СПАДКУВАННЯ

Спадкування - це властивість системи, що дозволяє описати новий клас на основі вже існуючого з частково або повністю позичає функціональністю. Клас, від якого виробляється спадкування, називається базовим або батьківським. Новий клас - нащадком, спадкоємцем або похідним класом.

Спадкування є одним з трьох основних принципів об'єктно-орієнтованого програмування, Оскільки воно допускає створення ієрархічних класифікацій. Завдяки спадкоємства можна створити загальний клас, в якому визначаються характерні особливості, властиві безлічі пов'язаних елементів. Від цього класу можуть потім наслідувати інші, більш конкретні класи, додаючи в нього свої індивідуальні особливості.

У мові C # клас, який успадковується, називається базовим, а клас, який успадковує, - похідним.

Отже, похідний клас є спеціалізованим варіант базового класу. Він успадковує всі змінні, методи, властивості і індексатори, що визначаються в базовому класі, додаючи до них свої власні елементи.

основи спадкування

• Підтримка успадкування в C # полягає в тому, що в оголошення одного класу дозволяється вводити інший клас. Для цього при оголошенні похідного класу вказується базовий клас.

• При описі класу ім'я його предка записується в заго-спритно класу після двокрапки:

[Атрибути] [специфікатор] class <ім'я_класу> [: предки] <тіло класу>

• Якщо ім'я предка не вказано, предком вважається базовий клас всієї ієрархії System.Object.

• Елементи базового класу, певні як private, в похідному класі недоступні. Поля, певні зі специфікатором protected, будуть доступні методам всіх класів, похідних від базового.

Приклад. Створимо клас для визначення двомірних фігур типу, прямокутник, квадрат, ромб і.т.д. з параметрами висота і ширина і методом для виведення цих значень. Опишемо похідний клас -треугольніка, що містить методи розрахунку площі і виведення типу трикутника (прямокутний, рівнобедрений і т.п.).
діаграма класів

фрагмент коду

У класі Triangle створюється особливий тип об'єкту класу Figura2D (в даному випадку - трикутник). Крім того, в клас Triangle входять всі члени класу Figura2D, до яких, зокрема, додаються методи Pl () і ShowStyle (). Так, опис типу трикутника зберігається в змінної Style, метод Pl () розраховує і повертає площа трикутника, а метод ShowStyle () відображає тип трикутника.
Зверніть увагу на синтаксис, який використовується в класі Triangle для наслідування
класу TwoDShape.
class Triangle: TwoDShape {
Цей синтаксис може бути узагальнений. Всякий раз, коли один клас успадковує від іншого, після назви класу вказується ім'я похідного класу, виділення двокрапкою. У C # синтаксис успадкування класу дивно простий і зручний у використанні.
В клас Triangle входять всі члени його базового класу Figura2D, і тому
в ньому змінні W і H доступні для методу Pl (). Крім того, об'єкти t1 і t2 в методі Main () можуть звертатися безпосередньо до змінних W і
H, як ніби вони є членами класу Triangle. На рис. схематично показано, яким чином клас Figura2D вводиться в клас Triangle.

Незважаючи на те що клас Figura2D є базовим для класу Triangle, в той

Водночас він є абсолютно незалежний і самодостатній клас. Якщо клас слугує базовим для похідного класу, то це зовсім не означає, що він не може бути використаний самостійно.

Для будь-якого похідного класу можна вказати тільки один базовий клас. У C # не передбачено спадкування декількох базових класів в одному похідному класі. (В цьому відношенні C # відрізняється від C ++, де допускається спадкування декількох
базових класів. Дана обставина слід брати до уваги при перенесенні коду C ++ в C #.) Проте можна створити ієрархію спадкування, в якій похідний клас стає базовим для іншого похідного класу. (Зрозуміло, жоден з класів не може бути базовим для самого себе як безпосередньо, так і опосередковано.) Але в будь-якому випадку похідний клас успадковує всі члени свого базового класу, в тому числі змінні екземпляра, методи, властивості і індексатори.

Головна перевага успадкування полягає в наступному: як тільки буде створено базовий клас, в якому визначено загальні для безлічі об'єктів атрибути, він може бути використаний для створення будь-якого числа більш конкретних похідних класів. А в кожному похідному класі може бути точно вибудувана своя власна класифікація.

Приклад. Додамо ще один клас спадкоємець класу Figura2D і інкапсулює прямокутники, що містить два методу - визначення площі і перевірки на квадрат.

