ПОЛІМОРФІЗМ

Будь-яке навчання водінню не мало б сенсу, якби людина, яка навчилася водити, скажімо, ВАЗ 2106 не міг потім водити ВАЗ 2110 або BMW X3. З іншого боку, важко уявити людину, яка змогла б нормально керувати автомобілем, в якому педаль газу знаходиться лівіше педалі гальма, а замість керма - джойстик.

Вся справа в тому, що основні елементи управління автомобіля мають одну і ту ж конструкцію і принцип дії. Водій точно знає, що для того, щоб повернути ліворуч, він повинен повернути кермо, незалежно від того, є там гідропідсилювач чи ні.

Якщо людині треба доїхати з роботи до дому, то він сяде за кермо автомобіля і буде виконувати одні і ті ж дії, незалежно від того, який саме тип автомобіля він використовує. По суті, можна сказати, що всі автомобілі мають один і той же інтерфейс, а водій, абстрагуючись від сутності автомобіля, працює саме з цим інтерфейсом. Якщо водієві доведеться їхати по німецькому автобану, він, ймовірно вибере швидкий автомобіль з низькою посадкою, а якщо має бути повертатися з віддаленого маральніка в Гірському Алтаї після дощу, швидше за все, буде обраний УАЗ з армійськими мостами. Але, незалежно від того, яким чином буде реалізовуватися рух і внутрішнє функціонування машини, інтерфейс залишиться колишнім.

Поліморфізм - це властивість системи використовувати об'єкти з однаковим інтерфейсом без інформації про тип і внутрішню структуру об'єкта.

Віртуальні методи

Віртуальним називається такий метод, який оголошується як virtual в базовому

класі. Віртуальний метод відрізняється тим, що він може бути перевизначений в одному або декількох похідних класах. Отже, у кожного похідного класу

може бути свій варіант віртуального методу.

Крім того, віртуальні методи цікаві тим, що саме відбувається при їх виклик за посиланням на базовий клас. В цьому випадку засобами мови C # визначається саме той варіант віртуального методу, який слід викликати, виходячи з типу об'єкта, до якого відбувається звернення за посиланням, причому це робиться під час виконання. Тому при посиланні на різні типи об'єктів виконуються різні варіанти віртуального методу. Іншими словами, варіант виконуваного віртуального методу вибирається по типу об'єкта, а не по типу посилання на цей об'єкт. Так, якщо базовий клас містить віртуальний метод і від нього отримані похідні класи, то при зверненні до різних типів об'єктів по посиланню на базовий клас виконуються різні варіанти цього віртуального методу.

Метод оголошується як віртуальний в базовому класі за допомогою ключового слова

virtual, зазначених вище перед його ім'ям. Коли ж віртуальний метод переопределяется

в похідному класі, то для цього використовується модифікатор override.

А сам процес повторного визначення віртуального методу в похідному класі

називається перевизначенням методу. При перевизначенні ім'я, повертається тип

і сигнатура переобумовленої методу повинні бути точно такими ж, як і у того

віртуального методу, який переопределяется. Крім того, віртуальний метод не

може бути оголошений як static або abstract (докладніше це питання розглядається

далі в цьому розділі).

Перевизначення методу служить підставою для втілення одного з найбільш

ефективних в C # принципів: динамічної диспетчеризації методів, яка представляє собою механізм дозволу виклику під час виконання, а не компіляції. Значення динамічної диспетчеризації методів полягає в тому, що саме завдяки їй в C # реалізується динамічний поліморфізм.

Розглянемо найпростіший приклад. Створимо базовий клас віртуальним методом і два класи нащадка, які скасовують даний метод.

```
class Base
{
   // Створити віртуальний метод в базовому класі.
   public virtual void Who (Label l)
   {
      l.Text = "Метод Who () в класі Base";
```

```
    }
}
class Derivedl: Base
{
    // Відхилити метод Who () в похідному класі.
    public override void Who (Label l)
    {
        l.Text = "Метод Who () в класі Derivedl";
    }
}
class Derived2: Base
{
    // Знову перевизначити метод Who () в ще одному похідному класі.
    public override void Who (Label l)
    {
        l.Text = "Метод Who () в класі Derived2";
    }
}
```

Код основної програми

```
    private void button1_Click (object sender, EventArgs e)
    {
        Base b = new Base ();
```

```
        Derivedl d1 = new Derivedl ();
        Derived2 d2 = new Derived2 ();
        b.Who (label1);
        d1.Who (label2);
        d2.Who (label3);
    }
```

У коді з наведеного вище прикладу створюються базовий клас Base і два похідних

від нього класу - Derived1 і Derived2. У класі Base оголошується віртуальний

метод Who (), який переопределяется в обох похідних класах. потім в

методі Main () оголошуються об'єкти типу Base, Derived1 і Derived2. Крім того,

оголошується змінна baseRef посилального типу Base. Далі посилання на кожен тип

об'єкта присвоюється пере тимчасової baseRef і потім використовується для виклику методаWho (). Як випливає з результату виконання наведеного вище коду, варіантвиполняемого методу Who () визначається за типом об'єкта, до якого проісходітобращеніе за посиланням під час виклику цього методу, а не по типу класу переменнойbaseRef.Но перевизначати віртуальний метод зовсім не обов'язково. Адже якщо в проізводномклассе не надається власний варіант віртуального

