زنگ سی‌شارپ – قسمت بیست و پنجم

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:

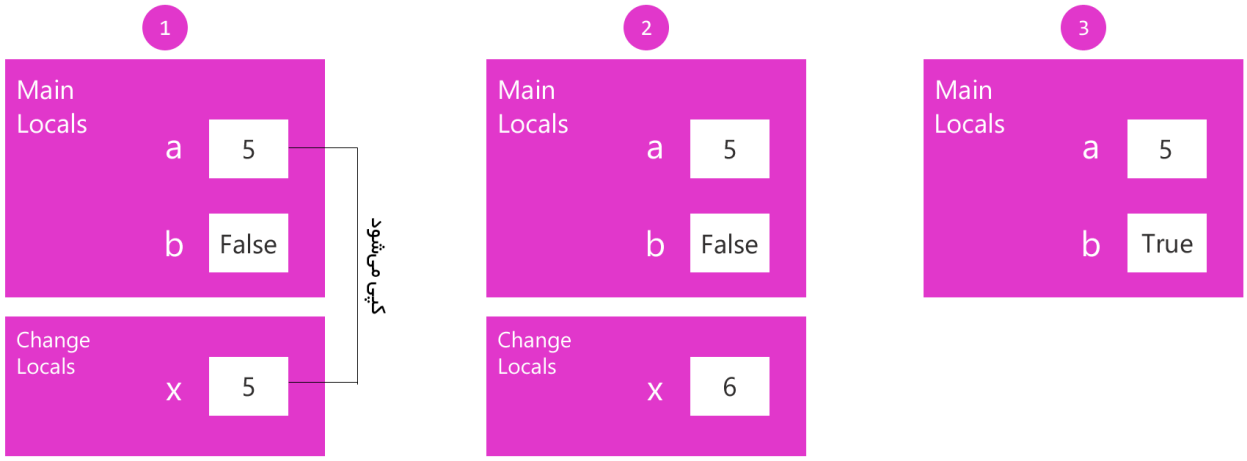
- Call-by-value
- Call-by-reference

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. هنگامی‌که 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.

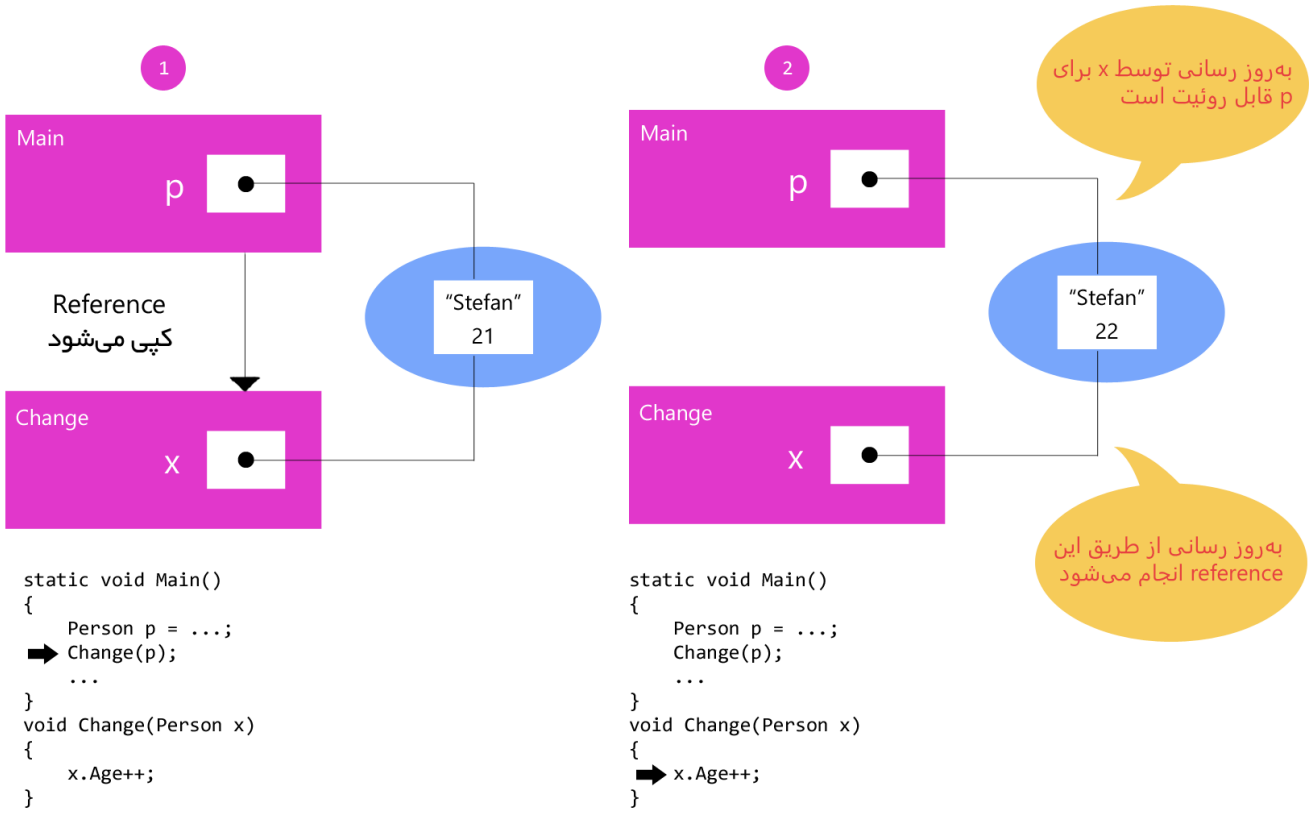
در روش دوم (call-by-reference) یک reference به‌عنوان argument به متد داده شده و کپی این reference به پارامتر فرستاده می‌شود. درون متد، پارامتر به همان شیءای رجوع می‌کند که argument رجوع می‌کند. این یعنی اگر هر تغییری روی پارامتر انجام دهید، این تغییر روی argument نیز تاثیر می‌گذارد زیرا هردو به یک شیء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