код

Доступ до членів класу і наслідування

Члени класу часто оголошуються закритими, щоб виключити їх несанкціоноване або незаконне використання. Але спадкування класу не скасовує обмеження, що накладаються на доступ до закритих членів класу. Тому якщо в похідних ий клас і

входять всі члени його базового класу, в ньому все одно виявляються недоступними ті члени базового класу, які є закритимі.Закритий член класу залишається закритим в своєму класі. Він недоступний коду за межами свого класу, включаючи і похідні класси.Для подолання даного обмеження в C # передбачені різні способи.Один з них полягає у використанні захищених (protected) членів класу, розглянутих далі, а другий - в застосуванні відкритих властивостей длядоступа до закритим данним.Організація захищеного доступаКак пояснювалося вище, відкритий член базового класу недоступний для похідного класу. З цього можна припустити, що для доступу до деякого члену базового класу з похідного класу цей член необхідно зробити відкритим. Але якщо зробити член класу відкритим, то він стане доступним для всього коду, що далеко не завжди бажано. Правда, згадане припущення вірне лише частково, оскільки в C # допускається створення захищеного члена класу. Захищений член є відкритим в межах ієрархії класів, але закритим за межами цієї ієрархії.Защіщенний член створюється за допомогою модифікатора доступу protected. Якщо член класу оголошується як protected, він стає закритим, але за винятком одного випадку, коли захищений член успадковується. В цьому випадку захищений член базового класу стає захищеним членом похідного класу, а значить, доступним для похідного класу. Таким чином, використовуючи модифікатор доступу protected, можна створити члени класу, які є закритими для свого класу, але все ж успадкованими і доступними для похідного класса.Ніже наведено простий приклад застосування модифікатора доступу protected.// Продемонструвати застосування модифікатора доступу protected.using System; class B {protected int i, j; // Члени, закриті для класу B, // але доступні для класу Dpublic void Set (int a, int b) {i = a; j = b;} public void Show () {Console.WriteLine (i + "" + j);}} class D: B {int k; // Закритий член // члени i і j класу B доступні для класу Dpublic void Setk () {k = i * j;} public void Showk () {Console.WriteLine (k);}} class ProtectedDemo {static void Main ( ) {D ob = new D (); ob.Set (2, 3); // Допустимо, оскільки є для класу Dob.Show (); // Допустимо, оскільки є для класу Dob.Setk (); // Допустимо, оскільки входить в клас Dob.Showk (); //

Допустимо, оскільки входить в клас D}} В даному прикладі клас B успадковується класом D, а його члени i і j оголошені як protected, і тому вони доступні для методу Setk (). Якби члени i і j класу B були оголошені як private, то вони були б недоступними для класу D, і наведений вище код не можна було б скомпілювать.Аналогічно станом public і private, стан protected зберігається зачленом класу незалежно від кількості рівнів спадкування. Тому коли похідний клас використовується в якості базового для іншого похідного класу, будь-який захищений член вихідного базового класу, успадковані першим похідним класом, успадковується як захищений і другим похідним классом.Несмотря на всю свою корисність, захищений доступ придатний далеко не для всіх ситуацій. Модифікатор доступу protected слід застосовувати в тому випадку, якщо потрібно створити член класу, доступний для всієї ієрархії класів, але для решти коду він повинен бути закритим. А для управління доступом до значення члена класу краще скористатися свойством.Конструктори і спадкування ієрархії класів допускається, щоб у базових і похідних класів були свої власні конструктори. У зв'язку з цим виникає наступне резонне питання: який конструктор відповідає за побудову об'єкта похідного класу: конструктор базового класу, конструктор похідного класу або ж обидва? На це питання можна відповісти так: конструктор базового класу конструює базову частину об'єкта, а конструктор похідного класу - похідну частину цього об'єкта. І в цьому є своя логіка, оскільки базового класу невідомі і недоступні будь-які елементи похідного класу, а значить, їх конструювання має відбуватися окремо. Якщо конструктор визначений тільки в похідному класі, то все відбувається дуже просто: конструюється об'єкт похідного класу, а базова частина об'єкта автоматично конструюється його конструктором, використовується за умовчанням. Приклад. Додамо в базовий клас якості для полів для організації захищеного доступу, а в похідний конструктор для поля Style. Код Спадкування і приховування іменВ похідному класі можна визначити член з таким же ім'ям, як і у члена його базового класу. В цьому випадку член базового класу ховається в похідному класі. І хоча формально в C # це не вважається помилкою, компілятор

