

Лабораторна робота №7

Успадкування і віртуальні функції

Мета. Одержати практичні навички створення ієрархії класів і використання віртуальних методів класу.

Основний зміст роботи

Написати програму, в якій створюється ієрархія класів. Показати використання віртуальних функцій.

Короткі теоретичні відомості

Фундаментальною основою людського мислення є пошук, виявлення й побудова взаємин між різними концепціями. Щоб збагнути хитросплетення відносин між речами і явищами, використовують ієрархічні побудови, матриці, мережі та інші засоби візуалізації. Щоб краще виразити суть відносин між об'єктами, в C# використовується ієрархічна система успадкування.

Ієрархія й успадкування

Коли ми говоримо про собаку, як представника класу ссавців, ми маємо на увазі, що вона успадковує всі ознаки, загальні для класу ссавців.

Оскільки собака - ссавець, можна припустити, що це рухливий вид тварин, що дихають повітрям.

Всі ссавці за визначенням рухаються й дихають повітрям.

Якщо визначити якийсь об'єкт як собаку, це додасть до оголошення здатність виляти хвостом, бігати по будинку й гавкати.

У свою чергу, собак можна розділити на службових, спортивних і мисливських. Потім можна піти далі й описати породу собаки: спанієль, лабрадор і т.д.

Таким чином, ми можемо сказати, наприклад, що фокстер'єр - це порода мисливських собак, у якій представлені всі ознаки, загальні для собак взагалі, а також всі ознаки, загальні для ссавців і т.д., включаючи ознаки всіх таксонів, до яких належить фокстер'єр.

В C# ієрархія реалізована в концепції класів, де один клас може походити, або успадковуватися від класу більше високого рівня. У спадкуванні класів реалізуються принципи їхньої ієрархічної підпорядкованості.

Припустимо, ми робимо новий клас Dog (Собака) від класу Mammal (Ссавець). Інакше кажучи, клас Mammal є базовим для класу Dog. Точно так само, як опис виду собака несе в собі ознаки, що деталізують опис ссавців у цілому, так і клас Dog містить ряд методів і даних, що доповнюють методи й дані, які представлені в класі Mammal.

Як правило, з базовим класом зв'язано кілька похідних класів. Оскільки собаки, кішки й коні є представниками ссавців, то з погляду C# можна сказати, що всі ці класи є похідними від класу `Mammal`.

Синтаксис опису успадкування класів

Для створення нового похідного класу використовується ключове слово `class`, після якого вказується ім'я нового класу, двокрапка, тип оголошення класу (`public` або який-небудь інший), а потім ім'я базового класу, як у наступному прикладі:

```
class Dog : public Mammal
```

Клас, з якого буде походити новий клас, повинен бути оголошений раніше, інакше компілятор покаже повідомлення про помилку.

Приклад успадкування класу `Dog` від класу `Mammal` показаний у програмі 1.

Програма 1. Просте спадкування

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Example1
{
    public enum BREED { GOLDEN, CAIRN, DANDIE, SHETLAND, OOBERMAN, LAB };
    //class Mammal
    public class Mammal
    {
        protected int itsAge;
        protected int itsWeight;

        // Конструктори
        public Mammal() { }
        // Методи доступу до даних
        public int Age
        {
            get;
            set;
        }
        public int Weight
        {
            get;
            set;
        }
    }
}
```