استفاده از پارامترهای ref و out

همان‌طور که گفته شد، value type ها از طریق مقدارشان به یک متد فرستاده می‌شوند و تغییر در پارامترهایی که argument را دریافت می‌کنند، در argument تغییری ایجاد نمی‌کند. سی‌شارپ به شما اجازه می‌دهد که این رفتار را از طریق کلمات کلیدی ref و out تغییر دهید. در سی‌شارپ این امکان وجود دارد که value type را توسط reference ارسال کنید. این کار به متد این امکان را می‌دهد تا بتواند argument را تغییر دهد.

پیش از اینکه به چگونگی مکانیسم ref و out بپردازیم نیاز است بدانید که چرا باید value type را توسط reference ارسال کنیم. به‌صورت کلی، دو دلیل برای این کار وجود دارد: اجازه دادن به متد تا بتواند محتوای argument اش را تغییر دهد. اجازه دادن به متد برای اینکه بتواند بیشتر از یک مقدار را return کند.

ref Parameter Modifier

اغلب شما می‌خواهید یک متد بتواند روی argument هایی که به آن داده می‌شود تغییرات اعمال کند. ساده‌ترین مثال می‌تواند متد Swap() باشد که دو متغیر را می‌گیرد و مقادیر آن‌ها را جابه‌جا می‌کند. در حالت عادی این امکان وجود ندارد که متدی بنویسید تا عملیات swap را انجام دهد اما کلمه کلیدی ref این مشکل را حل می‌کند.

ref modifier موجب می‌شود سی‌شارپ به‌جای call-by-value، call-by-reference ایجاد کند. ref modifier زمانی که متد ساخته و صدا زده می‌شود، مشخص می‌شود. بهتر است با یک مثال موضوع را روشن‌تر کنیم. برنامه زیر یک متد به اسم Sqr() دارد که این متد مقدار داده شده به آن را به توان ۲ می‌رساند:

```
// Use ref to pass a value type by reference.
using System;
class RefTest
{
    // This method changes its argument. Notice the use of ref.
    public void Sqr(ref int i)
    {
        i = i * i;
    }
}
class RefDemo
{
    static void Main()
    {
        RefTest ob = new RefTest();
        int a = 10;
        Console.WriteLine("a before call: " + a);
        ob.Sqr(ref a); // notice the use of ref
        Console.WriteLine("a after call: " + a);
    }
}
```

همان‌طور که می‌بینید ref قبل از پارامتر و قبل از argument آمده است و موجب شده که مقدار a بعد از فراخوانی متد تغییر کند.

