What actually Base Keyword does in Constructor initialization?

Ask Question

5

1

I have been confused by the `base` keyword in C#. Say I have some code like this:

```
class A
{
    // member declaration....
    A(int s, int t, int x)
    {
        // assign the values of members
    }
}

class B : A
{
    // member declaration....
    B(int a, int b, int c, int d) : base(a,b,c)
    {
        // does the base keyword play a role in
        // this constructor?
    }
```

```
}
```

Now what is the exact use of `base` keyword in `B` 's constructor? I have few questions to clarify:

1. Using the `base` keyword will call the base class `A` constructor(in our example)? Is this correct?
2. Providing the `base` keyword in derived class changes the behavior of the derived class constructor?

And basically when `base` keyword should be used in the constructor(some examples would be good)?

Edit:

I have another question. How can I inform the `base` class* about our `derived` class, via `base` keyword?

Thanks in advance.

c# constructor

edited Nov 17 '11 at 16:23
jwheron
2,406 2 24 39

asked Nov 17 '11 at 15:44
Ant's
6,442 16 82 132

3 Answers

Yes - `base` chains to a constructor in the base class.

6

If you *don't* specify `base` or `this` (to chain to another constructor in the *same* class), the effect is as if you had `base()` chaining to a parameterless constructor in the base class. In your example this would cause a compilation failure as your base class doesn't *have* a parameterless constructor.

See my article on constructors for more information.

answered Nov 17 '11 at 15:46

Jon Skeet
1096k 696 7985 8471

+1 for the link and explaining `this` keyword too ;) – Ant's Nov 17 '11 at 15:50

I have a doubt, so what is the basic use of informing the `base` class about our `derived` class, via `base` keyword? – Ant's Nov 17 '11 at 15:52

1 You aren't informing, you are extending the base constructor by adding additional functionality in the derived class and then aslo executing the functionality in the base constructor. – Maess Nov 17 '11 at 15:55

@Ant's: Part of the initialization of an instance of the derived class is running the initialization steps of the base class. Imagine if it *didn't* do that, but the constructor of `A` in your example was setting some fields - suddenly you'd be trying to use an instance of `B` (which *is* an instance of `A`) without it being completely initialized. – Jon Skeet Nov 17 '11 at 15:55

@JonSkeet: oh thanks for the answer.. :) – Ant's Nov 17 '11 at 16:04

4

I think the manual is pretty clear on `base` :

> The base keyword is used to access members of the base class from within a derived class:
>
> - Call a method on the base class that has been overridden by another method.
> - Specify which base-class constructor should be called when creating instances of the derived class.
>
> *

```
public class BaseClass
{
    int num;

    public BaseClass()
    {
        Console.WriteLine("in BaseClass()");
    }

    public BaseClass(int i)
    {
        num = i;
        Console.WriteLine("in BaseClass(int i)");
    }

    public int GetNum()
    {
        return num;
    }
}

public class DerivedClass : BaseClass
{
    // This constructor will call BaseClass.BaseClass()
    public DerivedClass() : base()
    {
```

```
    }

    // This constructor will call BaseClass.BaseClass(in
    public DerivedClass(int i) : base(i)
    {
    }

    static void Main()
    {
        DerivedClass md = new DerivedClass();
        DerivedClass md1 = new DerivedClass(1);
    }
}



/*
Output:
in BaseClass()
in BaseClass(int i)
*/
```

answered Nov 17 '11 at 15:48

CodeCaster
110k 17 148 200

3

1. Yes, this calls the base class constructor.
2. A derived constructor *always* calls a base class constructor; if you don't specify one, it will try to call the parameterless constructor (if one exists.) This syntax lets you pass parameters to a specific base class constructor. Note that, in this case, there is no public or protected base class constructor without parameters, so you need to use the 'base' syntax with parameters.

answered Nov 17 '11 at 15:47

Dan Bryant
24.3k 3 45 89

1 A derived constructor may end up calling it indirectly though, via another constructor in the same class, using `this` . – Jon Skeet Nov 17 '11 at 15:56

@Jon, good point; do you happen to know whether this is enforced at the jitter/verification level? I know you need to explicitly emit the IL call to the base class constructor, but I've never actually tried emitting a constructor that didn't make the call. – Dan Bryant Nov 17 '11 at 16:38

*