Stop Using VAR Everywhere And Think Before Using Underscore With Private Variable In C#

Be a Thinker First, Then a Doer

Coverage Topic

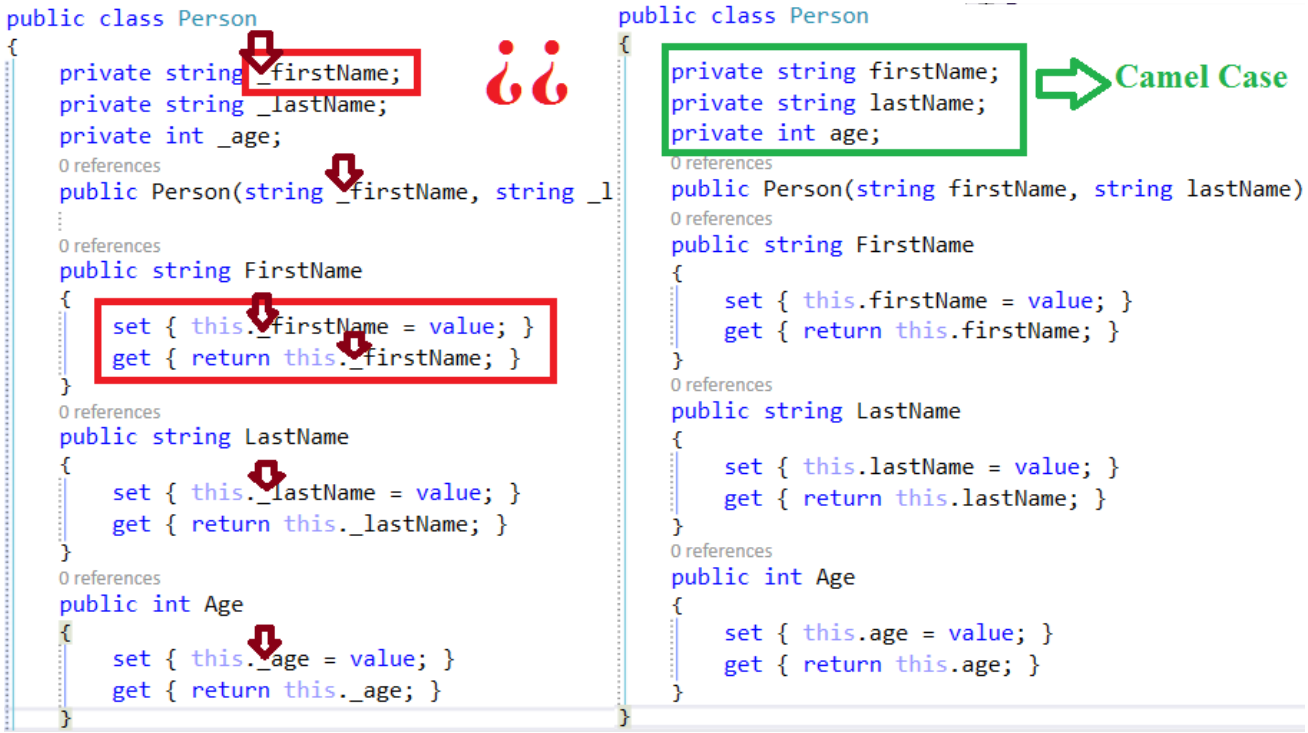
- Underscore(_) or not underscore with the *private* member variable
- Golden Hammer - Anti-Pattern
- Personal Preference vs. Best Practice
- Best Practice Guidance
- Shouldn't use *var* everywhere
- When to use '*var*'
- When '*var*' can't be used

Let's Drill Down the Basic Concept

The Problem

some people use underscore with *private* variables and some peoples use *camel case* with *private* variables. Another issue he knows about is why *var* was introduced; but now people are using "*var*" everywhere as a data [illegible] # as a programming language and .NET as a development platform.

Naming Convention for Private Member Variables



Question

Adam asked me, which convention is standard with underscore or without underscore according to the best practice guidance?

Answer

have their own preference.

But Adam was not satisfied with the answer. Because, if everybody follows his/her own prefe[illegible]eview and maintain for another person.