به مثال زیر توجه کنید:

```
// Swap two values.
using System;
class ValueSwap
{
    // This method now changes its arguments.
    public void Swap(ref int a, ref int b)
    {
        int t;
        t = a;
        a = b;
        b = t;
```

```
    }
}
class ValueSwapDemo
{
    static void Main()
    {
        ValueSwap ob = new ValueSwap();
        int x = 10, y = 20;
        Console.WriteLine("x and y before call: " + x + " " + y);
        ob.Swap(ref x, ref y);
        Console.WriteLine("x and y after call: " + x + " " + y);
    }
}
```

همان‌طور که می‌بینید بعد از فراخوانی متد Swap() مقدار متغیرهای x و y عوض شده است. نکته‌ی مهم در مورد ref این است که argument ای که از طریق ref ارسال می‌شود باید از پیش تعریف شده باشد. به این دلیل که پارمتر مطمئن شود مقداری که دریافت می‌کند، مقداری معتبر است. بنابراین نمی‌تونید مستقیماً مثلاً مقدار ۱۰ را در argument بنویسید.

out Parameter Modifier

گاهی نیاز دارید که از پارامتر reference استفاده کرده تا مقداری را از یک متد دریافت کنید اما مقداری را به آن ندهید. برای مثال تصور کنید متدی دارید که یک سری عملیات (مثل باز کردن سوکت یک شبکه) را انجام می‌دهد و در نهایت کد موفقیت یا عدم موفقیت را به پارامتر reference باز می‌گرداند. در این‌طور موارد، اطلاعات ارسالی به متد وجود ندارد اما یک‌سری اطلاعات هستند که متد آن‌ها را باز می‌گرداند. مشکل این سناریو این است که ref parameter ابتدا بایستی در یک متغیر مقداردهی اولیه شود بنابراین با استفاده از ref parameter بایستی یک argument با یک مقدار ساختگی (که یک کار بیهوده است) را ارسال کنیم تا به‌اصطلاح دهان compiler را ببندیم که خطایی به برنامه نگیرد. خوشبختانه سی‌شارپ پیشنهاد بهتری برای این مشکل دارد: out parameter

پارامتر out مشابه پارامتر ref است اما یک تفاوت دارد. پارامتر out فقط مقداری را از متد باز می‌گرداند و اطلاعات ارسالی به متد ندارد. نیازی نیست متغیری که به‌عنوان out argument به‌کار می‌رود، در قسمت فراخوانی متد، مقداردهی اولیه شود چراکه متد، خودش یک مقدار را به آن متغیر می‌دهد. علاوه‌براین، درون متد، پارامتر out به‌صورت unassigned در نظر گرفته شده است. بدین معنی که فرض بر این است متغیر مربوطه مقداردهی اولیه نشده است و متد باید خودش مقداری را به متغیر اختصاص دهد. با این تفاسیر بعد از فراخوانی متد، out parameter شامل یک مقدار خواهد بود.

به مثال زیر توجه کنید:

```
class OutExample
{
    static void Method(out int i)
    {
        i = 44;
    }
    static void Main()
    {
        int value;
        Method(out value);
        // value is now 44
    }
}
```

درون متد Method پارامتر out شامل مقدار ۴۴ است. این مقدار از طریق متغیر i وارد متغیر value می‌شود. همان‌طور که می‌بینید value مقداردهی اولیه نشده و پس از فراخوانی متد، مقدار ۴۴ به آن اختصاص داده شده است.

یکی دیگر از کاربردهای out parameter modifier برای این است که یک متد به‌جای یک خروجی، بتواند چندین خروجی داشته باشد.

به مثال زیر دقت کنید:

```
using System;
class OutReturnExample
{
    static void MyMethod(out int i, out string s1, out string s2)
    {
        i = 180;
        s1 = "I've been returned.";
        s2 = "This example uses out to return three variables.";
    }
    static void Main()
    {
        int i;
        string str1, str2;
        MyMethod(out i, out str1, out str2);
        Console.WriteLine(i);
        Console.WriteLine(str1);
        Console.WriteLine(str2);
    }
}
```

همان‌طور که می‌بینید از طریق out parameter modifier توانستیم ۳ مقدار را (به‌جای ۱ مقدار) return کنیم. البته اگر return-type متد را به‌جای void مثلاً bool در نظر می‌گرفتیم می‌توانستیم علاوه بر آن سه مقدار، یک مقدار بولین هم برگردانیم و جمعاً ۴ مقدار را return کنیم.