```

        // Інші методи
        void Speak() { }
        void Sleep() { }
    };
// class Dog
    public class Dog : Mammal
    {
        protected BREED itsBreed;
        // Конструктори
        public Dog() { }
        // Методи доступу до даних
        public BREED Breed
        {
            get;
            set;
        }
        // Інші методи
        void WagTail() { }
        void BegForFood() { }
    }
}

```

Програма нічого не виводить на екран, тому що поки містить тільки оголошення й установки класів. Ніяких функцій ця програма поки не виконує.

Зверніть увагу, що клас Mammal не походить ні від якого іншого класу, хоча в реальному житті можна сказати, що клас ссавців походить від класу тварин. Але в ООП відображається не увесь навколишній світ, а лише модель деякої його частини.

Програма розпочинається із класу Mammal, тому деякі змінні члени, які необхідні для роботи базового класу, повинні бути представлені в оголошенні цього класу. Наприклад, всі тварини незалежно від виду й породи мають вік і вагу. Якби клас Mammal походив від класу Animals, то можна було б очікувати, що він успадкує ці атрибути. При цьому атрибути базового класу стають атрибутами похідного класу.

Щоб полегшити роботу із програмою й обмежити її складність, у класі Mammal представлені тільки: три властивості та чотири методи функції Speak(), Sleep(), WagTail(), BegForFood().

Клас Dog успадковується із класу Mammal. Всі об'єкти класу Dog будуть мати три змінн-члени: itsAge, itsWeight і itsBreed. Зверніть увагу, що в оголошенні класу Dog не зазначені змінні itsAge і itsWeight. Об'єкти класу Dog успадкували ці змінні із класу Mammal разом з методами, оголошеними в класі Mammal, за винятком конструктора.

Закритий або захищений

В програмі 1 використовується нове ключове слово protected.

Члени класу, оголошені як `private`, недоступні для спадкування.

Звичайно, можна було в попередній програмі визначити змінні-члени `itsAge` і `itsWeight` як `public`, але це небажано, оскільки прямий доступ до цих змінних одержали б всі інші класи програми.

Нашу мету можна сформулювати в такий спосіб: зробити змінні-член видимі для цього класу й для всіх класів, похідних від нього. Саме такими є захищені дані, обумовлені ключовим словом `protected`. Захищені дані доступні для всіх похідних класів, але недоступні для всіх зовнішніх класів.

Узагальнимо: існує три специфікатори доступу - `public`, `protected` і `private`. Якщо у функцію передаються об'єкти класу, то вона може використати дані всіх змінних-членів і функцій-членів, оголошених із специфікатором `public`.

Функція-член класу, крім того, може використати всі закриті дані цього класу (оголошені як `private`) і захищені дані будь-якого іншого класу, похідними від цього класу (оголошені як `protected`).

Так, у нашому прикладі функція `WagTail()` може використовувати значення закритої змінної `itsBreed` і всі змінні класу `Mammal`, оголошені як `public` і `protected`.

Навіть якби клас `Dog` був утворений не від класу `Mammal` безпосередньо, а від якого-небудь проміжного класу (наприклад, `DomesticAnimals`), однаково із класу `Dog` зберігся б доступ до захищених даних класу `Mammal`, правда тільки в тому випадку, якщо клас `Dog` і всі проміжні класи оголошувалися як `public`.

У програмі 2 показане створення об'єкта в класі `Dog` з доступом до всіх даних і функцій цього типу.

Програма 2. Використання успадкованих об'єктів

//----- Файл `Mammal.cs`

```
using System;
namespace Example1
{
    public enum BREED { GOLDEN, CAIRN, DANDIE, SHETLAND, OOBERMAN, LAB };
    public class Mammal
    {
        protected int age;
        protected int weight;
        // Конструктори
        public Mammal()
        {
            age = 2;
            weight = 5;
        }
        // Методи доступу до даних
```

```

    public int Age
    {
        get
        {
            return age;
        }
        set
        {
            age = value;
        }
    }
    public int Weight
    {
        get;
        set;
    }
    // Інші методи
    public void Speak() { Console.WriteLine("Mammal sound!\n"); }
    public void Sleep() { Console.WriteLine("Shhh. I'm sleeping.\n"); }
};

public class Dog : Mammal
{
    //protected BREED breed;
    // Конструктори
    public Dog() { }
    // Методи доступу до даних
    public BREED Breed
    {
        get;
        set;
    }
    // Інші методи
    public void WagTail() { Console.WriteLine("Tail wagging...\n"); }
    public void BegForFood() { Console.WriteLine("Begging for food...\n"); }
}

}

//----- Файл Program.cs
using System;
namespace Example1
{

```

```

class Program
{
    static void Main(string[] args)
    {
        Dog fido = new Dog();
        fido.Speak();
        fido.WagTail();
        Console.WriteLine(string.Format("Fido is {0} years old\n",
                                         fido.Age));