все ж видасть повідомлення, яке попереджає про те, що ім'я ховається. Якщо член базового класу потрібно приховати навмисно, то перед його ім'ям слід вказати ключове слово new, щоб уникнути появи подібного застережливого повідомлення. Слід, однак, мати на увазі, що це абсолютно відділ ьное застосування ключового слова new, не схоже на його застосування при створенні екземпляра об'єкта.Ніже наведено приклад приховування імені .// Приклад приховування імені зі спадковою связью.using System; class A {public int i = 0;} // Створити похідний клас. class B: A {new int i; // Цей член приховує член і з класу Apublic B (int b) {i = b; // Член і в класі B} public void Show () {Console.WriteLine ("Член і в похідному класі:" + i);}} class NameHiding {static void Main () {У ob = new У (2); ob.Show ();}} Перш за все зверніть увагу на використання ключового слова new в наступному рядку кода.new int i; // Цей член приховує член і з класу AВ цьому рядку компілятору, по суті, повідомляється про те, що знову створювана змінна і навмисно приховує змінну і з базового класу A і що автору програми про це відомо. Якщо ж опустити ключове слово new в цьому рядку коду, то компілятор видасть попереджувала сообщеніе.Вот до якого результату приводить виконання наведеного вище кода.Член і в похідному класі: 2В класі B визначається власна змінна екземпляра i, яка приховує змінну i з базового класу A . Тому при виклику методу Show () для об'єкта типу в виводиться значення змінної i, визначеної в класі в, а не тієї, що визначена в класі A.Прімененіе ключового слова base для доступу до прихованого іменіІмеется ще одна форма ключового слова base, яка діє подібно ключовим словом this, за винятком того, що вона завжди посилається на базовий клас в тому похідному класі, в якому вона використовується. Нижче ця форма наведена в загальному вигляді: base.переменнаягде змінна може позначати метод або змінну примірника. Ця форма ключового слова base найчастіше застосовується в тих випадках, коли під іменами членів похідного класу ховаються члени базового класу з тими ж самими іменами. Як приклад нижче наведено інший варіант ієрархії класів з попереднього прикладу .// Застосування ключового слова base для подолання // перешкоди, пов'язаного з приховуванням імен.using System; class A

{public int i = 0;} // Створити похідний класс.class B: A {new int i; // Цей член приховує член i з класу Apublic B (int a, int b) {base.i = a; // Тут виявляється прихований член з класу A i = b; // Член i з класу B} public void Show () {// Тут виводиться член i з класу A.Console.WriteLine ("Член i в базовому класі:" + base.i); // А тут виводиться член i з класу B.Console.WriteLine ("Член i в похідному класі:" + i);}} class UncoverName {static void Main () {y ob = new в (1, 2); ob.Show ();}} Виконання цього коду призводить до наступного результату.Член i в базовому класі: 1Член i в похідному класі: 2Несмотря на те що змінна екземпляра i в похідному класі в приховує змінну i з базового класу A, ключове слово base надати їм доступ до змінної i, визначеної в базовому классе.С допомогою ключового слова base можуть також викликатися приховані методи.Напрімер, в наведеному нижче коді клас в успадковує клас A і в обох классахоб'является метод Show (). А потім в методі Show () класу B за допомогою ключового слова base викликається варіант методу Show (), визначений в класі A.using System; class A {public int i = 0; // Метод Show () в класі Apublic void Show ( ) {Console.WriteLine ("Член i в базовому класі:" + i);}} // Створити похідний класс.class в: A {new int i; // Цей член приховує член i з класу Apublic B (int a, int b) {base.i = a; // Тут виявляється прихований член з класу A i = b; // Член i з класу B} // Тут ховається метод Show () з класу A. Зверніть // увагу на застосування ключового слова new.new public void Show () {base.Show (); // Тут викликається метод Show () з класу A // далі виводиться член i з класу BConsole.WriteLine ("Член i в похідному класі:" + i);}} class UncoverName {static void Main () {У ob = new в (1, 2); ob.Show ();}} Виконання цього коду призводить до наступного результату.Член i в базовому класі: 1Член i в похідному класі: 2Как бачите, в вираженні base.Show () викликається варіант методу Show ( ) з базового класса.Обратіте також увагу на наступне: ключове слово new використовується в наведеному вище коді з метою повідомити компілятору про те, що метод Show (), знову оголошений в похідному класі в, навмисно приховує інший метод Show (), визначений в базовому класі A.Созданіе багаторівневої ієрархії классовВ представлених до сих пір прикладах програм використовувалися прості ієрархії

класів, що складалися тільки з базового і похідного класів. Але в C # можна також будувати ієрархії, що складаються з будь-якого числа рівнів спадкування. Як згадувалося вище, багаторівнева ієрархія ідеально підходить для використання одного похідного класу в якості базового для іншого похідного класу. Так, якщо є при класу, A, B і C, то клас C може успадковувати від класу B, а той, у свою чергу, від класу A. У такому разі кожен похідний клас успадковує характерні риси всіх своїх базових класів. Зокрема, клас C успадковує всечлени ласів B і A. Приклад. Додамо похідний клас ColorTriangle, який буде наслідувати класу Triangle і буде містити поля r, g, b - для опису кольору в палітрі RGB.