Few programing languages are *case-insensitive*, for example, say, Visual Basic (old version). In the VB, there is no difference between pascal case (i.e., ***V****ariableName*) and camel case (i.e., ***v****ariableName*). Therefore, they have to use underscore with the *private* member variable (i.e, *_variableName*).

C# is a case sensitive programing language and it knows the difference between pascal-case (*VariableName*) and camel-case (*variableName*).

Therefore,

- If C# is a case sensitive programing language, then why am I using underscore (_) prefix before *private* member variable?
- What is the extra benefit to use the underscore(_) prefix before *private* member variable?
- Am I a doer only? Few people are following an exceptional convention and so why am I using the same convention?
- Is it the beauty of readability?
- In the best practice guidelines, it is clearly suggested that using camel-case for the private variable, so then what is the problem to follow the general guidelines?
- If people use the underscore with the *private* variables, then this is an exceptional case. Exception is not an example for the best practice, then why aren't they thinking out of the box?

Solution

Be a THINKER first, then a DOER.

Real-life Investigation

- In C++, underscore was used for *private* variables. Hungarian notation(i.e, '*m_*') was used as a prefix with member variables for MFC. But current practice states "*Don't use underscores at-all*".

- Only use underscore, if you are writing unit testing methods. In BDD naming convention, underscore is used with test methods to make it more readable.

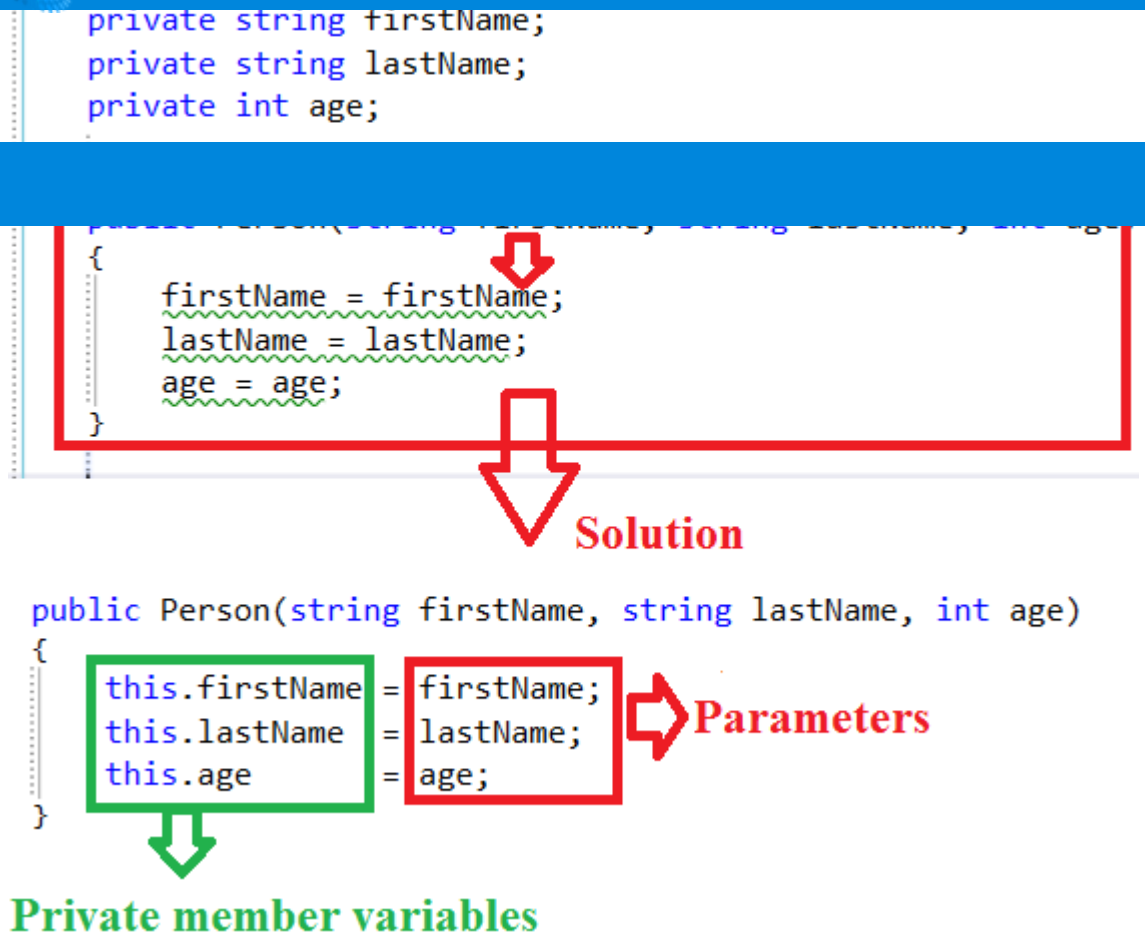
Given_Zero_Values_As_Params_When_GetSum_IsCalled_Then_It_Should_Throw_Invalid_Argument_Exception.

