زنگ سی‌شارپ – قسمت بیست و چهارم

نوشته‌ی مسعود درویشیان

لینک مستقیم این مطلب در وب‌تارگت

فرستادن Reference به متدها

تا این قسمت از زنگ سی‌شارپ، پارامترهایی که به متد داده می‌شدند همه‌گی value type بودند (مانند int یا double و...) اما علاوه بر value type شما می‌توانید از reference type نیز به‌عنوان پارامتر استفاده کنید. این کار به یک شیء اجازه می‌دهد تا بتواند به یک متد فرستاده شود.

به مثال زیر توجه کنید:

```
using System;
class MyClass
{
    string Name;
    string Surname;
    int Age;

    // Constructor
    public MyClass(string name, string surname, int age)
    {
        Name = name;
        Surname = surname;
        Age = age;
    }

    // Return true if ob contains the same values as the invoking object.
    public bool SameAs(MyClass ob)
    {
        if (Name == ob.Name && Surname == ob.Surname && Age == ob.Age)
            return true;
        else return false;
    }

    // Make a copy of ob
    public void Copy(MyClass ob)
    {
        Name = ob.Name;
        Surname = ob.Surname;
        Age = ob.Age;
    }

    public void Show()
    {
        Console.WriteLine("  Name: {0}, Surname: {1}, Age: {2}",
            Name, Surname, Age);
    }
}
class PassOb
{
```

```
    static void Main()
    {
        MyClass ob1 = new MyClass("Damon", "Salvatore", 22);
        MyClass ob2 = new MyClass("Stefan", "Salvatore", 21);

        Console.WriteLine("ob1: ");
        ob1.Show();
        Console.WriteLine("ob2: ");
        ob2.Show();

        Console.WriteLine();

        if(ob1.SameAs(ob2))
            Console.WriteLine("ob1 and ob2 have the same values.");
        else
            Console.WriteLine("ob1 and ob2 have different values.");

        // Now, make ob1 a copy of ob2
        ob1.Copy(ob2);
        Console.WriteLine();
        Console.WriteLine("ob1 after copy: ");
        ob1.Show();

        Console.WriteLine();
        if (ob1.SameAs(ob2))
            Console.WriteLine("ob1 and ob2 have the same values.");
        else
            Console.WriteLine("ob1 and ob2 have different values.");
    }
}
```

متد SameAs() و متد Copy() هرکدام یک Reference را به‌عنوان argument دریافت می‌کنند (همان‌طور که می‌دانید Reference آدرس یک شیء در حافظه است). متد SameAs() مقادیر Name، Surname و Age را از متدی که فراخوانی شده با مقادیر Name، Surname و Age شیء ob که به متد داده شده است، مقایسه می‌کند و در صورت یکسان بودن این مقادیر، true برمی‌گرداند. متد Copy() نیز مقادیر شیء ob را به مقادیر شیء فراخوانی‌شده اختصاص می‌دهد. همان‌طور که می‌بینید به همان روشی که value type ها به متدها داده می‌شوند، reference type ها نیز داده شده‌اند.

از دو طریق Argument ها به parameter ها فرستاده می‌شوند:

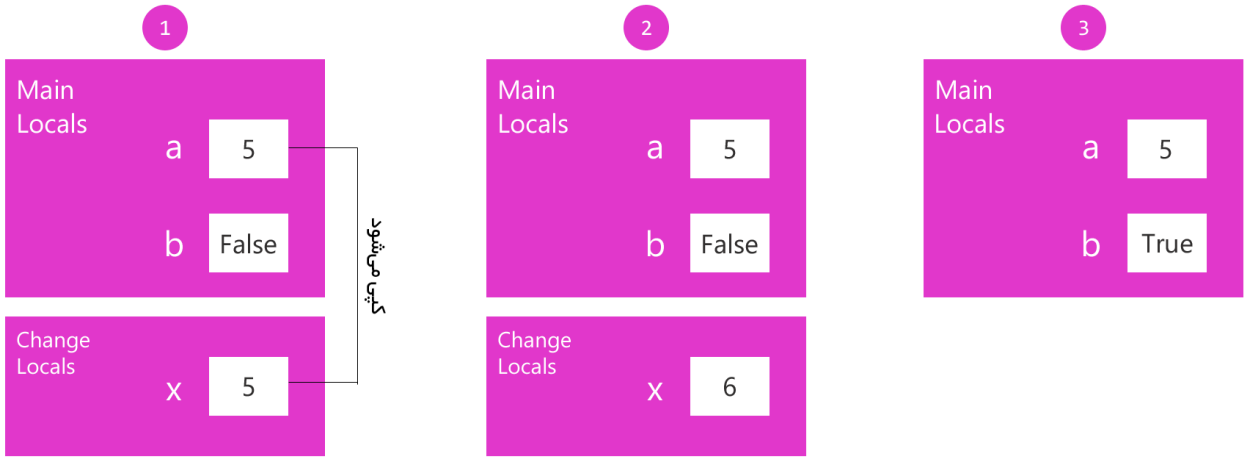
- Call-by-value
- Call-by-reference

در روش اول (call-by-value) از مقدار argument یک کپی گرفته شده و به پارامتر داده می‌شود. از این‌رو، هر بلایی که سر پارامتر آورید هیچ تغییری روی argument صورت نمی‌گیرد.

به مثال زیر توجه کنید:

```
using System;
class MyClass
{
    static void Main()
    {
        int a = 5;
        Change(a);
        Console.WriteLine(a);
    }
    static void Change(int x)
    {
        x++;
    }
}
```

مقدار a برابر با ۵ است. argument متد Change() مقدار a است. از این مقدار یک کپی گرفته شده و به پارامتر این متد (x) فرستاده می‌شود. هنگامی که درون این متد مقدار x افزایش می‌یابد. تغییر x هیچ تاثیری روی مقدار a نمی‌گذارد. به شکل زیر توجه کنید:




```
static void Main()
{
    int a = 5;
    Change(a);
    bool b = (a == 5);
}
static void Change(int x)
{
    x++;
}
```

```
static void Main()
{
    int a = 5;
    Change(a);
    bool b = (a == 5);
}
static void Change(int x)
{
    x++;
}
```

```
static void Main()
{
    int a = 5;
    Change(a);
    bool b = (a == 5);
}
static void Change(int x)
{
    x++;
}
```

همان‌طور که می‌بینید با تغییر پارامتر، مقدار a هیچ تغییری نمی‌کند زیرا کپی مقدار a در x قرار دارد و این‌دو کاملاً مستقل از هم هستند.

