✓ 200 XP

Choose variable names that follow rules and conventions

6 minutes

A software developer once famously said, "The hardest part of software development is naming things." Not only does the name of a variable have to follow certain syntax rules, it should also be used to make the code more human-readable and understandable. That's a lot to ask of one line of code!

Variable name rules

There are some variable naming rules that are enforced by the C# compiler.

Variable name rules

- Variable names can contain alphanumeric characters and the underscore character. Special characters like the pound `#`, the dash `-`, and the dollar sign `$` are not allowed.
- Variable names must begin with an alphabetical letter or an underscore, not a number. Developers use the underscore for a special purpose, so try to not use that for now.
- Variable names must NOT be a C# keyword. For example, these variable name declarations won't be allowed: `float float;` or `string string;`.
- Variable names are case-sensitive, meaning that `string MyValue;` and `string myValue;` are two different variables.

Variable name conventions

Conventions are suggestions that are agreed upon by the software development community. While you're free to decide not to follow these conventions, they're so popular that it might make it difficult for other developers to understand your code. You should practice adopting these conventions and make them part of your own coding habits.

Variable name conventions

- Variable names should use **camel case**, which is a style of writing that uses a lower-case letter at the beginning of the first word and an upper-case letter at the beginning of each subsequent word. >For example: `string thisIsCamelCase;`.
- Variable names should be descriptive and meaningful in your application. You should choose a name for your variable that represents the kind of data it will hold (not the data type). >For example: `bool orderComplete;`, NOT `bool isComplete;`.
- Variable names should be one or more entire words appended together. Don't use contractions because the name of the variable may be unclear to others who are reading your code. >For example: `decimal orderAmount;`, NOT `decimal odrAmt;`.
- Variable names shouldn't include the data type of the variable. You might see some advice to use a style like `string strMyValue;`. It was a popular style years ago. However, most developers don't follow this advice anymore and there are good reasons not to use it.

The example `string firstName;` follows all of these rules and conventions, assuming I want to use this variable to store data that represents someone's first name.

Variable name examples

Here's a few examples of variable declarations (using common data types):

C#

```csharp
char userOption;

int gameScore;

float particlesPerMillion;

bool processedCustomer;
```

Other naming conventions

The rules and conventions described above are for *local variables*. A **local variable** is a variable that is scoped within the body of a method, or a variable in a console application that uses top-level statements (like the code in this module).

There are other types of constructs that you can use in your applications, and many have their own conventions. For example, classes are often used in C# programming. Classes support *fields*, which are members of a class that act like variables inasmuch that they store values, or rather, state. Classes also support *accessibility modifiers*, which allow some values to be *private* or *public*. A *private* member can only be referenced by other members in the same class. A *public* member can be referenced outside of the class. So, you can create *private fields* or *public fields*. Although you won't be creating classes in this module, it's important for you to know that the naming conventions you just learned about fit into a larger naming framework.

Check your knowledge

1. Which of the following is an example of correct camel case formatting of a variable name? *

○ UserOption.

○ user Option.

○ userOption.

Check your answers

Module complete:

Unlock achievement