- If I use any intelligence tool and if it doesn't have best practices as a default (say default is underscore with private variable), then there is an option to change the default settings.
- I know programing very well to solve any problem quickly. I have implemented hundreds of applications. But if I used to write code according to my personal preference, then it doesn't mean that I know all of the best practices.

  Remember, tomorrow doesn't come; but tomorrow NEVER dies.

Underscore vs. this

- If I use same name for the *private* variables and *parameters*, then I have to use '`this`' with *private* variable.
- This is good practice, if you use '*this*' for all references of the *private* variables.



Underscore is a Thorn for Private Variable

- Doctor sometimes uses drugs as a treatment to the patient; but now if general people start to use the same drug for their pleasure, then this is illegal; because it is harmful for the human body.
- If I'm in the Amazon rainforest as a survivor game changer, then it's okay to eat some raw food because, there is no alternative option. It doesn't mean that - in my real life, every time I will eat that raw food.

Similarly, I'm writing code using VB (old) or C++. So, in VB, there is no difference between Pascal case (i.e, ***V****ariableName*) and camel case (i.e., ***v****ariableName*). Therefore, I am using underscore with the *private* member variable (i.e., _***v****ariableName*). It

Therefore the most worst excuse,

- I'm used to use underscore with the *private* variable

- I don't need to write '*this*' with *private* variables.
- I don't need to use '*this*' when both *private* variables and parameters have the same name or bla- bla reasons.

Moral Points

I know, some programing languages use underscore. Because, they have some explanation for that.

- Now I want to fit the underscore everywhere. Even I know that C# doesn't have any limitation that it needs to use underscore with *private* variables. But I want to fit it *by hook or by crook*.
- This is my personal preference and I'm writing it only for me; in future, nobody needs to maintain it.
- I don't need to worry about the other team members or the best practice guidance.
- Even I am leading a team and I am advising the team members to do the worst practice like me.

These kind of immature thoughts are one kind of anti-pattern such as GOLDEN HAMMER.

I am not writing the code only for the machine or myself. If so, then it doesn't need modern languages, in general, it just knows only *0* and *1*.

Remember, I am writing code for humans; so that different people can maintain it. If I write code for me only, then I don't need to follow any best practices. In these case, my personal preference is enough.

Golden Hammer - Anti-Pattern

"*If all you have is a hammer, everything looks like a nail.*"

I'm using particular technology, tools, methodology or architecture to solve all kinds of problems; although I know that there are alternative and better solutions to solve that problem, only because, I'm familiar and used to it.

Definition of best practices:

"*A procedure or set of procedures that is preferred or considered standard within an organiza*

languages to the world? We all know the answer, I'm avoiding it to make it short.

> *If you follow the best practice guidelines, then there is no question at-all. Although, there are some exceptions and they are used to it. They don't care about the best practice. But exception is not the best example to follow. If you follow that, then sometimes be ready to face "okay, but ...".*

Best Practice

- Don't use Hungarian notation
- Don't use an Underscore prefix for *private* member variables.
- Do use camel casing for private member variables
- Do use '*this*' for all references of the *private* variables.(For VB, use '*me*')

> *Keep personal preferences aside, pick the best technique to solve the problem.*

General naming convention from Microsoft - *"Don't use underscore and Hungarian notation"*.

