Лабораторна робота №11

Розробка програм з ієрархією класів

Мета роботи – придбання практичних навичок щодо розробки об'єктно-орієнтованих програм з ієрархією класів.

Дана лабораторна робота сприяє напрацюванню наступних **компетентностей** у відповідності до Національної рамки кваліфікації:

Знання:

призначення спадкування як фундаментального принципу ООП;

організація доступу до елементів класу при спадкуванні;

механізм використання захищеного доступу до елементів класу за допомогою модифікатора protected;

особливості використання конструкторів при спадкуванні,

Уміння:

для заданої предметної області написати програму, яка:

складається з базового класу і виведеного з нього двох-трьох похідних класів;

здійснює елементарну обробку і виведення інформації про властивості та результати обробки щодо окремих об'єктів успадкованого класу.

Комунікації:

обґрунтування рекомендацій команді учасників проекту щодо доцільності застосування базових та успадкованих класів;

робота в команді над класами, які створюють ієрархію.

Автономність і відповідальність:

прийняття рішення щодо доцільності включення в клас певної ієрархії класів;

самостійне обґрунтування можливих варіантів створювання спадкоємних класів.

Основні положення

Спадкування - один з фундаментальних принципів об'єктно-орієнтованого програмування, оскільки саме завдяки йому можливо створення ієрархічних класифікацій.

Використовуючи спадкування, можливо створити загальний клас, який визначає характеристики відносно властивостей безлічі пов'язаних елементів.

Цей клас потім може бути успадкований іншими, вузькоспеціалізованими класами з додаванням в кожен з них своїх, унікальних особливостей.

У мові C# клас, який успадковується, називається базовим. Клас, який успадковує базовий клас, називається похідним.

Отже, похідний клас - це спеціалізована версія базового класу.

В похідний клас, що успадковує всі змінні, методи і властивості, які визначені в базовому класі, можуть бути додані унікальні елементи.

C# підтримує спадкування, дозволяючи в оголошення класу вбудовувати другий клас. Це реалізується за допомогою завдання базового класу при оголошенні прохідного класу.

Розглянемо клас TwoDShape, в якому визначаються атрибути «узагальненої» двовимірної геометричної фігури (наприклад, квадрата, прямокутника, трикутника і т д.).

```
class TwoDShape
{
        public double width;
        public double height;
        public void showDim ()
        {
        Console.WriteLine ("Ширина і висота дорівнюють" +
        width + "і" + height);
        }
}
```

Клас TwoDShape можна використовувати в якості базового (тобто як стартовий майданчик) для класів, які описують специфічні типи двовимірних об'єктів.

Наприклад, в наступній програмі клас TwoDShape використовується для виведення класу Triangle.

Приклад 1. Проста ієрархія класів (клас двовимірних об'єктів).

```
using System;
// Клас двовимірних об'єктів
class TwoDShape
{
   public double width;
```

```
    public double height;

    public void showDim()
    {
        Console.WriteLine("Ширина і висота дорівнюють  " +
        width + "  і  " + height);
    }
}
// Клас Triangle виводиться з класу TwoDShape
class Triangle : TwoDShape
{
    public string style; // тип трикутника
    // Метод повертає площу трикутника
    public double area()
    {
        return width * height / 2;
    }
    // Відображаємо тип трикутника.
    public void showStyle()
    {
        Console.WriteLine("Трикутник  " + style);
    }
}
class Shapes
{
    public static void Main()
    {
        Triangle t1 = new Triangle();
        Triangle t2 = new Triangle();
        t1.width = 4.0;
        t1.height = 4.0;
        t1.style = " рівнобедрений";
        t2.width = 8.0;
        t2.height = 12.0;
        t2.style = " прямокутний";
        Console.WriteLine("Інформація про t1:  ");
        t1.showStyle();
```

```
        t1.showDim();
        Console.WriteLine("Площа дорівнює  " + t1.area());
        Console.WriteLine();
        Console.WriteLine("Інформація про t2:  ");
        t2.showStyle();
        t2.showDim();
        Console.WriteLine("Площа дорівнює  " + t2.area());
    }
}
```

Результат роботи програми.

```
Інформація про t1:
Трикутник   рівнобедрений
Ширина і висота дорівнюють  4  і  4
Площа дорівнює  8

Інформація про t2:
Трикутник   прямокутний
Ширина і висота дорівнюють  8  і  12
Площа дорівнює  48
```

У класі створюється специфічний тип об'єкту класу TwoDShape, в даному випадку трикутник.

Клас Triangle містить всі елементи класу TwoDShape і, крім того, поле style, метод area () і метод showStyle ().

У змінній style зберігається опис типу трикутника, метод area () обчислює і повертає його площу, а метод showStyle () відображає дані про тип трикутника.

