نوشته‌ی مسعود درویشیان

لینک مستقیم این مطلب در وب‌تارگت

## Overload کردن True و False

کلمات کلیدی true و false نیز می‌توانند به‌عنوان unary operators به‌منظور overload کردن مورد استفاده قرار گیرند. نسخه‌ی overload شده‌ی این operator ها با توجه به کلاسی که شما می‌سازید، شخصی‌سازی می‌شود. هنگامی‌که true و false برای یک کلاس overload می‌شوند، می‌توانید از اشیای آن کلاس برای کنترل کردن if، for، while و do-while و همچنین ? استفاده کنید.

Operator های true و false باید باهم overload شوند و نمی‌توانید فقط یکی از آن‌ها را overload کنید. هر دوی آن‌ها unary operator هستند و فرم کلی آن‌ها به‌صورت زیر است:

```
public static bool operator true(param-type operand)
{
    // return true or false
}

public static bool operator false(param-type operand)
{
    // return true or false
}
```

دقت کنید که هریک مقدار bool را return می‌کند.

مثال زیر نشان می‌دهد که چگونه true و false می‌توانند در کلاس TwoD اعمال شوند. فرض بر این است که اگر حداقل یکی از فیلدها غیر صفر باشد، شیء true است و اگر همه‌ی فیلدها صفر باشند، شیء false است:

```
using System;
class TwoD
{
    int X, Y;
    public TwoD()
    {
        X = Y = 0;
    }
    public TwoD(int a, int b)
    {
        X = a;
        Y = b;
```

```
    }
    public static bool operator true(TwoD op)
    {
        return (op.X != 0 || op.Y != 0);
    }
    public static bool operator false(TwoD op)
    {
        return (op.X == 0 && op.Y == 0);
    }
    public static TwoD operator --(TwoD op)
    {
        TwoD result = new TwoD();
        result.X = op.X - 1;
        result.Y = op.Y - 1;
        return result;
    }
    public static TwoD operator ++(TwoD op)
    {
        TwoD result = new TwoD();
        result.X = op.X + 1;
        result.Y = op.Y + 1;
        return result;
    }
    public void Show()
    {
        Console.WriteLine("{0}, {1}", X, Y);
    }
}
class OpOverloadingDemo
{
    static void Main()
    {
        TwoD ob = new TwoD(5, 5);

        if (ob)
            Console.WriteLine("ob is true");
        Console.WriteLine();

        for (; ob; ob--)
        {
            ob.Show();
        }

        Console.WriteLine();
        ob = new TwoD(-3, -3);

        while (ob)
        {
            ob.Show();
            ob++;
        }
    }
}
```

دقت کنید که چگونه از شیء TwoD برای کنترل if و while و for استفاده شده است. به‌عنوان مثال، در مورد if، شیء TwoD توسط true ارزیابی می‌شود. اگر حداقل یکی از فیلدهای این شیء مخالف صفر باشد، دستور if اجرا خواهد شد. در

مورد حلقه‌ها، حلقه تا زمانی اجرا می‌شود که شیء true باشد (فیلدهای شیء مخالف صفر باشد) و به‌محض این‌که شیء false شد (همه‌ی فیلدها صفر شدند) برنامه از حلقه خارج می‌شود.

### Overload کردن عملگرهای منطقی

همان‌طور که می‌دانید، سی‌شارپ شامل عملگرهای منطقی & ، | ، ! و && و || است. البته فقط & و | و ! می‌توانند overload شوند. با این حال با دنبال کردن یک سری قوانین می‌توانید از مزیت‌های && و || بهره ببرید.

اگر قصد استفاده از عملگرهای منطقی short-circuit را نداشته باشید، می‌توانید بسیار ساده & و | را overload کنید که هر کدام از آن‌ها یک مقدار bool را return می‌کند. overload کردن ! نیز مقدار bool را return می‌کند.