        Console.ReadKey();
    }
}

```

Mammal sound!

Tail wagging...

Fido is 2 years old

Спочатку оголошується клас Mammal, потім клас Dog як похідний від класу Mammal. У результаті об'єкту fido цього класу доступна як функція похідного класу WagTai(), так і функції базового класу Speak() і Sleep().

Конструктори й деструктори

Об'єкти класу Dog одночасно є об'єктами класу Mammal. У цьому суть ієрархічних відносин між класами. Коли в класі Dog створюється об'єкт Fido, то для цього із класу Mammal викликається базовий конструктор, названий першим. Потім викликається конструктор класу Dog, що завершує створення об'єкта.

Оскільки об'єкт Fido не містить ніяких параметрів, в обох випадках викликається конструктор, заданий за замовчуванням.

Об'єкт Fido не існує доти, поки повністю не буде завершено його створення з використанням обох конструкторів класу Mammal і класу Dog.

При видаленні об'єкта Fido з пам'яті комп'ютера спочатку викликається деструктор класу Dog, а потім деструктор класу Mammal. Кожний деструктор видаляє ту частину об'єкта, що була створена відповідним конструктором похідного або базового класів.

Віртуальні методи

Об'єкти класу Dog одночасно є об'єктами класу Mammal. Під цим малося на увазі, що об'єкти класу Dog успадковують всі атрибути (дані) і можливості (методи)

базового класу. Але в мові C# принципи ієрархічної побудови класів несуть у собі ще більш глибинний зміст.

Поліморфізм в C# допускає присвоєння посиланням типу базового класу посилання на об'єкти похідних класів, як у наступному прикладі:

```
Mammal mammal = new Dog();
```

Даний вираз створює в області динамічної пам'яті новий об'єкт класу Dog і повертає посилання на цей об'єкт, яке має тип посилання на об'єкт класу Mammal. Це цілком логічно, тому що собака - представник ссавців.

Потім цей покажчик можна використати для виклику будь-якого методу класу Mammal.

Причому якщо метод був заміщений, скажемо, у класі Dog, то при звертанні до методу через покажчик буде викликатися саме варіант, зазначений у даному похідному класі. У цьому суть використання віртуальних функцій.

Приклад 3 показує, як працює віртуальна функція й що відбувається з невіртуальною функцією.

Приклад 3. Використання віртуальних методів

```
using System;
namespace Example1
{
    public enum BREED { GOLDEN, CAIRN, DANDIE, SHETLAND, OOBERMAN, LAB };
    public class Mammal
    {
        protected int age;
        protected int weight;

        // Конструктори
        public Mammal()
        {
            age = 1;
            weight = 5;
        }

        // Методи доступу до даних
        public int Age
        {
            get
            {
                return age;
            }
            set
            {

```

```

        age = value;
    }
}

public int Weight
{
    get;
    set;
}

// Інші методи
virtual public void Speak() { Console.WriteLine("Mammal sound!\n"); }
public void Sleep() { Console.WriteLine("Shhh. I'm sleeping.\n"); }
public void Move() { Console.WriteLine("Mammal move one step\n"); }
}

public class Dog : Mammal
{
    //protected BREED breed;
    // Конструктори
    public Dog() { }
    // Методи доступу до даних
    public BREED Breed
    {
        get;
        set;
    }
    // Інші методи
    public void WagTail() { Console.WriteLine("Tail wagging...\n"); }
    public void BegForFood() { Console.WriteLine("Begging for food...\n"); }
    new public void Speak(){ Console.WriteLine("Woof!\n"); }
    public void Move(){ Console.WriteLine("Dog moves 5 steps...\n"); }
}

}

using System;
namespace Example1
{
    class Program
    {
        static void Main(string[] args)
        {
            Dog fido = new Dog();
            fido.Speak();
        }
    }
}

```

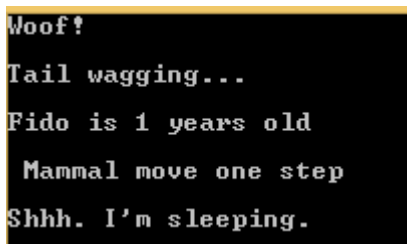


