Тема 4

Класове в C#. Методи. Обекти (инстанции) на класовете

1. Класове и методи, модификатори на достъп

1.1. Определение за клас в C#

Клас в C#, какво и в други езици за програмиране е тип данни, дефиниран от програмиста, който се състои от данни, наричани полета и функции за работа с тези данни, наречени методи. Чрез използването на класове се позволява да бъдат моделирани в едно данни и функционалност обекти от реалния свят. Тази особеност на класовете е едно от най-големите предимства на обектно-ориентираното програмиране. В C# класовете спадат към адресните типове данни, за разлика от структурите, с който са близки по функционалност, но са стойностни типове.

Общият вид на декларацията на клас в C# е следния:

```
class ИмеКлас
{
    // тяло на клас
}
```

В език C# целия код за даден клас се разполага в един файл. За разлика от C++ няма различия между декларация и дефиниция т.е. не се използва концепцията за декларация на функция.

В тялото на C# клас се дефинират следните членове на класа:

- полета;
- свойства;
- методи.

Полета се наричат данните, дефинирани в тялото на класа. Те служат за описание на характеристиките на реалния обект.

Пример:

```
class Rectangle
    {
        double height; //поле
        double width;  //поле
        double Area()  //метод
        {
            return height * height;
        }
    }
```

1.2. Дефиниране и извикване на методи. Оператор return

В C# методът е именувана последователност от конструкции. Методите в C# са аналози на функциите в процедурно-ориентираните програмни езици. Тъй като C# е изцяло обектно-ориентиран, на практика всички функции са методи на класовете т.е. езикът не позволява съществуването на външни функции*.

Общият вид на дефиниране на метод е следния:

```
тип ИмеМетод(списък формални параметри)
{
    // тяло на метод
}
```

където:

- тип - тип на върнатия резултат;
- ИмеМетод - идентификатор, определящ еднозначно метода;
- списък формални параметри - параметрите на метода, определени чрез тип и име (идентификатор)

Тялото на метода съдържа блок оператори и управляващи конструкции, които се изпълняват при стартиране на метода. Както и в C и C++ в тялото на функцията-метод трябва да се съдържа оператор **return**, който връща резултата от изпълнението на метода. Типът на върнатата стойност съвпада с типа на метода.

Общият вид на оператора е:

```
return израз;
```

Методи от тип **void** не връщат резултат и в този случай общият вид на оператора е:

```
return;
```

Ако методът не връща резултат и в тялото му липсва оператор **return**, тогава той връща управлението при достигане на затварящата фигурна скоба **}** . Въпреки че това е често срещано на практика, липсата на оператор **return** се приема като лош стил на програмиране.

Извикването на метод е чрез неговото име и списъка с фактически параметри. Подобно на C и C++ при извикването на метод фактическите параметри предават своите стойности на формалните.

В C# има два типа методи статични методи и методи на екземплярите. Статичните методи могат да се извикват без да е необходимо да се използва обект (екземпляр) от класа подобно на външните функции в C и C++. За разлика от тях обаче, задължително при извикването на метода трябва да се посочи и името на класа. Методите на екземплярите задължително изискват обект чрез който да бъдат извикани. Двата типа методи ще бъдат разгледани по-подробно в следващите раздели.

---

* Под понятието „външна функция” се разбира функция, която е дефинирана на глобално ниво и не принадлежи на даден клас. Понятието е заимствано от език C++, където е възможно дефиниране на функция на глобално ниво.

1.3. Модификатори на достъп

Модификаторите на достъп определят по какъв начин членовете на дадения клас са видими. Чрез тях се реализира една от основните концепции в обектно-ориентираното програмиране – капсулация (encapsulation). Според този принцип, част от членовете на класа могат да бъдат видими само за методи, принадлежащи на класа и скрити за останалите. Обикновено данните в класа са достъпни единствено за неговите методи т.е. функциите на класа са тези, които могат работят с тези данни, а за всички останали методи данните са скрити. Така от една страна се постига защита на данните и от друга, не е необходимо всеки метод да има достъп до всички данни, а само до тези, към които има отношение. В голяма част от случаите методите са достъпни от всякъде, откъдето е видим класът. Език C# поддържа следните модификатори на достъп: **private**, **public**, **protected** и **internal**. Модификатори **protected** и **internal** се използват при изграждането на йерархии от класове. Възможно е те да бъдат използвани съвместно.

Модификаторите на достъп в език C# определят следното:

- **public** – определя, че членовете са достъпни от всякъде от където се вижда класът.
- **private** – определя, че даденият член достъпен само за класа, в който е дефиниран. За членове на клас модификатор **private** е по подразбиране т.е. ако бъде пропуснат модификатор пред дефиницията на метод, например, то този метод е достъпен само за останалите методи на класа.
- **protected** – членът на дадения клас е достъпен както за класа, в който е определен, така и за производните класове. За останалите е недостъпен.
- **internal** – членът на дадения клас е достъпен за всички класове в множеството, за което е определен. Извън него е недостъпен.
- **protected internal** – определя, че членът на дадения клас е достъпен за собствените методи на класа и само за методите на тези наследени класове, които са в множеството.

Освен за членовете на клас, модификаторите на достъп се използват и в декларациите на класовете. Ако бъде изпуснат модификатор на достъп пред декларацията на класа, той по подразбиране е **internal**.

2. Дефиниране на обекти (инстанции на класа)

Тъй като класът е референтен тип данни, създаването на обект-екземпляр на класа е чрез оператор **new**.

Общият вид на дефиницията на обект е:

ИмеКлас имеОбект = new ИмеКлас();

Например, нека е деклариан клас `Rectangle`. Дефинирането на обект от класа във тялото на метод Main е по следния начин:

```
class Program
```

```
{
      static void Main()
      {
            Rectangle firstRect = new Rectangle();
      }
}
```

Чрез посочения програмен ред се дефинира обект **`firstRect`** от клас **`Rectangle`**. Важно е да се отбележи, че името на обекта е референция, която се пази в стека. Референцията на обект сочи мястото в динамичната памет, където логически се съхраняват полетата на обекта. При дефинирането на обект в динамичната памет се заделя място за всяко едно от полетата на класа. На практика програмният обект (екземпляр на класа) е логическа съвкупност от всички полета на класа.

От даден клас могат да бъдат създадени неопределен брой програмни обекти – толкова, колкото е необходимо. Съответно, за всеки един от обектите в динамичната памет се съхраняват копия на полетата на класа. На практика класът се явява тип, от който се дефинират програмни обекти.

3. Извикване на методи

В C# има два типа методи статични методи и методи на екземплярите. Методите на екземплярите задължително изискват обект чрез който да бъдат извикани. В случая се използва един от най-често срещаните оператори в обектно-ориентираното програмиране - оператор . (точка). Общият вид на извикването на метод е следния:

имеОбект.ИмеМетод(списък фактически параметри);

Например, нека в клас `Rectangle` е дефиниран метод `Area()` , който връща като резултат площта на правоъгълника. Чрез следния програмен ред се извежда на екрана площта на правоъгълника, съответстващ на вече дефинирания обект **`firstRect`** :

```
Console.WriteLine("The area is " + firstRect.Area());
```

Общъщението към метода е `firstRect.Area()`. В случая, метод Area не изисква параметри и затова не се посочват в скобите конкретни параметри.

Относно въпроси по темата на адрес:
ln_zh_st@yahoo.com