Нижче наведено синтаксис, який використовується в оголошенні класу Triangle, щоб зробити його похідним від класу TwoDShape.

```
class Triangle: TwoDShape
        {
        // тіло класу
        }
```

Таким чином, якщо один клас успадковує інший, то ім'я базового класу вказується після імені похідного, причому імена класів розділяються двокрапкою.

Оскільки клас Triangle включає всі елементи базового класу, TwoDShape, він може звертатися до елементів width і height всередині методу area ().

Крім того, всередині методу Main () об'єкти t1 і t2 можуть прямо посилатися на елементи width і height, як би вони були частиною класу Triangle.

Незважаючи на те що клас TwoDShape є базовим для класу Triangle, це абсолютно незалежний і автономний клас. Те, що його використовують як базовий похідний клас (класи), не означає неможливість використання його самого.

Наприклад, наступний фрагмент коду абсолютно легальний:

```
TwoDShape shape = new TwoDShape ();
shape.width = 10;
shape.height = 20;
shape.showDim ();
```

Загальна форма оголошення класу, який успадковує базовий клас, має такий вигляд:

```
class ім’я__похідного_ класу : ім’я_базового_класу
{
// Тіло класу
}
```

Для створюваного похідного класу можна вказати тільки один базовий клас.

C# не підтримує спадкування декількох базових класів в одному похідному класі. І звичайно ж, жоден клас не може бути базовим (ні прямо, ні опосередковано) для самого себе.

У кожному похідному класі можна потім точно «налаштувати» власну класіфікацію.

Ось, наприклад, як з базового класу TwoDShape можна вивести похідний клас, який інкапсулює прямокутники:

```
class Rectangle: TwoDShape
{
    // Метод повертає значення true, якщо прямокутник є квадратом.
    public bool isSquare ()
    {
    if (width == height) return true;
    return false;
    }
// Метод повертає значення площі прямокутника.
    public double area ()
    {
    return width * height;
    }
}
```

Клас Rectangle включає клас TwoDShape і додає метод isSquare (),який визначає, чи є прямокутник квадратом, і метод area (), що обчислює площу прямокутника.

Доступ до елементів класу та успадкування.

Щоб запобігти несанкціоноване використання та внесення змін елементи класу часто оголошуються закритими.

Спадкування класу не скасовує обмеження, пов'язані з закритим доступом. Таким чином, незважаючи на те, що похідний клас включає всі елементи базового класу, він не може отримати доступ до тих з них, які оголошені закритими.

Наприклад, як показано в наступному коді, якщо елементи width і height являються private-елементами в класі TwoDShape, то клас Triangle не зможе отримати до них доступ.

Приклад 2. Доступ до закритих елементами не успадковується.

```
// Цей приклад не компілюється
using System;
// Клас двовимірних об'єктів
class TwoDShape
{
```

```
        double width;      //тепер це private-елемент
        double height;     // тепер це private-елемент
        public void showDim ()
        {
        Console.WriteLine ("Ширина і висота дорівнюють" +
        width + "and" + height);
        }
}
// Клас Triangle виводиться з класу TwoDShape
class Triangle: TwoDShape
        {
        public string style; // тип прямокутника
        // Метод повертає значення площі трикутника.
        public double area ()
        {
                return width * height / 2; // помилка, не можна отримати
        прямий доступ до закритого елемента
        }
        // Відображаємо тип трикутника.
        public void showStyle ()
        {
        Console.WriteLine ("Triangle is" + style);
        }
}
```

Повідомлення про помилку.

'TwoDShape.width' is inaccessible due to its protection level

Клас Triangle не компілюється, оскільки посилання на елементи width і height всередині методу area () викликає помилку порушення прав доступу.

Оскільки width і height - закриті елементи, вони доступні тільки для елементів їх власного класу. На похідні класи ця доступність не поширюється.

Закритий елемент класу залишається закритим в рамках цього класу. До нього не можна отримати доступ з коду, визначеного поза цього класу, включаючи похідні класи.

На перший погляд може здатися, що неможливість доступу до закритих елементів базового класу з боку похідного - серйозне обмеження. Однак це не так, оскільки в C# передбачені можливості вирішення цієї проблеми.

Одна з них - protected-елементи, друга можливість - використання відкритих властивостей і методів, що дозволяють отримати доступ до закритих даних.

Нижче наведена нова версія класу TwoDShape, в якої колишні елементи width і height стали властивостями.