методу, то іспользуетсяего варіант з базового класу. Якщо при наявності багаторівневої ієрархії віртуальний метод не переопределяется в похідному класі, то виконується найближчий його варіант, обнаружіваемийвверх по ієрархії.Что дає перевизначення методовБлагодаря перевизначення методів в C # підтримується динамічний поліморфізм.В об'єктно-орієнтованому програмуванні поліморфізм іграеточень важливу роль, тому що він дозволяє визначити в загальному класі методи, які стають загальними для всіх похідних від нього класів, а в проізводнихклассах - визначити конкретну реалізацію деяких або ж усіх цих методов.Переопределеніе методів - це ще один спосіб втілити в C # головний прінціпполіморфізма: один інтерфейс - безліч методов.Удачное застосування поліморфізму частково залежить від правильного поніманіятой особливості, що базові та похідні класи утворюють ієрархію, яка просувається від меншої до більшої спеціалізації. При належному застосуванні базовий клас надає всі необхідні елементи, які можуть іспользоватьсяв похідному класі безпосередньо. А за допомогою віртуальних методів в базовомклассе визначаються ті методи, які можуть бути самостійно реалізовані впроізводном класі. Таким чином, поєднуючи успадкування з віртуальними методами, можна визначити в базовому класі загальну форму методів, які будуть використовуватися у всіх його похідних классах.Прімер. Скласти програму з одним батьківським класом і двома нащадками. Нащадки повинні містити віртуальні функції. Створити віртуальну функцію видачі результатів розрахунку методів на екран монітора із зазначенням назв і полів і їх значень відповідного об'єкта. Скласти тестуючу програму з видачею протоколу на екран

монітора. При цьому створити об'єкти базового і похідних типів, використовуючи поліморфний контейнер - масив посилань базового класу на об'єкти базового і похідних класів (кількість об'єктів> = 5) .Варіанти Батьківський клас Нащадки Поліморфні методи 1. Тривимірна Фігура (поле назву) Конус (радіус r, h - висота) Правильна чотирикутна піраміда (a - сторона квадрата, h- апофема, тобто висота) Розрахунок площі Діаграма класів Наведемо класси1. Базовий клас для тривимірної фігури 2. Клас для конуса 3. Клас для піраміди Основна програма Застосування абстрактних классовІногда потрібно створити базовий клас, в якому визначається лише сама общаяформа для всіх його похідних класів, а наповнення її деталями предоставляетсякаждому з цих класів. У такому класі визначається лише характер методів, які повинні бути конкретно реалізовані в похідних класах, а не в самому базовому класі. Подібна ситуація виникає, наприклад, у зв'язку з неможливістю получітьсодержательную реалізацію методу в базовому класі. Створюючи власні бібліотеки класів, ви можете самі переконатися в тому, чтоу методу часто відсутня змістовне визначення в контексті його базовогокласса. Подібна ситуація вирішується двома способами. Один з них, полягає в тому, щоб просто видати попередження. Такий спосіб може стати в нагоді в певних ситуаціях, наприклад при налагодженні, але в практиці програмування він зазвичай не застосовується. Адже в базовому класі можуть бути оголошені методи, які повинні бути перевизначені в похідному класі, щоб цей клас став змістовним. У подібних випадках потрібно якийсь спосіб, який гарантує, що в похідному класі дійсно будуть перевизначені всі необхідні методи. І такий спосіб в C # є. Він полягає у використанні абстрактного

метода.Абстрактний метод створюється за допомогою зазначених вище модифікатора типаabstract. У абстрактного методу відсутнє тіло, і тому він не реалізується в базовомклассе. Це означає, що він повинен бути перевизначений в похідному класі, оскільки його варіант з базового класу просто непридатний для використання. Неважко здогадатися, що абстрактний метод автоматично стає віртуальним іне вимагає вказівки модифікатора virtual. Насправді спільне використання модифікаторів virtual і abstract вважається ошібкой.Для визначення абстрактного методу служить наведена нижче загальна форма.abstract тип ім'я (спісок_параметров); Як бачите, у абстрактного методу відсутнє тіло. Модифікатор abstract можетпріменяться тільки в методах примірника, але не в статичних методах (static) .Абстрактнимі можуть бути також індексатори і свойства.Класс, що містить один або більше абстрактних методів, повинен бути також оголошений як абстрактний, і для цього перед його оголошенням class вказується м одіфікаторabstract. А оскільки реалізація абстрактного класу не определяетсяполностью, то у нього не може бути об'єктів. Отже, спроба створити об'єкт абстрактного класу за допомогою оператора new призведе до помилки під час компіляції.Когда похідний клас успадковує абстрактний клас, в ньому повинні бути реалізованивсе абстрактні методи базового класу. В іншому випадку проізводнийкласс повинен бути також визначено як abstract. Таким чином, атрибут abstractнаследуется до тих пір, поки не буде досягнута повна реалізація класса.Прімер. Скласти програму з абстрактним батьківським класом і двома об'єктами - нащадками. Для цього модифікувати завдання 1. Скласти тестуючу програму з видачею протоколу на екран монітора. У ній потрібно

реалізувати циклічний висновок параметрів об'єктів, використовуючи поліморфний контейнер - масив об'єктів базового класу (кількість об'єктів> = 5) .Заданіе. Знайти об'єкт з максимальною площадью.Опішем класси1. Базовий клас для тривимірної фігури 2. Клас для конуса Клас для піраміди Основна програма