مثال زیر عملگرهای منطقی ! و & و | را overload می‌کند. مانند قبل فرض بر این است که شیء TwoD در صورتی true است که حداقل یکی از فیلدهای آن غیر صفر باشد و در صورتی‌که همه‌ی فیلدهای TwoD صفر باشند، شیء false است:

```
using System;
class TwoD
{
    int X, Y;
    public TwoD()
    {
        X = Y = 0;
    }
    public TwoD(int a, int b)
    {
        X = a;
        Y = b;
    }

    public static bool operator &(TwoD op1, TwoD op2)
    {
        return ((op1.X != 0 && op1.Y != 0) && (op2.X != 0 && op2.Y != 0));
    }
    public static bool operator |(TwoD op1, TwoD op2)
    {
        return ((op1.X != 0 || op1.Y != 0) || (op2.X != 0 || op2.Y != 0));
    }
    public static bool operator !(TwoD op)
    {
        return (op.X == 0 & op.Y == 0);
    }

    public void Show()
    {
        Console.WriteLine("{0}, {1}", X, Y);
    }
}
```

```
class OpOverloadingDemo
{
    static void Main()
    {
        TwoD a = new TwoD(1, 2);
        TwoD b = new TwoD(8, 8);
        TwoD c = new TwoD();

        Console.Write("Here is a: ");
        a.Show();
        Console.Write("Here is b: ");
        b.Show();
        Console.Write("Here is c: ");
        c.Show();
        Console.WriteLine();

        if(!a)
            Console.WriteLine("a is false");
        if(!b)
            Console.WriteLine("b is false");
        if(!c)
            Console.WriteLine("c is false");

        Console.WriteLine();

        if (a & b)
            Console.WriteLine("a & b is true");
        else
            Console.WriteLine("a & b is false");
        if(a & c)
            Console.WriteLine("a & c is true");
        else
            Console.WriteLine("a & c is false");

        Console.WriteLine();

        if(a | b)
            Console.WriteLine("a | b is true");
        else
            Console.WriteLine("a | b i false");
        if(a | c)
            Console.WriteLine("a | c is true");
        else
            Console.WriteLine("a | c is false");
    }
}
```

خروجی:

```
Here is a: 1, 2
Here is b: 8, 8
Here is c: 0, 0

c is false

a & b is true
a & c is false

a | b is true
a | c is true
```

در این روش، operator method های & و | و ! هرکدام یک مقدار bool را return می‌کنند. به‌دلیل این‌که از این operator ها در حالت استاندارد خودشان استفاده شود، return کردن مقدار bool ضروری است. روشی که در مثال قبل مشاهده کردید، فقط برای زمانی است که به short-circuit نیاز ندارید.

### فعال کردن operator های short-circuit

برای فعال کردن && و || باید از چهار قاعده پیروی کنید. یک، کلاس مورد نظر باید & و | را overload کند. دو، return type متدهای overload شده‌ی & و | باید از نوع همان کلاسی باشد که این operator ها در آن overload می‌شوند. سه، هر پارامتر باید به یک شیء از جنس همان کلاسی که operator در آن overload شده است، رجوع کند. چهار، operator های true و false باید برای کلاس مربوطه overload شود.

برنامه زیر نشان می‌دهد که چگونه & و | را برای کلاس TwoD اجرا کرده تا بتوان از مزیت && و || استفاده کنید:

```
using System;
class TwoD
{
    int X, Y;
    public TwoD()
    {
        X = Y = 0;
    }
    public TwoD(int a, int b)
    {
        X = a;
        Y = b;
    }

    // Overload & for short-circuit evaluation.
    public static TwoD operator &(TwoD op1, TwoD op2)
    {
        if ((op1.X != 0 && op1.Y != 0) && (op2.X != 0 && op2.Y != 0))
            return new TwoD(1, 1);
        else
            return new TwoD();
    }
    // Overload | for short-circuit evaluation.
    public static TwoD operator |(TwoD op1, TwoD op2)
    {
        if ((op1.X != 0 || op1.Y != 0) || (op2.X != 0 || op2.Y != 0))
            return new TwoD(1, 1);
        else
            return new TwoD();
    }

    // Overload true.
    public static bool operator true(TwoD op)
```

```
    {
        return (op.X != 0 || op.Y != 0); // at least one field is non-zero
    }
    // Overload false.
    public static bool operator false(TwoD op)
    {
        return (op.X == 0 & op.Y == 0); // all fields are zero
    }

    // Overload !.
    public static bool operator !(TwoD op)
    {
        return (op.X == 0 & op.Y == 0);
    }

    public void Show()
    {
        Console.WriteLine("{0}, {1}", X, Y);
    }
}
class OpOverloadingDemo
{
    static void Main()
    {
        TwoD a = new TwoD(5, 6);
        TwoD b = new TwoD(10, 10);
        TwoD c = new TwoD();

        Console.Write("Here is a: ");
        a.Show();
        Console.Write("Here is b: ");
        b.Show();
        Console.Write("Here is c: ");
        c.Show();
        Console.WriteLine();

        if (!a)
            Console.WriteLine("a is false");
        if (!b)
            Console.WriteLine("b is false");
        if (!c)
            Console.WriteLine("c is false");

        Console.WriteLine();

        if (a & b)
            Console.WriteLine("a & b is true");
        else
            Console.WriteLine("a & b is false");
        if (a & c)
            Console.WriteLine("a & c is true");
        else
            Console.WriteLine("a & c is false");

        Console.WriteLine();

        if (a | b)
            Console.WriteLine("a | b is true");
        else
            Console.WriteLine("a | b i false");
        if (a | c)
```

```
            Console.WriteLine("a | c is true");
        else
            Console.WriteLine("a | c is false");

        Console.WriteLine();

        // Now use short-circuit ops.
        Console.WriteLine("Use short-circuit && and ||");
        if (a && b)
            Console.WriteLine("a && b is true.");
        else
            Console.WriteLine("a && b is false.");

        if (a && c)
            Console.WriteLine("a && c is true.");
        else
            Console.WriteLine("a && c is false.");

        if (a || b)
            Console.WriteLine("a || b is true.");
        else
            Console.WriteLine("a || b is false.");

        if (a || c)
            Console.WriteLine("a || c is true.");
        else
            Console.WriteLine("a || c is false.");
    }
}
```