References

First Reference

Second Reference

Improper Use of 'VAR' Everywhere

Have Questions

- In the above example, is there the beauty of readability?
- Why am I using *var* for a simple declaration?
- Why can't I say that, these kinds of improper use, are an anti-pattern?

Investigation to var or not to var

'*var*' should not be used in simple declarations, as shown below:

When Must 'var' be Used

- Should be used in *LINQ*
- Should be used with *anonymous types*.

When Can't It Be Used

statements.

- As a type of field
- As a type of parameter

Solution

First, be a thinker, then a doer.

Anti-Pattern using VAR Everywhere

Why was *var* introduced? It was introduced for LINQ, anonymous types, etc.

Now if I use it everywhere, then it is one kind of anti-pattern such as Golden Hammer. So, avoid the anti-pattern. Because, it is like a thorn for the software development. It is better to avoid type guessing.

PLease note:

I apologize, if I hurt your feelings. This is mainly a focus on the best practices in C# language. Few people who are used to their personal preference, this article is not pointing at them. This is general guidance for new developers.

Next Recommended Article
Guide to Improving Code Performance in .NET: Part I

Hr Rony TOP 1000

Lazy software engineer in web/mobile application development, architecture design, and automated unit testing. Don't like study;

701 304.1k 1

8 14

Type your comment here and press Enter Key (Minimum 10 characters)

I have always typed private members without underscore. Lately I'm mostly writing wpf applications which means I have a lot of properties with private backing fields. It happened a couple times that I accidently typed the backing field instead of the property in my code. It's sometimes hard to spot these bugs so I'm actually considering using underscores for private members.

Qubie1 Qubie1

1892 10 0

May 16, 2019

0 0 Reply

The core FX standar states: We use _camelCase for internal and private fields and use readonly where possible. Prefix internal and private instance fields with _, static fields with s_ and thread static fields with t_. When used on static fields, readonly should come after static (e.g. static readonly not readonly static). Public fields should be used sparingly and should use PascalCasing with no prefix when used.

Antonio Ortiz

1899 3 0

Feb 01, 2019

0 0 Reply

But what about the corefx standard? https://github.com/dotnet/corefx/blob/master/Documentation/coding-guidelines/coding-style.md

Antonio Ortiz

1899 3 0

Feb 01, 2019

0 0 Reply

To me, in the beginning the entire introduction of the var-keyword in a language like C# felt like making it a kind of JavaScript, a language in which explicit type declarations don't even exist (with all the disadvantages that it has), and in which every variable is declared using that "var" word. But I got used to it after some years (for years I have never used it because I actually hated it). I still think that explicit type declaration is a thing of quality. I have never considered it a "boiler plate" that should be removed. Furthermore, I

generation was not available in that way yet) now defaults to auto generating property back field generate a "full" property)! Is Microsoft itself changing standards?

Eduard De Jong

Jan 19, 2019

documentation. It saves you from frivolous *over specification*. As you can obviously tell that "my String" is a string. Why write string? It's pointless in my opinion. I will agree that it is overused, and in the case of var someVal = getSomeVal(), I will concede that this is a BAD use case of var as I have no clue what someVal actually is. I would not call using var to simplify self documenting code as a *bad practice*! ~my humble opinion.

Marcus J

1900 2 0

Oct 31, 2018

1 1 Reply

Best practice is a general communication guideline for each other. I know, I don't need to explain; because, you know why we need that. If you wanna customize to make your best without making communication problem then you're good.

Hr Rony

701 2.1k 304.1k

Nov 02, 2018

0

The approach to encourage to use "this" vs "underscore" is interesting, but in many points it feels like you're fighting against yourself. You call "excuses" to points that you're not discussing at all: being a private variable is a very atractive reason to use "underscore" for some reason that you already mention: clear distinction between private vs parameters. You encourage that "this" is better for that purpose, despite its verbosity. There is also a difference on intellisense, both of them have pros and cons. I dont see a clear unbiassed analysis on it. For example what about code density? And as a bonus, you mention the "var disadvantages". Why to use var? Because you dont need to know the type, only the context: var a = 2 + 2; means 4, doesn't matter if it is a real, double, int or byte. var name = "Pepe"; more of the same. A name is a name, you dont need to know it is a string or not. var person = GetPersonById(); better yet: a person is a person and we should use good variable naming to focus on the business context.

