How do I use a property to initialize another property?

Asked 3 years, 3 months ago Active 3 years, 3 months ago Viewed 458 times

1

Why is this:

```
public int X { get; } = 5
public int Y { get; } = X;
```

not possible?

Because doing it manually:

```
public TestClass()
{
  X = 5;
  Y = X;
}
```

Works, and so does (obviously?) this:

```
public static int X { get; } = 5;
public static int Y { get; } = X;
```

Is there a way to get the first example to compile, or do I have to do it manually in the ctor?

(My real problem is far more complex, not just ints, but instances that are then being used to create other instances, but this example is easier to discuss)

c# constructor properties roslyn c#-6.0


5,270 1 25 46

"Because that's what the C# spec says" is the unfortunate answer "... it is a compile-time error to reference `this` in a variable initializer, as it is a compile-time error for a variable initializer to reference any instance member through a simple-name" – Damien_The_Unbeliever Jun 16 '16 at 12:00

@Damien_The_Unbeliever thanks, whats a simple-name? – Mafii Jun 16 '16 at 12:02

A simple name is basically reference to a member without explicitly saying `this.` before it. E.g. you're using a simple name to reference the `X` member in the second line of your first example. – Damien_The_Unbeliever Jun 16 '16 at 12:04

10.4.5.2 Instance field initialization – Steve Jun 16 '16 at 12:05

Does it necessarily have to be readonly properties backed by a field? One could argue that `Y` could be defined to just return the value of `X`. You don't necessarily need storage for it. – Jeff Mercado Jun 16 '16 at 20:59

2 Answers

6

The reason why this is not possible is that these initializations are done *before* the constructor is called. So it happens in a **static** context. The object is not yet fully initialized and there is no `this` reference yet. So you cannot access a *non-static* property like `X`.

For the same reason it works for the *static* properties in your third example.

So I don't see a workaround but doing this kind of initialization in a constructor.

edited Jun 16 '16 at 12:06

answered Jun 16 '16 at 12:00


René Vogt
35.7k 13 53 72

"these initializations are done before the constructor is called" Not true. They are incorporated in the constructor as far as I know. – Patrick Hofman Jun 16 '16 at 12:01

3 @PatrickHofman they are done before everything else in the constructor logic. The first thing done when constructing an object is reserving the memory for the object, then the initialization is done and then the constructor logic is executed. – Adwaenyth Jun 16 '16 at 12:02

@PatrickHofman I guess you are right regarding the code generated by the compiler, but from the developer's point of view it is *before* the constructor, at least before `this` is available, but `this` can be accessed in a constructor. – René Vogt Jun 16 '16 at 12:03

"Before anything else" isn't that much different than the other approach @Adwaenyth – Patrick Hofman Jun 16 '16 at 12:03

2

You cannot use the value of a non-static property `x` in an initializer expression of property `Y` for the same reason that you cannot use the value of a non-static field `x` in an initializer expression of field `y`, i.e.

```
public int x = 5;
public int y = x; // Not allowed
```

This triggers error CS0236, because compiler is allowed to decide on the order in which it processes field and property initializers for partial classes. Although `x` is guaranteed to be initialized ahead of `Y` when both properties are defined in the same file, there is no such guarantee for properties defined in different files containing the code of a partial class.

Compiler designers could implement it differently by allowing initializers to reference other fields and properties defined before the field or property being initialized, but the feature is not worth the trouble, because you can easily work around it by moving intialization into the constructor.

Doing the same inside a constructor does not present a problem, because you are in control of the order of assignments. When you say in the constructor that `x` must be initialized before `Y`, the compiler is not allowed to change that order.

edited Jun 16 '16 at 12:15

answered Jun 16 '16 at 12:03

dasblinkenlight

632k 64 850 1262

Indeed. I think the only real reason is order of execution. When you write a constructor by hand, you can force when what is called. If we let the compiler or CLR do that, we can get unexpected and non-deterministic behavior. – Patrick Hofman Jun 16 '16 at 12:04

2 "compiler is allowed to change the order in which it processes field and property initializers" - no it isn't. According to the C# spec, they're processed in textual order. – Damien_The_Unbeliever Jun 16 '16 at 12:05

3 (C# Spec, Version 5, Section 10.5.5.2): "The variable initializers are executed in the textual order in which they appear in the class declaration. " – Damien_The_Unbeliever Jun 16 '16 at 12:06

1 But it is possible to call a variable that isn't initialized yet. This might cause problems. If you don't allow that, you 'fix' that problem. – Patrick Hofman Jun 16 '16 at 12:07

2 (Re: your edit to partial classes. Whilst it is true that I cannot find a rule in the C# spec regarding how textual order is defined for partial classes, based on a quick read, it should be noted that this prohibition predates partial classes) – Damien_The_Unbeliever Jun 16 '16 at 12:11