Тема 5

Област на действие на програмните единици. Статични членове на клас. Модификатори на параметри

1. Полета. Област на видимост на променливите

В програмен код на език C# променливите се дефинират или в телата на методите, или в телата на класовете. Променливите, дефинирани в тялото на метод се наричат **локални**. Те се използват само в тялото на метода. Формалните параметри на методите също спадат към локалните променливи. Променливите, дефинирани в тялото на класа се наричат **полета**. За разлика от локалните променливи те се използват за споделяне на информация между методите на класа.

Блокът, определен от отварящата и затваряща фигурни скоби определя областта на видимост на даден идентификатор, в частност променлива. Ако дадена променлива е дефинирани в тялото на метод, то тя е видима само в тялото на този метод (локална променлива) и съществува по време на неговото изпълнение. След приключване на работата му локалната променлива ще бъде премахната. Например, следващият програмен код ще генерира грешка в ред 9, тъй като променливата var е локална за метод **`firstMethod()`**и съответно е недостъпна за втория метод **`secondMethod()`**.

```
class Examp
{
    public void firstMethod()
    {
        int var = 10;
    }
    public void secondMethod()
    {
             var = 20; // грешка
    }
}
```

Отварящата и затварящата фигурни скоби за тялото на класа също пораждат локална област на видимост. Всички променливи, дефинирани в тялото на класа, извън методите имат за локална област на видимост границите на същия този клас. Под термина **поле** се разбира променлива, дефинирана в тялото на класа. За разлика от локалните променливи те се използват за споделяне на информация между методите на класа. Възможно е, име на поле да съвпада с име на локална променлива (променлива, дефинирана в тялото на метода или формален параметър на метода). В случая, локалното име е с по-висок приоритет. За да се осигури достъп до полето на класа, може да се използва ключова дума this, която е референция към текущият обект. Пример:

```
class Examp
{
    private int var=20;
```

```
    public void firstMethod()
    {
        int var = 10;
        Console.WriteLine(var);      //10
        Console.WriteLine(this.var); //20
    }
}
```

2. Статични методи и статични данни

2.1. Статични методи и методи на екземплярите

Както беше вече споменато (лекция № 4), в език C# има два типа методи – статични методи и методи на екземплярите. Методите на екземплярите (instance methods) се извикват чрез обект от класа и съответно, са свързани с обекта (екземпляр на класа). Съответно, за да бъде извикан такъв метод, необходимо е да бъде дефиниран обект от класа.

Методите на класовете могат да бъдат обявени като статични чрез ключова дума **static**. Статичните методи могат да бъдат извиквани без да е необходимо да бъде дефиниран обект от клас. Статичен метод (static method) е този, който може да бъде извикан чрез класа без да е необходимо да бъде създаден екземпляр (обект) от този клас. Извикването на такъв метод е по следния начин:

ИмеКлас.ИмеМетод(фактически параметри);

Пример:

```
class Examp
{

 public static void Method()
      {     ...
      }

}
```

В случая, извикването на метода е чрез програмен ред:

```
Examp.Method();
```

Идентификатор **`Examp`** е име на клас, тъй като метода **`Method()`** е статичен.

Ако даден метод не е обявен като статичен, по подразбиране се счита, че той е метод на екземпляр.

2.2. Статични данни

Класовете в C# могат да съдържат не само статични методи, но и статични данни. За разлика от останалите, статичните данни се използват съвместно от

всички обекти на класа. Това означава, че всички обекти от класа ще имат достъп до една и съща променлива (поле).

Например, нека в клас е дефинирано поле, която не е статично:

```
class Number
{
      public int num;
      ...
}
```

Ако бъдат дефинирани обекти от този клас, за всеки обект ще бъде направено отделно копие на тази променлива:

```
Number f1=new Number();
f1.num = 100;

Number f2=new Number();
f2.num = 10;

Number f3=new Number();
f3.num = 1;
```

В този случай и за трите екземпляра (обекта) на класа (**f1,f2, f3**) съществуват копия на полето **num**.

