Should I always/ever/never initialize object fields to default values?

Asked 15 years, 9 months ago Modified 9 years, 7 months ago Viewed 3k times

34

Code styling question here.

I looked at this question which asks if the .NET CLR will really always initialize field values. (The answer is *yes*.) But it strikes me that I'm not sure that it's always a good idea to have it do this. My thinking is that if I see a declaration like this:

```
int myBlorgleCount = 0;
```

I have a pretty good idea that the programmer expects the count to start at zero, and is okay with that, at least for the immediate future. On the other hand, if I just see:

```
int myBlorgleCount;
```

I have no real immediate idea if 0 is a legal or reasonable value. And if the programmer just starts reading and modifying it, I don't know whether the programmer meant to start using it before they set a value to it, or if they were expecting it to be zero, etc.

On the other hand, some fairly smart people, and the Visual Studio code cleanup utility, tell me to remove these redundant declarations. What is the general consensus on this? (Is there a consensus?)

I marked this as language agnostic, but if there is an odd case out there where it's specifically a good idea to go against the grain for a particular language, that's probably worth pointing out.

EDIT: While I did put that this question was language agnostic, it obviously doesn't apply to languages like C, where no value initialization is done.

EDIT: I appreciate John's answer, but it is exactly what I'm **not** looking for. I understand that .NET (or Java or whatever) will do the job and initialize the values consistently and correctly. What I'm saying is that if I see code that is modifying a value that hasn't been previously explicitly set in code, I, as a code maintainer, don't know if the original coder meant it to be the default value, or just forgot to set the value, or was expecting it to be set somewhere else, etc.

language-agnostic coding-style

Share
Improve this question
Follow

edited May 23, 2017 at 11:46
Community Bot
1 • 1

asked Mar 11, 2009 at 19:59

Beska
12.7k • 14 • 79 • 113

17 Answers

Sorted by: Highest score (default)

26

Think long term maintenance.

- Keep the code as explicit as possible.
- Don't rely on language specific ways to initialize if you don't have to. Maybe a newer version of the language will work differently?
- Future programmers will thank you.
- Management will thank you.
- Why obfuscate things even the slightest?

Update: Future maintainers may come from a different background. It really isn't about what is "right" it is more what will be easiest in the long run.

Share Improve this answer Follow

answered Mar 11, 2009 at 20:32

Anders Hansson
3,796 • 3 • 30 • 27

6 For the future maintainers: first day they ask, tell them what world they're in. End of problem. And we're not talking about a language feature. It's a *platform* feature. It's true for evey .NET language, and always will be true. *Think* what would happen if this behavior changed! – John Saunders Mar 11, 2009 at 20:45

2 That was not really the point was it? Not all software is .NET? Or am I living in a parallel universe? – Anders Hansson Mar 11, 2009 at 20:47

3 John: The maintainer may understand the issue perfectly, but how do they know that the coder meant to get the default when they didn't initialize the value? Maybe it was supposed to be the default, but maybe not. After all, if everything was perfect with the code, they wouldn't be maintaining it. – Beska Mar 11, 2009 at 20:57

1 @Subtwo: I would hope that the intention of the original programmer was confirmed by automated testing and by human QA. That way, if the programmer left the field uninitialized, but meant for it to be initialized to something else, we'd know about it and keep knowing about it. Otherwise, assume the original programmer and his managers and those doing code reviews, collectively knew they were living in a world where fields are always initialized to their default values. – John Saunders May 10, 2009 at 2:13

2 @John Saunders: Well... Not all projects are following good practice regarding testing, etc. Again - I can't really see the benefit of holding back on clarity. If you can think of a case where it would be more clear to actually leave fields uninitialized please inform me, I would like to be enlightened. – Anders Hansson May 18, 2009 at 15:03

14

You are always safe in assuming the platform works the way the platform works. The .NET platform initializes all fields to default values. If you see a field that is not initialized by the code, it means the field is initialized by the CLR, not that it is uninitialized.

This concern is valid for platforms which do not guarantee initialization, but not here. In .NET, is more often indicates ignorance from the developer, thinking initialization is necessary.

Another unnecessary hangover from the past is the following:

```
string foo = null;
foo = MethodCall();
```

I've seen that from people who should know better.

Share Improve this answer Follow

answered Mar 11, 2009 at 20:05

John Saunders
162k ● 26 ● 249 ● 402

2 In C# (and probably VB.NET) the compiler will throw an error if you try to read a variable that has never been set in code. This alone prevents a situation like you describe, since the variable must always be set somewhere prior to reading from it, even if not in the declaration. – Chris Mar 11, 2009 at 20:38

1 You're missing the point that the value is always set, under all circumstances. There's no point in imagining that something different might happen. It can never be a bug unless you're dealing with a really stupid developer who expects "int i;" to init to 1. Fire him, problem solved. – John Saunders Mar 11, 2009 at 20:38

1 @Beska: in general, we always have to hope our colleagues know what they're doing. We test our code to prove it. The code itself isn't the place to prove this. You don't want to put an Assert(i == 2) after every i = 1+1. It's always true. – John Saunders Mar 11, 2009 at 22:06

1 @John having been a maintenance coder, I can safely say that people who know what they're doing (even exceptionally good people) can overlook simple mistakes like this. In some cases, assuming that they knew what they were doing is detrimental, because it leads to assumptions about their behaviour. – DevinB Mar 16, 2009 at 18:36

2 (cont) In the case of explicit declaration, there are no assumptions. I know exactly what the previous coder wanted the value to be. – DevinB Mar 16, 2009 at 18:39

4

I think that it makes sense to initialize the values if it clarifies the developer's intent.