Приклад 3. Використання властивостей для запису і читання закритих елементів класу.

```
using System;
// Клас двовимірних об'єктів.
class TwoDShape
{
   double pri_width;     // тепер це private-елемент
   double pri_height;    // тепер це private-елемент
   // Властивості width і height.
   public double width
   {
      get { return pri_width; }
      set { pri_width = value; }
   }
   public double height
   {
      get { return pri_height; }
      set { pri_height = value; }
   }
   public void showDim()
   {
      Console.WriteLine("Ширина і висота дорівнюють " +
      width + " і " + height);
   }
}
// Клас трикутника - похідний від класу TwoDShape
class Triangle : TwoDShape
{
```

```
    public string style; // тип трикутника.
    // Метод повертає значення площі трикутника
    public double area()
    {
        return width * height / 2;
    }
    // Відображаємо тип трикутника
    public void showStyle()
    {
        Console.WriteLine("Трикутник  " + style);
    }
}
class Shapes2
{
    public static void Main()
    {
        Triangle t1 = new Triangle();
        Triangle t2 = new Triangle();
        t1.width = 4.0;
        t1.height = 4.0;
        t1.style = " рівнобедрений";
        t2.width = 8.0;
        t2.height = 12.0;
        t2.style = " прямокутний";
        Console.WriteLine("Інформація про t1  :");
        t1.showStyle();
        t1.showDim();
        Console.WriteLine("Площа дорівнює  " + t1.area());
        Console.WriteLine();
        Console.WriteLine("Інформація про t2  :");
        t2.showStyle();
        t2.showDim();
        Console.WriteLine("Площа дорівнює  " + t2.area());
    }
}
```

Результат роботи програми.

Інформація про t1 :

Трикутник рівнобедрений
Ширина і висота дорівнюють 4 ? 4
Площа дорівнює 8

Інформація про t2 :
Трикутник прямокутний
Ширина і висота дорівнюють 8 ? 12
Площа дорівнює 48

Використання захищеного доступу.

Закритий елемент базового класу недоступний для похідного класу. Здавалося б, це означає, що, якщо похідний клас повинен мати доступ до елементу базового класу, його потрібно зробити відкритим. При цьому доведеться зміриться з тим, що відкритий елемент буде доступним для будь-якого іншого коду, що іноді небажано.

Однак, таких ситуацій можна уникнути, оскільки C# дозволяє створювати захищені елементи.

Захищеним є елемент, який відкритий для своєї ієрархії класів, але закритий поза цією ієрархії.

Захищений елемент створюється за допомогою модифікатора доступу protected. При оголошенні protected- елементу він по суті є закритим, але з одним винятком. Виняток набирає чинності, коли захищений елемент успадковується. У цьому випадку захищений елемент базового класу стає захищеним елементом похідного класу, а отже, і доступним для цього класу.

Таким чином, використовуючи модифікатор доступу protected, можна створювати закриті (для «зовнішнього світу») елементи класу, але разом з тим вони будуть успадковуватися з можливістю доступу з боку похідних класів.

Приклад 4. Демонстрація використання захищених елементів класу.

```
using System;
class B
{
   protected int i, j;       // закрито усередині класу B, але
                             // доступно для класу D
   public void set(int a, int b)
```

```
    {
        i = a;
        j = b;
    }
    public void show()
    {
        Console.WriteLine(i + "  " + j);
    }
}
// Клас D отримує доступ до елементів i and j класу B
class D : B
{
    int k; // закритий елемент.
    public void setk()
    {
        k = i * j;
    }
    public void showk()
    {
        Console.WriteLine(k);
    }
}
class ProtectedDemo
{
    public static void Main()
    {
        D ob = new D();
        ob.set(2, 3);         // OK, так як D "бачить" B-елементи i та j
        ob.show();            // OK, так як D "бачить" B-елементи i та j
        ob.setk();            // OK, так як це частина самого класу D
        ob.showk();           // OK, так як це частина самого класу D
    }
}
```

Результат виконання програми.

```
2  3
6
```

Оскільки в цьому прикладі клас B успадковується класом D і елементи «i» та «j» були об’явлені захищеними в класі B (тобто з використанням модифікатора доступу protected), метод setk () може отримати до них доступ. Якби елементи «i» та «j» були оголошені в класі B закритими, клас D не мав би до них права доступу, і програму не була б скомпільована.

Подібно модифікаторам public і private модифікатор protected залишається зі своїм елементом незалежно від реалізованої кількості рівнів успадкування.

Таким чином, при використанні похідного класу в якості базового для створення другого похідного класу, будь який захищений елемент вихідного базового класу, який успадковується першим похідним класом, також успадковується в статусі захищеного і другим похідним класом.

Конструктори і спадкування.

В ієрархії класів як базові, так і похідні класи можуть мати власні конструктори. При цьому виникає важливе питання: який конструктор відповідає за створення об'єкта похідного класу? Конструктор базової або конструктор похідного класу, або обидва одночасно?