خروجی:

```
Here is a: 5, 6
Here is b: 10, 10
Here is c: 0, 0

c is false

a & b is true
a & c is false

a | b is true
a | c is true

Use short-circuit && and ||
a && b is true.
a && c is false.
a || b is true.
a || c is true.
```

در این‌جا نگاهی دقیق‌تر به برنامه‌ی بالا می‌اندازیم و می‌بینیم که چطور & و | اجرا شده‌اند. به این قسمت از کد بالا توجه کنید:

```
// Overload & for short-circuit evaluation.
public static TwoD operator &(TwoD op1, TwoD op2)
{
    if ((op1.X != 0 && op1.Y != 0) && (op2.X != 0 && op2.Y != 0))
        return new TwoD(1, 1);
    else
        return new TwoD();
```

```
}
// Overload | for short-circuit evaluation.
public static TwoD operator |(TwoD op1, TwoD op2)
{
    if ((op1.X != 0 || op1.Y != 0) || (op2.X != 0 || op2.Y != 0))
        return new TwoD(1, 1);
    else
        return new TwoD();
}
```

دقت کنید که هر دو یک شیء از جنس TwoD را return می‌کنند. توجه داشته باشید که چگونه این شیء به‌وجود آمده است. اگر خروجی عملیاتی که انجام شده true باشد، بنابراین یک شیء TwoD که true است (شیءای که حداقل یکی از فیلدهای آن غیر صفر است) return می‌شود. اگر خروجی false باشد، یک شیء false ساخته شده و سپس return می‌شود.

بنابراین در عباراتی به شکل زیر:

```
if (a & b)
        Console.WriteLine("a & b is true");
else
        Console.WriteLine("a & b is false");
```

خروجی a & b یک شیء TwoD است که در این مورد یک شیء true است. از آن‌جا که operator های true و false در این کلاس تعریف شده‌اند، شیءای که return شده است مربوط به true operator بوده و در نهایت یک مقدار بولین return می‌شود که در این‌جا این شیء true است و در نتیجه مقدار true نیز return می‌شود و محتویات دستور if اجرا خواهد شد.

به‌دلیل این‌که قوانین مورد نیاز رعایت شده‌اند، operator های short-circuit برای استفاده از اشیای TwoD نیز فعال هستند و به‌این صورت کار می‌کند که، اولین operand توسط operator true (برای ||) یا operator false (برای &&) بررسی می‌شود. اگر خروجی عملیات قابل تشخیص باشد، بنابراین & یا | مربوطه مورد ارزیابی قرار نمی‌گیرد. در غیر این‌صورت، از & یا | مربوطه برای یافتن خروجی استفاده می‌شود. بنابراین، استفاده از && یا || موجب می‌شود & یا | مربوطه تنها زمانی فراخوانی شود که operand اول نتواند خروجی را مشخص کند. برای نمونه، به کد زیر توجه کنید:

```
if (a || c) Console.WriteLine("a || c is true.");
```