In C#, there's no overhead as the values are all initialized anyway. In C/C++, uninitialized values will contain garbage/unknown values (whatever was in the

memory location), so initialization was more important.

Share Improve this answer Follow

answered Mar 11, 2009 at 20:04

Ryan Emerle

15.8k ● 8 ● 54 ● 70

FxCop has a rule under the Performance section that specifically calls out unnecessary initialization. There must be some waste if they tell you not to do it. Not that it matters in most real-world applications... – Pedro Mar 11, 2009 at 20:10

1 Exactly...my code examination tool has a similar rule. But I'm not clear on exactly why...especially since I've been told (but have not verified) that it usually does not actually make any waste...that the redundant initialization is optimized out. – Beska Mar 11, 2009 at 20:22

4

I think it should be done if it really helps to make the code more understandable.

But I think this is a general problem with all language features. My opinion on that is: **If it is an official feature of the language, you can use it.** (Of course there are some anti-features which should be used with caution or avoided at all, like a missing `option explicit` in Visual Basic or diamond inheritance in C++)

There was I time when I was very paranoid and added all kinds of unnecessary initializations, explicit casts, über-paranoid try-finally blocks, ... I once even thought about ignoring auto-boxing and replacing all occurrences with explicit type conversions, just "to be on the safe side".

The problem is: There is no end. You can avoid almost all language features, because you do not want to trust them.

Remember: It's only magic until you understand it :)

Share

Improve this answer

Follow

edited Jul 2, 2012 at 7:12

answered Mar 11, 2009 at 20:12

Daniel Rikowski

72.4k ● 62 ● 256 ● 336

1 Sorry, but I take issue with "If it is an official feature of the language, you should use it." Diamond inheritance in C++ makes a wonderful counterpoint, just for starters. – Not Sure Mar 11, 2009 at 20:18

1 Yes, I understand what you mean, but I think that is over-interpreting the sentence. Of course it is not always good to do everything which is possible. Anyway, I changed it to "can". – Daniel Rikowski Mar 11, 2009 at 21:06

@NotSure: After reading your comment again three years later, I noticed that what you are saying has no relevance for my answer. What I'm saying is: If a language has the diamond inheritance feature and you want to rely on that diamond inheritance feature with all its ugly

consequences, you should not re-implement it yourself but rather use the existing feature. – Daniel Rikowski Jul 2, 2012 at 7:16

@DanielRikowski It's not a matter of not trusting the framework feature. There is no doubt in my mind that the value will be initialized by the framework, regardless of what the developer does. The question is for code clarity. If I'm trying to track down a bug in someone's code, and they have started using this variable without explicitly initializing it, that's a possible red flag for me; I may think that 0 might not be a valid value for this variable and I have to read more code to find out. If the programmer has explicitly set it to 0, I immediately know "0 is a valid value; move on." – Beska Jul 16, 2012 at 21:21

@Beska: That's exactly what I'm saying in the first sentence. :) – Daniel Rikowski Jul 17, 2012 at 9:51

2

I agree with you; it may be verbose, but I like to see:

```
int myBlorgleCount = 0;
```

Now, I always initial strings though:

```
string myString = string.Empty;
```

(I just hate null strings.)

Share Improve this answer Follow

answered Mar 11, 2009 at 20:03

Richard Morgan

7,681 ● 9 ● 50 ● 87

1

In the case where I cannot immediately set it to something useful

```
int myValue = SomeMethod();
```

I will set it to 0. That is more to avoid having to think about what the value would be otherwise. For me, the fact that integers are always set to 0 is not on the tip of my fingers, so when I see

```
int myValue;
```

it will take me a second to pull up that fact and remember what it will be set to, disrupting my thought process. For someone who has that knowledge readily available, they will encounter

```
int myValue = 0;
```

and wonder why the hell is that person setting it to zero, when the compiler would just do it for them. This thought would interrupt their thought process.

So do which ever makes the most sense for both you and the team you are working in. If the common practice is to set it, then set it, otherwise don't.

Share Improve this answer Follow

answered Mar 11, 2009 at 20:23

Ross Goddard

4,302 ● 4 ● 28 ● 24

1

In my experience I've found that explicitly initializing local variables (in .NET) adds more clutter than clarity.

Class-wide variables, on the other hand should always be initialized. In the past we defined system-wide custom "null" values for common variable types. This way we could always know what was uninitialized by error and what was initialized on purpose.

Share Improve this answer Follow

answered Jul 28, 2009 at 16:01

JonnyD

591 ● 1 ● 6 ● 12

1

I always initialize fields explicitly in the constructor. For me, it's THE place to do it.