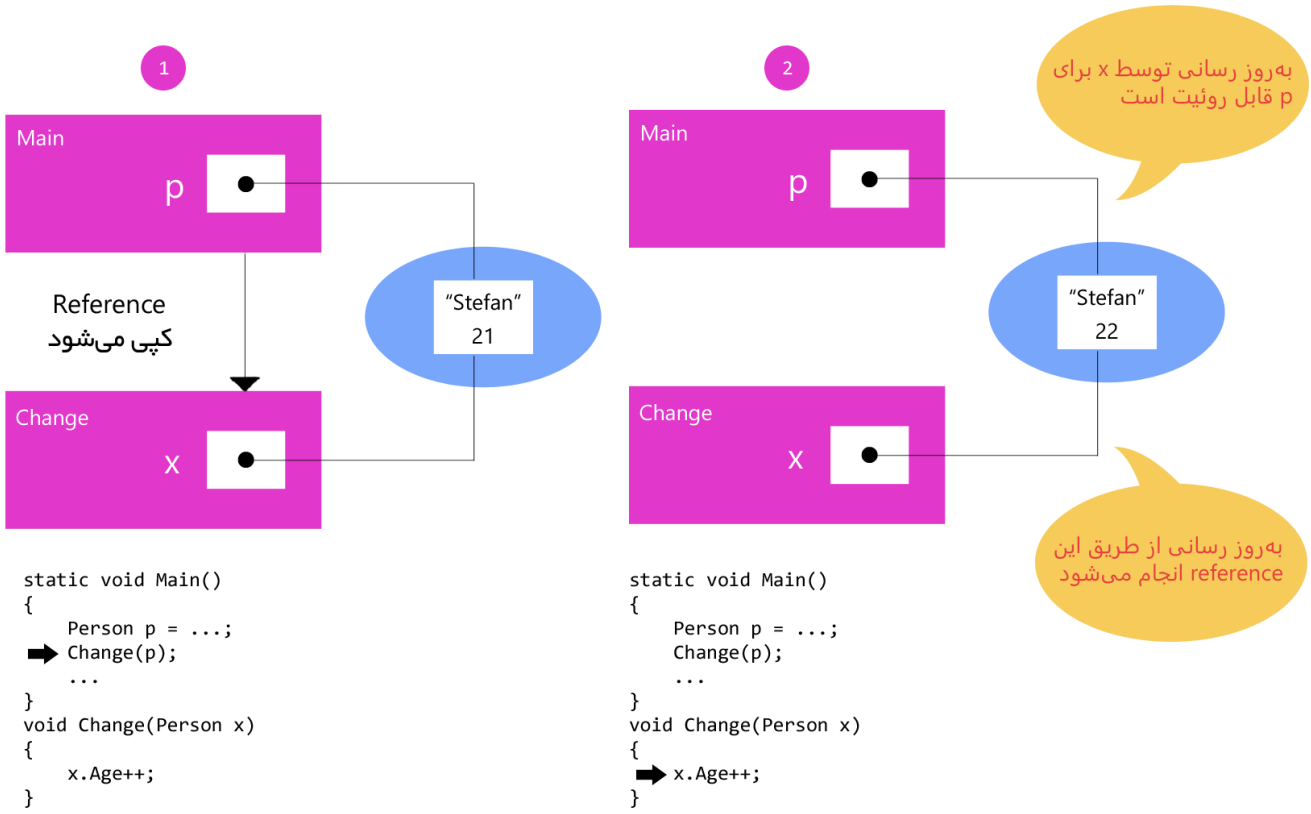
در روش دوم (call-by-reference) یک reference به‌عنوان argument به متد داده شده و کپی این reference به پارامتر فرستاده می‌شود. درون متد، پارامتر به همان شیءای رجوع می‌کند که argument رجوع می‌کند. این یعنی اگر هر تغییری روی پارامتر انجام دهید، این تغییر روی argument نیز تاثیر می‌گذارد زیرا هردو به یک شیء وصل هستند و تغییر هرکدام، شیء را تحت تاثیر قرار می‌دهد.

به مثال زیر توجه کنید:

```
// Objects are passed by reference.
using System;
class Test
{
    public int a, b;
    public Test(int i, int j)
    {
        a = i;
        b = j;
    }
    /* Pass an object. Now, ob.a and ob.b in object
    used in the call will be changed. */
    public void Change(Test ob)
    {
        ob.a = ob.a + ob.b;
        ob.b = -ob.b;
    }
}
class CallByRef
{
    static void Main()
    {
        Test ob = new Test(15, 20);
        Console.WriteLine("ob.a and ob.b before call: " +
        ob.a + " " + ob.b);
        ob.Change(ob);
        Console.WriteLine("ob.a and ob.b after call: " +
        ob.a + " " + ob.b);
    }
}
```

همان‌طور که می‌بینید تغییر در پارامتر متد Change() باعث تغییر مقادیر شیء می‌شود.

به شکل زیر دقت کنید:




```
static void Main()
{
    Person p = ...;
    Change(p);
    ...
}
void Change(Person x)
{
    x.Age++;
}
```

```
static void Main()
{
    Person p = ...;
    Change(p);
    ...
}
void Change(Person x)
{
    x.Age++;
}
```

همان‌طور که می‌بینید تغییرات در متد Change() موجب تغییر شیءای می‌شود که argument نیز به آن رجوع می‌کرد.