Ако дадено поле бъде дефинирано от тип **static**, това означава, че ще съществува едно единствено копие на тази променлива и всички обекти от класа ще го използват.

Пример:

```
class Number
{
      public static int numberOfObjects;
      ...
}
```

В този случай, трите екземпляра на класа ще споделят една обща променлива **numberofobjects**. Например, програмен код

```
f1.numberOfObjects = 10;
Console.WriteLine(f2.numberOfObjects);
```

Ще изведе стойност 10 на екрана, тъй като полето **numberofobjects** е общо за всички обекти на класа.

3. Параметри на методите. Модификатори на параметри

Методите формират функционалността на класа. За да могат да обработват някаква информация те трябва да получат входни данни, предадени като параметри. За да се определи от какъв тип са параметрите могат да се използват следните модификатори:

in – определя, че параметърът е входен. При входните параметри променливата на метода се инициализира с копие на стойността от извикващия процес.

Модификатор **in** е по подразбиране, което означава, че по подразбиране параметрите са входни.

Пример:

```
class Calc
{
 public static int Squared(int n)
 {
       return n*n;
 }
}
```

В този случай, параметърът n в метод **`Squared`** е входен. Методът получава копие на стойността на n и няма как да промени стойността, предадена като параметър. Методът връща като резултат стойността на параметъра, повдигната на квадрат. Следващите програмни редове показват как се извиква този метод:

```
int n = Int32.Parse(Console.ReadLine());
int k = Calc.Squared(n);
```

Може да се направи известна аналогия с предаването на параметри по стойност в програмни езици C и C++. Съществува обаче една особеност в C#, където част от типовете са адресни и в действителност тяхната стойност е равносилна на адрес.

out – параметърът се предава като изходен. Това означава, че този тип параметри се използват само за върнат резултат.

Пример:

```
class Clac
{
 public static void Squared(int n, out int k)
 {
       k=n*n;
 }
}
```

В този случай, вторият параметът k се използва за резултат от изчислението. Извикването на този вариант на метода се илюстрира чрез следните програмни редове:

```
int n = Int32.Parse(Console.ReadLine());
int k;
Calc.Squared(n, out k);
```

Променливата k, която се предава като изходен параметър не трябва да се изниициализира. Ключова дума out задължително се пише пред името на параметъра при извикване на метода.

ref – параметърът се предава чрез псевдоним, което означава, че се предава чрез адрес. Такъв параметър може да бъде модифициран в метода.

Пример:

```
class Clac
{
 public static void Squared(ref int n)
 {
       n=n*n;
 }
}
```

За да се предаде такъв параметър е необходимо да се добави ключова дума ref както в дефиницията на параметъра, така и при неговото извикване.

Пример:

```
int n = Int32.Parse(Console.ReadLine());
Calc.Squared(ref n);
```

Да се има предвид, че ако се предава параметър чрез псевдоним, той задължително трябва да е инициализиран.

params – модификаторът се използва когато се предават набор параметри като един. Ключовата дума params може и да не бъде обявена при извикване на метода.

Пример:

```
class Example
{
      public static void ShowIntArray(string msg, params int [] array)
      {
            Console.WriteLine(msg);
            foeach(int k in array)
            {
                  Console.Write(k+" ");
            }
      }
}
```

В посочения примерен код, метод **`ShowIntArray`** извежда на екрана стойностите на елементите на едномерен масив, предаден като един параметър. Следващите програмни редове демонстрират извеждането на екрана на целочислени стойности, елементи на масив, с помощта на метода:

```
int [] firstArray = {1,3,5,7,9};
Example.ShowIntArray("First Array:", firstArray);
Example.ShowIntArray("Second Array:",2,4,6,8,10);
```

При извеждането на елементите на масива `firstArray` метод `ShowIntArray` ще изведе стойностите на елементите на масива дори и параметърът да не е обявен с ключова дума params. При второто извикване на метода `ShowIntArray`, групата параметри `2,4,6,8,10` се предават като един благодарение на факта, че се използва модификатор params. Необходимо е да се отбележи, че ключова дума params не се пише при извикване на метода. В дефиницията на метода ключова дума params задължително е пред последния параметър.

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