Share Improve this answer Follow

answered Dec 21, 2009 at 12:38

bloparod

1,726 ● 1 ● 14 ● 17

0

I think a lot of that comes down to past experiences.

In older and unamanged languages, the expectation is that the value is unknown. This expectation is retained by programmers coming from these languages.

Almost all modern or managed languages have defined values for recently created variables, whether that's from class constructors or language features.

For now, I think it's perfectly fine to initialize a value; what was once implicit becomes explicit. In the long run, say, in the next 10 to 20 years, people may start learning that a default value is possible, expected, and known - especially if they stay consistent across languages (eg, empty string for strings, 0 for numerics).

Share Improve this answer Follow

answered Mar 11, 2009 at 20:03

Robert P

16k ● 11 ● 73 ● 116

Except in some languages, like Java, an empty string, "", is actually a String. You should instead initialize a String variable to null so you don't create a String object unnecessarily. – David Koelle Mar 11, 2009 at 20:06

Yup, exactly my point: Right now, defaults are all but certain across languages. :) – Robert P Mar 11, 2009 at 20:52

@David, the same is true in C#. (If you write `var x = "";` you have just created a new string object.) – drwatsoncode Nov 7, 2013 at 4:15

0

You Should do it, there is no need to, but it is better if you do so, because you never know if the language you are using initialize the values. By doing it yourself, you ensure your values are both initialized and with standard predefined values set. There is nothing wrong on doing it except perhaps a bit of 'time wasted'. I would recommend it strongly. While the commend by John is quite informative, on general use it is better to go the safe path.

Share Improve this answer Follow

answered Mar 11, 2009 at 20:07

Sebastian Oliva

355 ● 3 ● 7

Since he was talking about .NET, leaving off the redundant initializations *is* the safe path, as the fields are always initialized. If he is not using .NET, then the answer depends on the platform. In the case of unmanaged C/C++, it's necessary. – John Saunders Mar 11, 2009 at 20:11

I usually do it for strings and in some cases collections where I don't want nulls floating around. The general consensus where I work is "Not to do it explicitly for value types."

0

Share Improve this answer Follow

answered Mar 11, 2009 at 20:11

SharePoint Newbie

6,084 ● 12 ● 67 ● 103

I wouldn't do it. C# initializes an int to zero anyways, so the two lines are functionally equivalent. One is just longer and redundant, although more descriptive to a programmer who doesn't know C#.

0

Share Improve this answer Follow

answered Mar 11, 2009 at 20:12

James Orr

5,105 ● 7 ● 43 ● 63

This is tagged as language-agnostic but most of the answers are regarding C#.

In C and C++, the best practice is to **always** initialize your values. There are some cases where this will be done for you such as static globals, but there shouldn't be a performance hit of any kind for redundantly initializing these values with most compilers.

0

Share Improve this answer Follow

answered Mar 11, 2009 at 20:18

Dan Olson

23.3k ● 4 ● 43 ● 56

1 He's asking about .net languages where the initialisation is guaranteed. – Ray Mar 11, 2009 at 20:23

Well, not .NET necessarily, but any language where some kind of default initialization is guaranteed. The question is: despite this guarantee, should the value be specifically initialized anyway? – Beska Aug 4, 2009 at 19:35

I wouldn't initialise them. If you keep the declaration as close as possible to the first use, then there shouldn't be any confusion.

0

Share Improve this answer Follow

answered Mar 11, 2009 at 20:22

Ray
46.5k • 29 • 127 • 170

0

Another thing to remember is, if you are gonna use automatic properties, you have to rely on implicit values, like:

```
public int Count { get; set; }
```

Share Improve this answer Follow

answered May 10, 2009 at 1:53

Joan Venge
330k • 222 • 493 • 705

0

http://www.geekherocomic.com/2009/07/27/common-pitfalls-initialize-your-variables/

Share Improve this answer Follow

answered Jul 28, 2009 at 15:52

leppie
117k • 18 • 200 • 300

0

If a field will often have new values stored into it without regard for what was there previously, and if it should behave as though a zero was stored there initially but there's nothing "special" about zero, then the value should be stored explicitly.

If the field represents a count or total which will never have a non-zero value written to it directly, but will instead always have other amounts added or subtracted, then zero should be considered an "empty" value, and thus need not be explicitly stated.

To use a crude analogy, consider the following two conditions:

1. `if (xposition != 0) ...
2. `if ((flags & WoozleModes.deluxe) != 0) ...

In the former scenario, comparison to the literal zero makes sense because it is checking for a position which is semantically no different from any other. In the second scenario, however, I would suggest that the comparison to the literal zero adds nothing to readability because code isn't really interested in whether the value of the

expression `(flags & WoozleModes.deluxe)` happens to be a number other than zero, but rather whether it's "non-empty".

I don't know of any programming languages that provide separate ways of distinguishing numeric values for "zero" and "empty", other than by not requiring the use of literal zeros when indicating emptiness.

answered May 15, 2015 at 23:27

supercat

80.8k ● 9 ● 174 ● 220