استفاده از ref و out تنها به فرستادن value type ها محدود نمی‌شود بلکه هنگام فرستادن reference نیز می‌توانند مورد استفاده قرار گیرد. این کار به یک متد اجازه می‌دهد تا بتواند reference ای که به شیءای رجوع کرده را تغییر دهد تا به شیء دیگری رجوع کند. در مثال زیر از ref reference parameter استفاده شده تا دو reference که به ۲ شیء جدا رجوع می‌کنند را با هم تعویض کند:

```
using System;
class MyClass
{
    static void Main()
    {
        Swap swapRef1 = new Swap(2, 3);
        Swap swapRef2 = new Swap(4, 5);

        Console.Write("swapRef1 befor change: ");
        swapRef1.Show();
        Console.Write("swapRef2 befor change: ");
        swapRef2.Show();
        Console.WriteLine();
        swapRef1.swapping(ref swapRef1, ref swapRef2);
        Console.Write("swapRef1 after change: ");
        swapRef1.Show();
        Console.Write("swapRef2 after change: ");
        swapRef2.Show();
    }
}
class Swap
{
    int a, b;
    public Swap(int a, int b)
    {
        this.a = a;
        this.b = b;
    }
    public void swapping(ref Swap ob1, ref Swap ob2)
    {
        Swap t;
        t = ob1;
        ob1 = ob2;
        ob2 = t;
    }
    public void Show()
    {
        Console.WriteLine("a = {0}, b = {1}", a, b);
    }
}
```

در این مثال، متد Swap() اشیای argument هایی که به متد پاس داده شده‌اند را با هم تعویض می‌کند. قبل از فراخوانی متد Swap() متغیر swapRef1 به شیءای با مقادیر ۲ و ۳ و متغیر swapRef2 به شیءای با مقادیر ۴ و ۵ رجوع می‌کند اما بعد از فراخوانی Swap() متغیر swapRef1 به شیءای با مقادیر ۴ و ۵ و متغیر swapRef2 به شیءای با مقادیر ۲ و ۳ رجوع می‌کند. اگر ref را حذف کنید تغییرات اعمال شده فقط درون متد Swap() می‌ماند و در بیرون از متد تاثیری ندارد.

استفاده از argument به تعداد دل‌خواه

هنگامی‌که یک متد می‌سازید می‌دانید که چه تعداد پارامتر دارید و از آن‌طرف چه تعداد argument قرار است که دریافت کنید اما همیشه قضیه به این صورت نیست. به‌عنوان مثال متدی را در نظر بگیرید که میانگین را حساب می‌کند. آیا همیشه باید ۲ یا ۳ عدد را به آن داده تا میانگین‌شان را بدست آورد؟ مسلماً نه! این‌چنین متدی باید هر تعداد که argument به آن داده می‌شود میانگین‌اش را بدست آورد و حد و مرزی نداشته باشد. ممکن است یک‌بار به آن ۲ مقدار داده شود، بار دیگر ۳ یا ۷ مقدار داده شود تا میانگین آن‌ها را حساب کند. برای این منظور باید از کلمه‌ی کلیدی params استفاده کنید. params modifier برای تعریف آرایه‌ای از پارامترها به‌کار می‌رود که می‌تواند صفر یا بیشتر از صفر argument را دریافت کند. سایز این آرایه با توجه به تعداد argument هایی که به متد داده شده تعیین می‌شود.

به مثال زیر توجه کنید:

```
using System;
class MyClass
{

    static void Main()
    {
        ParamTest ob = new ParamTest();

        Console.WriteLine(ob.Average(2, 3, 5, 6));
        Console.WriteLine(ob.Average(2, 3.5));

        double[] myArray = { 5, 8, 6, 12, 15.5, 7 };
        Console.WriteLine(ob.Average(myArray));
    }
}
class ParamTest
{
    public double Average(params double[] nums)
    {
        double result = 0;
        for (int i = 0; i < nums.Length; i++)
        {
            result += nums[i];
        }
        return result / nums.Length;
    }
}
```

به نحوه‌ی استفاده‌ی params در متد ()Average توجه کنید. همان‌طور که می‌بینید در هر بار تعداد متفاوتی از مقادیر نوع double را به متد پاس داده‌ایم و در قسمت آخر آرایه‌ای از جنس double را به متد داده‌ایم تا میانگین آن حساب شود.