```

        fido.WagTail();
        Console.WriteLine(string.Format("Fido is {0} years old\n",
fido.Age));

        Console.ReadKey();
        Mammal fido1 = new Dog();
        fido1.Move();
        fido1.Sleep();
        Console.ReadKey();
    }
}
}

```



```

Woof!
Tail wagging...
Fido is 1 years old
Mammal move one step
Shhh. I'm sleeping.

```

Метод `Speak()` класу `Mammal` оголошується як віртуальний. Передбачається, що даний клас повинен бути базовим для інших класів. Ймовірно також, що дана функція може бути заміщена в похідних класах.

Потім посилання типу `Mammal fido1`, але йому привласнюється адреса нового об'єкта похідного класу `Dog`. Оскільки собака є ссавцем, це цілком логічно. Даний покажчик потім використовується для виклику функції `Move()`. Оскільки `fido1` відомий компіляторові як покажчик класу `Mammal`, результат виходить таким же, як при звичайному виклику методу `Move()` з об'єкта класу `Mammal`.

У наступних рядках через посилання `fido1` робиться звертання до методу `Speak()`. У цьому випадку метод `Speak()` оголошений як віртуальний, тому викликається варіант функції `Speak()`, заміщений у класі `Dog`.

Хоча компілятор знає, що посилання `fido1` належить класу `Mammal`, проте відбувається виклик версії функції, оголошеної в іншому похідному класі.

Приклад 4. В похідному класі для методу `Speak` використайте такий опис

```

override public void Speak(){ Console.WriteLine("Woof!\n"); }

```

Проаналізуйте результат.

Методичні вказівки щодо виконання роботи

1. Для визначення ієрархії класів зв'язати відношенням спадкування класи, приведені в додатку (для заданого варіанта). З перерахованих класів вибрати один, який буде стояти на чолі ієрархії. Це абстрактний клас.

2. Визначити у класах усі необхідні конструктори і деструктор.

3. Поля-дані дані класу потрібно специфікувати як protected.

4. Визначити у класах усі необхідні властивості та методи.

5. Хоча б один метод оголосити як віртуальний в базовому та перевизначити його в похідних.

6. Визначення класів, їхню реалізацію та демонстраційну програму помістити в окремі файли.

7. Завдання реалізувати за допомогою шаблону Windows Forms.

Зміст звіту

1. Титульний лист: назва дисципліни, номер і найменування роботи, прізвище, ім'я, по батькові студента, дата виконання.

2. Постановка задачі. Варто дати конкретну постановку, тобто вказати які класи повинні бути реалізовані, які повинні бути в них конструктори, компоненти-функції і т.д.

3. Ієрархія класів у виді графа.

4. Визначення класів користувача з коментарями.

5. Реалізація конструкторів з параметрами і деструктора.

6. Реалізація методів для додавання об'єктів в список .

7. Реалізація методів для перегляду списку.

8. Лістинг демонстраційної програми.

9. Пояснення необхідності віртуальних функцій. Варто показати, які результати будуть у випадку віртуальних і не віртуальних функцій.

Варіанти завдань

Перелік класів.

1. Студент, викладач, персона, зав. кафедрою.

2. Службовець, персона, робітник, інженер.

3. Робітник, кадри, інженер, адміністрація.

4. Деталь, механізм, виріб, вузол.

5. Організація, страхова компанія, суднобудівна компанія, завод.

6. Журнал, книга, друковане видання, підручник.

7. Тест, іспит, випускний іспит, випробовування.

8. Місце, область, місто, мегаполіс.
9. Іграшка, продукт, товар, молочний продукт.
10. Квитанція, накладна, документ, чек.
11. Автомобіль, потяг, транспортний засіб, експрес.
12. Двигун, двигун внутрішнього згоряння, дизель, турбореактивний двигун.
13. Республіка, монархія, королівство, держава.
14. Ссавці, парнокопитні, птаха, тварина.
15. Корабель, пароплав, вітрильник, корвет.