

Forum:[Hack Forums](#)

Thread: ["+" and "-" operators override \[Resolved\]](#)

Mefie - November 18, 2015, 8:03 am

Hello!

If you read the title, you already read the question, but again: What do I need to write into my interface if I would like to require the classes implementing this interface to override "+" and "-" operators?

Thank you!

Update: OK. I have the following code

Code static DateTime operator +(DateTime c1, DateTime c2);

static DateTime operator -(DateTime c1, DateTime c2);

That I cannot put into an interface because

Quote:Error 1 Interfaces cannot contain operators ... 16 25 Calendar

How can I create an interface that requires classes to implement operator overload when an interface cannot contain operators?

+Haze - November 18, 2015, 8:07 am

I don't think you can override the functions of those operators at such a high level. In a lower level perspective then yes it's possible. But the question is, why would you want to do that?

Mouse? - November 18, 2015, 8:11 am

[https://msdn.microsoft.com/en-us/library/aa288467\(v=vs.71\).aspx](https://msdn.microsoft.com/en-us/library/aa288467(v=vs.71).aspx)

+Haze - November 18, 2015, 8:25 am

I guess I'm wrong then. Interesting post Mouse.

Odus - November 18, 2015, 8:35 am

(Nov 18, 2015 09:07 AM)+Haze Wrote: [\[CITE\]](#) But the question is, why would you want to do that? It can make life a little easier when you're working with objects, but I'd still go for a method rather than an overload.

Code public class Program

```
{
    static void Main(string[] args)
    {
        var r1 = new Rect(2, 4);
        var r2 = new Rect(4, 8);

        var r3 = r1 + r2;
        Console.WriteLine(r3.ToString());

        Console.ReadLine();
    }
}
```

```

public class Rect
{
    private int _x;
    private int _y;

    public Rect(int x, int y)
    {
        this._x = x;
        this._y = y;
    }

    public static Rect operator +(Rect r1, Rect r2) =>
        new Rect(r1._x + r2._x, r1._y + r2._y);

    public override string ToString() =>
        string.Format("X: {0}, Y: {1}", this._x, this._y);
}

```

Wanted to try out the immediate return syntax from Mouse's post, hadn't seen that until now.

Mefie - November 18, 2015, 8:38 am

(Nov 18, 2015 09:07 AM)+Haze Wrote: [\[CITE\]](#) I don't think you can override the functions of those operators at such a high level. In a lower level perspective then yes it's possible. But the question is, why would you want to do that?

Because I was asked by Uni to do it so ? Soon, I will get to know, but at the moment I do not have to know (I know it is related to the objects we will use, though) what we will use it for to require it with an interface.

(Nov 18, 2015 09:11 AM)?ouse Wrote: [\[CITE\]](#)
[https://msdn.microsoft.com/en-us/library/aa288467\(v=vs.71\).aspx](https://msdn.microsoft.com/en-us/library/aa288467(v=vs.71).aspx)

Thanks! I have just forgotten about this article. Actually, I remember, I found it not so long ago, but did not fully read because I had to do something and then did not return. Now, I read it and for second time I found what I wanted to know. Thank you!

Update:

(Nov 18, 2015 09:35 AM)Odus Wrote: [\[CITE\]](#) It can make life a little easier when you're working with objects, but I'd still go for a method rather than an overload.

```

Code public class Program
{
    static void Main(string[] args)
    {
        var r1 = new Rect(2, 4);
        var r2 = new Rect(4, 8);

        var r3 = r1 + r2;
        Console.WriteLine(r3.ToString());

        Console.ReadLine();
    }
}

```

```

public class Rect
{
    private int _x;
    private int _y;

    public Rect(int x, int y)
    {
        this._x = x;
        this._y = y;
    }

    public static Rect operator +(Rect r1, Rect r2) =>
        new Rect(r1._x + r2._x, r1._y + r2._y);

    public override string ToString() =>
        string.Format("X: {0}, Y: {1}", this._x, this._y);
}

```

Wanted to try out the immediate return syntax from Mouse's post, hadn't seen that until now. I was just about writing an own test code from it when I saw your post. Thank you, now I am sure how to do it ?

Update2: I have a new problem now... please check the thread update!

+Haze - November 18, 2015, 9:02 am

(Nov 18, 2015 09:35 AM)Odus Wrote: [CITE](#) It can make life a little easier when you're working with objects, but I'd still go for a method rather than an overload.

```

Code public class Program
{
    static void Main(string[] args)
    {
        var r1 = new Rect(2, 4);
        var r2 = new Rect(4, 8);

        var r3 = r1 + r2;
        Console.WriteLine(r3.ToString());

        Console.ReadLine();
    }
}

```

```

public class Rect
{
    private int _x;
    private int _y;

    public Rect(int x, int y)
    {
        this._x = x;
        this._y = y;
    }

    public static Rect operator +(Rect r1, Rect r2) =>

```

```
new Rect(r1._x + r2._x, r1._y + r2._y);

public override string ToString() =>
    string.Format("X: {0}, Y: {1}", this._x, this._y);
}
```

Wanted to try out the immediate return syntax from Mouse's post, hadn't seen that until now. Shidd, had no idea this was even possible. Seems pretty cool ngl. C# is evolved a lot.

Only3 - November 18, 2015, 1:00 pm

the problem with operators override is that its must be and used in the class

Mouse? - November 18, 2015, 1:13 pm

(Nov 18, 2015 02:00 PM)Only3 Wrote: [\[CITE\]](#) the problem with operators override is that its must be and used in the classWell that's one of if not the most common usage for operator overloading, so it works, at least for this.

Mefie - November 18, 2015, 10:43 pm

(Nov 18, 2015 02:00 PM)Only3 Wrote: [\[CITE\]](#) the problem with operators override is that its must be and used in the class

In the end, the solution was, I created a parent class of all object classes that could occur in use and implemented the operator overload in the parent class so that all of its children (and all objects that may occur) will have it. My code works now so it surely IS A solution, but if you have any more suggestion, I still am glad to listen ?