در ابتدا true operator روی a اعمال می‌شود و از آن‌جا که در این مورد a برابر با true است، دیگر نیازی به استفاده از | operator نیست. اما در کد زیر:

```
if (c || a) Console.WriteLine(“a || c is true.”);
```

ابتدا true operator روی c اعمال می‌شود که در این مورد c برابر با false است. بنابراین، operator | فراخوانی خواهد شد تا بررسی شود که آیا a برابر با true است یا خیر (که در این مورد، a برابر با true است). شاید در ابتدا فکر کنید که تکنیک استفاده شده برای فعال کردن short-circuit مقداری پیچیده و دشوار باشد اما اگر اندکی در مورد آن فکر کنید متوجه خواهید شد که کارهای انجام شده منطقی به‌نظر می‌رسد. در حالت کلی، بهتر است از همین روش استفاده کنید.

## Conversion Operators

در برخی از مواقع، می‌خواهید از شیء یک کلاس در عبارتی استفاده کنید که شامل data type های دیگری نیز است. در بعضی موارد، overload کردن یک یا چند operator می‌تواند این کار را برای شما انجام دهد اما گاهی چیزی که شما نیاز دارید یک تبدیل ساده از نوع کلاس به نوع مورد نظرتان است. برای انجام این دسته از موارد، سی‌شارپ به شما اجازه می‌دهد نوع خاصی از operator method را بسازید که conversion operator نامیده می‌شود. Conversion operator یک شیء از کلاس شما را به نوع دیگری که مد نظرتان است تبدیل می‌کند.

دو حالت از conversion operator موجود است: implicit و explicit که فرم کلی آن‌ها به شکل زیر است:

```
public static operator implicit target-type(source-type v) { return value; }
public static operator explicit target-type(source-type v) { return value; }
```

در این‌جا، target-type مشخص کننده‌ی نوعی است که قصد دارید source-type را به آن تبدیل کنید و value مقدار کلاس، بعد از تبدیل است. Conversion operator اطلاعات را مطابق با target-type باز می‌گرداند (return می‌کند).

اگر conversion operator به‌طور implicit مشخص شود، بنابراین conversion به‌صورت اتوماتیک انجام خواهد شد، مثل حالتی که شیء در یک عبارت همراه با یک data type دیگری از نوع target-type در تعامل است. هنگامی‌که conversion به‌صورت explicit تعریف شده باشد، بنابراین هنگامی که cast مورد نیاز است conversion فراخوانی می‌شود. توجه کنید که نمی‌توانید برای یک source-type و target-type هم implicit و explicit را تعریف کنید.

برای نشان دادن conversion operator آن را در کلاس TwoD اعمال می‌کنیم. با فرض اینکه می‌خواهید یک شیء TwoD را به یک int تبدیل کنید و در یک عبارت از جنس int از آن استفاده نمایید:

```
using System;
class TwoD
{
```

```
    int X, Y;
    public TwoD()
    {
        X = Y = 0;
    }
    public TwoD(int a, int b)
    {
        X = a;
        Y = b;
    }
    public static implicit operator int(TwoD op)
    {
        return op.X * op.Y;
    }
}
class OpOvDemo
{
    static void Main()
    {
        TwoD ob1 = new TwoD(2, 2);
        TwoD ob2 = new TwoD(1, 1);

        int i = ob1 * 3;
        Console.WriteLine(i);
        i = ob1 - 3;
        Console.WriteLine(i);
        i = ob1 + ob2;
        Console.WriteLine(i);
        i = ob1;
        Console.WriteLine(i);
    }
}
```

همان‌طور که برنامه بالا نشان می‌دهد، هنگامی‌که از شیء TwoD در یک عبارت integer (مثل `i = ob1 - 3`) استفاده شده عملیات convert به‌طور اتوماتیک اعمال می‌شود.

ادامه‌ی مبحث conversion operators را در قسمت بعد دنبال کنید.

کلیه حقوق مادی و معنوی برای وب‌سایت وب‌تارگت محفوظ است.
استفاده از این مطلب در سایر وب‌سایت‌ها و نشریات چاپی تنها با ذکر و درج لینک منبع مجاز است.