Zam Eb

1900 2 0

Jun 27, 2018

2 2 Reply

Agree. Thank you for your comment.

Hr Rony

701 2.1k 304.1k

Jul 19, 2018

0

I have to disagree on the use of var as in: var A = 2 + 2 The compiler will make the determination for you - you may wish A to be a decimal type, not an integer. Using var for numbers is horrible practice and leads to ambiguity (just put in what you want - no need to worry what the compiler's guess is).

Hi, I just found the article which says when the type of a variable is clear from the context, use va… r1 = "This is clearly a string "; Reference: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-

Saravanan V
1062 971 129.2k
2 2 Reply

Thank you for your question.Many developers like to use var. I have no complains for that. But think for another people who will review your codes for maintenance purpose. See the bellow example:Var data = object.GetData(); Now tell me, what is the type of the data? Now you have TWO options for the answer of that question 1. Ask to the developer who wrote the code, OR 2. Take help from the intelligence tool. It means you put a mask on your face. And if anybody needs to know about you, then he should call somebody to tell your identity. It means he is not independent to tell your identity. Every time he needs assistance. So, if you are a code reviewer then it will kill your time. Look back, once. string.Empty was faster than double quate("") in framework 1.1. But in framework 2.0 Microsoft has fixed that issue. But still string.Empty is popular than double quote (""), because of the readability and people suggest to use string.Empty. So, framework will be changed time to time; but good practice never changes. It just improves its standard. So, it is better to avoid type guessing. Don't think about for today only; think about for tomorrow too. My personal opinion, you can follow anything whatever you like most. This is your personal preference.

Hr Rony
701 2.1k 304.1k
Jun 19, 2017
3

Thanks for your detailed answer. In general, developers would use the Go To Definition option or mouse over the method to find out the return type. I understand your on view on good practice and agree with you. And, I'm personally following your articles...

Saravanan V
1062 971 129.2k
Jun 22, 2017
1

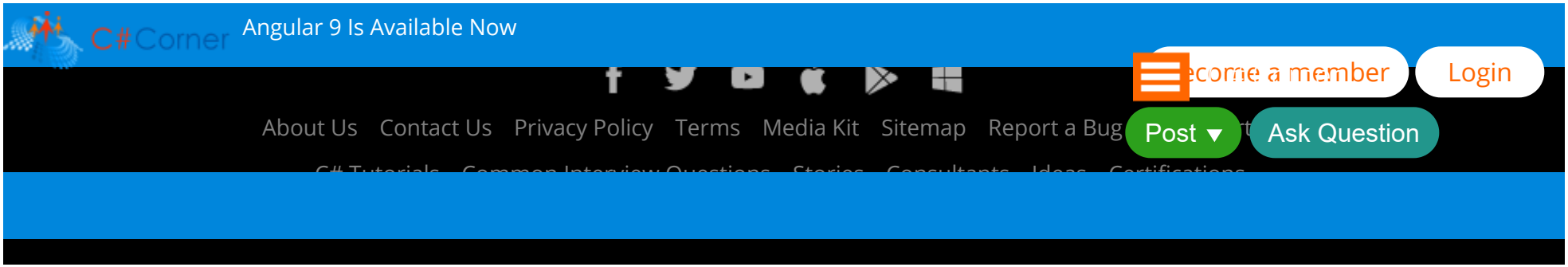
It's more than camelCase which is for Parameters it is CamelCase unless a Property requires a private backing Field. Think about it.

matthew sheeran
1857 45 0
Jun 08, 2017
2 0 Reply

An amazing read by the way.I enjoyed it seriously.

Vipin Tyagi
727 2k 421.1k
Jun 07, 2017
2 0 Reply

TRENDING UP


C#Corner
Angular 9 Is Available Now
ecome a member
Login
About Us
Contact Us
Privacy Policy
Terms
Media Kit
Sitemap
Report a Bug
Post
Ask Question