

    {

        Console.WriteLine("I'm explaining.");

    }

}

// Клас User

class User

{

    protected string name;

    protected int age;

    // Геттер та сеттер для імені

    public string GetName()

    {

        return name;

    }

    public void SetName(string name)

    {

        this.name = name;

    }

    // Геттер та сеттер для віку

    public int GetAge()

    {

        return age;

    }

    public void SetAge(int age)

    {

        this.age = age;

    }

}

// Клас Worker, що успадковується від User

class Worker : User

{

    private double salary;

    // Геттер та сеттер для зарплати

    public double GetSalary()

    {

        return salary;

    }

    public void SetSalary(double salary)

    {

        this.salary = salary;

    }

}

// Клас Student, що успадковується від User

class Students : User

{

    private double scholarship;

    private int course;

    // Геттери та сеттери для стипендії та курсу

    public double GetScholarship()

    {

        return scholarship;

    }

    public void SetScholarship(double scholarship)

    {

        this.scholarship = scholarship;

    }

    public int GetCourse()

    {

        return course;

    }

    public void SetCourse(int course)

    {

        this.course = course;

    }

}

// Клас Driver, що успадковується від Worker

class Driver : Worker

{

    private int drivingExperience;

    private string drivingCategory;

    // Геттери та сеттери для водійського стажу та категорії водіння

    public int GetDrivingExperience()

    {

        return drivingExperience;

    }

    public void SetDrivingExperience(int drivingExperience)

    {

        this.drivingExperience = drivingExperience;

    }

    public string GetDrivingCategory()

    {

        return drivingCategory;

    }

    public void SetDrivingCategory(string drivingCategory)

    {

        this.drivingCategory = drivingCategory;

    }

}

// Клас-батько "Дата"

class Date

{

    // Змінив рівень захисту полів на protected для забезпечення доступу в класі-нащадку

    public int day;

    public int month;

    public int year;

    // Конструктор для ініціалізації дати

    public Date(int day, int month, int year)

    {

        this.day = day;

        this.month = month;

        this.year = year;

    }

    // Функція-метод 1 обробки даних: визначити, чи збігаються номер місяця і число дня

    public bool IsMonthDayMatch()

    {

        return day == month;

    }

    // Функція-метод 2 обробки даних: збільшити дату на один місяць

    public void IncreaseByOneMonth()

    {

        month++;

        if (month > 12)

        {

            month = 1;

            year++;

        }

    }

    // Геттери для полів day, month і year, щоб мати можливість отримати їх значення в класі-нащадку

    public int GetDay()

    {

        return day;

    }

    public int GetMonth()

    {

        return month;

    }

    public int GetYear()

    {

        return year;

    }

}

// Клас-нащадок "Ліки"

class Medicine : Date

{

    private string name;

    private string manufacturer;

    // Конструктор для ініціалізації ліків

    public Medicine(string name, int day, int month, int year, string manufacturer)

        : base(day, month, year)

    {

        this.name = name;

        this.manufacturer = manufacturer;

    }

    // Функція-метод 3 обробки даних: скільки пройшло днів від виготовлення ліків

    public int DaysSinceManufacture()

    {

        DateTime today = DateTime.Now;

        DateTime manufactureDate = new DateTime(GetYear(), GetMonth(), GetDay());

        TimeSpan duration = today - manufactureDate;

        return duration.Days;

    }

}

class Program

{

    static void Main()

    {

        //task 1

        // Створення нової людини і привітання

        Person person = new Person();

        person.Greet();

        // Створення нового студента, встановлення віку, привітання та показ віку

        Student student = new Student();

        student.SetAge(20);

        student.Greet();

        student.ShowAge();

        student.Study(); // Виклик методу Study

        // Створення нового професора, встановлення віку, привітання та початок пояснення

        Professor professor = new Professor();

        professor.SetAge(45);

        professor.Greet();

        professor.Explain();

        Console.WriteLine();

        //task 2

        // Створення об'єктів класу Worker

        Worker ivan = new Worker();

        ivan.SetName("Іван");

        ivan.SetAge(25);

        ivan.SetSalary(1000);

        Worker vasya = new Worker();

        vasya.SetName("Вася");

        vasya.SetAge(26);

        vasya.SetSalary(2000);

        // Знайдення суми зарплат Івана і Васі

        double totalSalary = ivan.GetSalary() + vasya.GetSalary();

        Console.WriteLine($"Сума зарплат Івана і Васі: {totalSalary}");

        // Створення об'єкту класу Student

        Students student1 = new Students();

        student1.SetName("Студент");

        student1.SetAge(20);

        student1.SetScholarship(500);

        student1.SetCourse(2);

        // Створення об'єкту класу Driver

        Driver driver = new Driver();

        driver.SetName("Водій");

        driver.SetAge(30);

        driver.SetSalary(1500);

        driver.SetDrivingExperience(5);

        driver.SetDrivingCategory("B");

        // Виведення інформації про студента та водія

        Console.WriteLine($"Ім'я студента: {student1.GetName()}");

        Console.WriteLine($"Вік студента: {student1.GetAge()} років");

        Console.WriteLine($"Студент отримує стипендію: {student1.GetScholarship()} грн");

        Console.WriteLine($"Студент на {student1.GetCourse()}-му курсі");

        Console.WriteLine($"Ім'я водія: {driver.GetName()}");

        Console.WriteLine($"Вік водія: {driver.GetAge()} років");

        Console.WriteLine($"Зарплата водія: {driver.GetSalary()} грн");

        Console.WriteLine($"Водій має водійський стаж: {driver.GetDrivingExperience()} років");

        Console.WriteLine($"Водійська категорія водіння: {driver.GetDrivingCategory()}");

        Console.WriteLine();

        // task 3

        // Створення об'єкта класу Date

        Date currentDate = new Date(25, 12, 2023);

        // Функція-метод 1: перевірка, чи збігаються номер місяця і число дня

        Console.WriteLine($"Є відповідність місяця та дня: {currentDate.IsMonthDayMatch()}");

        // Функція-метод 2: збільшення дати на один місяць

        currentDate.IncreaseByOneMonth();

        Console.WriteLine($"Збільшена дата: {currentDate.day}/{currentDate.month}/{currentDate.year}");

        // Створення об'єкта класу Medicine

        Medicine medicine = new Medicine("Aspirin", 10, 11, 2023, "PharmaCompany");

        // Функція-метод 3: визначення, скільки пройшло днів від виготовлення ліків

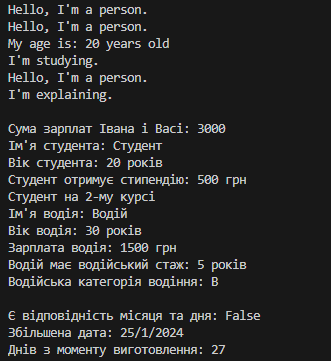
        int daysSinceManufacture = medicine.DaysSinceManufacture();

        Console.WriteLine($"Днів з моменту виготовлення: {daysSinceManufacture}");

    }

}

1. **Результат роботи.**



1. **Посилання на проект.**