Відповідь така: конструктор базового класу створює частину об'єкта, відповідну базового класу, а конструктор похідного класу - частину об'єкта, яка відповідна похідному класу.

У попередніх прикладах класи спиралися на конструктори за умовчанням, які створювалися автоматично засобами C#, і тому ми не стикалися з подібною проблемою. Але на практиці більшість класів має конструктори.

Якщо конструктор визначається тільки в похідному класі, в цьому разі просто створюється об'єкт похідного класу. Частина об'єкта, яка відповідає базовому класу, створюється автоматично за допомогою конструктора за замовчуванням.

Нижче наведена перероблена версія класу Triangle, в якій визначається конструктор. Тут елемент style оголошений private-елементом, оскільки тепер він встановлюється конструктором.

Приклад 5. Додавання конструктора в клас Triangle.

```
using System;
class TwoDShape
```

```
{
    double pri_width;     // закритий елемент
    double pri_height;    // закритий елемент
    // Властивості width і height.
    public double width
    {
        get { return pri_width; }
        set { pri_width = value; }
    }
    public double height
    {
        get { return pri_height; }
        set { pri_height = value; }
    }
    public void showDim()
    {
        Console.WriteLine("Ширина і висота дорівнює   " +
        width + "  і  " + height);
    }
}
// Клас трикутника - похідний від класу TwoDShape
class Triangle : TwoDShape
{
    string style; // private
    // Конструктор
    public Triangle(string s, double w, double h)
    {
        width = w;   // ініціалізує елемент базового класу
        height = h;  // ініціалізує елемент базового класу
        style = s; //   ініціалізує елемент свого класу
    }
    // Метод повертає значення площі трикутника
    public double area()
    {
        return width * height / 2;
    }
    // Відображаємо тип трикутника
```

```
    public void showStyle()
    {
        Console.WriteLine(" Трикутник   " + style);
    }
}
class Shapes3
{
    public static void Main()
    {
        Triangle t1 = new Triangle(" рівнобедрений   ", 4.0, 4.0);
        Triangle t2 = new Triangle(" прямокутний   ", 8.0, 12.0);
        Console.WriteLine("Інформація про t1:    ");
        t1.showStyle();
        t1.showDim();
        Console.WriteLine("Площа дорівнює    " + t1.area());
        Console.WriteLine();
        Console.WriteLine("Інформація про t2:    ");
        t2.showStyle();
        t2.showDim();
        Console.WriteLine("Площа дорівнює    " + t2.area());
    }
}
```

Результат роботи програми.

```
Інформація про t1:
 Трикутник    рівнобедрений
Ширина і висота дорівнює   4  і  4
Площа дорівнює    8

Інформація про t2:
 Трикутник    прямокутний
Ширина і висота дорівнює   8  і  12
Площа дорівнює    48
```

У цьому прикладі конструктор класу Triangle ініціалізує успадковані їм елементи класу TwoDShape, а також власне поле style.

Якщо конструктори визначені і в базовому, і в похідному класі, процес створення об'єктів дещо ускладняється, оскільки повинні виконатися конструктори обох класів.

У цьому випадку необхідно використовувати ще одне ключове слово C# base, яке має два призначення:

• викликати конструктор базового класу;

• отримати доступ до елементу базового класу, який прихований за елементом похідного класу.

Порядок виконання лабораторної роботи

Загальна частина.

1. Набрати, відкомпілювати і запустити на виконання приклади програм, які були наведені в розділі «Основні положення» даної лабораторної роботи.

2. Проекспериментуйте з програмами:

Зверніть увагу на особливості синтаксису визначення похідних клас-сов.

Індивідуальна частина.

Для заданої предметної області написати програму, яка:

складається з розробленого базового класу і виведеного з нього двох-трьох похідних класів;

здійснює елементарну обробку і виведення інформації про властивості та результати обробки для окремих об'єктів успадкованого класу.

Як прототип можна використовувати програму з прикладу 5.

Зміст звіту

1. Титульний лист.

2. Цілі лабораторного заняття і вказівка, які навички та вміння передбачається отримати в результаті його виконання.

3. Тексти налагоджених програм загальної частини лабораторного заняття з необхідними коментарями і результатом виконання.

4. Текст налагодженої програми з результатом виконання контрольних прикладів індивідуального завдання.

6. Висновки.

Контрольні питання

1. Коли доцільно використовувати ієрархію класів?

2. Як здійснюється організація доступу до елементів класу при спадкуванні?

3. Опішить механізм використання захищеного доступу до елементів класу за допомогою модифікатора protected.

4. Наведіть особливості використання конструкторів при спадкуванні.

5. Що таке «властивості» екземпляру об’єкта? Яким чином вони задається в класі. Опішить синтаксис завдання.