# C# Coding Standards

Coding conventions serve the following purposes:

* They create a consistent look to the code, so that readers can focus on content, not layout.
* They enable readers to understand the code more quickly by making assumptions based on previous experience.
* They facilitate copying, changing, and maintaining the code.
* They demonstrate C# best practices.

These conventions have been adapted from the c# conventions recommended by Microsoft at <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions>.

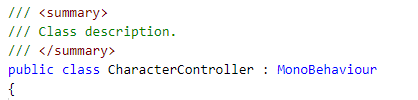
## Naming Conventions:

Use meaningful names for namespace, classes, variables, properties and methods of every part of your class. Avoid using abbreviations. Name your source files after their main classes and no space or special characters in file/folder names.

Always document your class as you create it while the intention is fresh in your mind. If you are using Visual Studio you can do auto initiate a document block by hitting the / key 3 times in a row i.e. /// above the thing you are documenting. This information is passed through to intellisense and can be extracted automatically into external documentation.

* Organize namespaces into a clearly defined structure. All classes must be namespaced. Use lowercase for namespaces:  
    
  namespace com.super.module.submodule

{

* Use PascalCasing for class names. Use noun or noun phrases for class names:  
  
* Use PascalCasing for property names and public variables. Camel Casing for private variables.

public int CharacterHealth = 10;

private int characterArmorType = 2;

* Use PascalCasing for methods. Use camelCasing for method parameters. Always fill in a description and document the method, member names, and returned information:

// <summary>

// Increases health by increment provided

// </summary>

// <param name=”increment”>The amount to increment health by</param>

public void IncreaseHealth(int increment)

{

}

* Always put member variables up at the top of the class. Put static variables first.

public class CharacterController : MonoBehaviour

{

// <summary>

// Character Class

// </summary>

public static string CharacterClass;

// <summary>

// Internal data for character health

// </summary>

private int characterHealth = 100;

* Use camelCasing for variables used locally within a method:

public int GetUberHealth()

{

int uberHealth = 0;

if(\_characterHealth <= 0)

{

uberHealth = 0;  
}

else

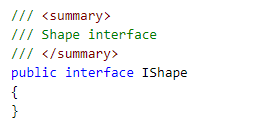
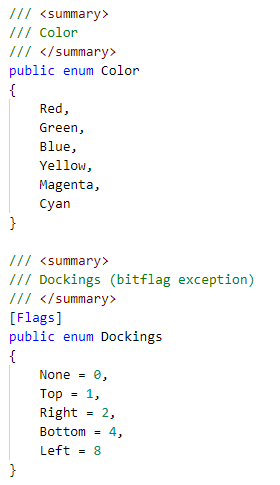
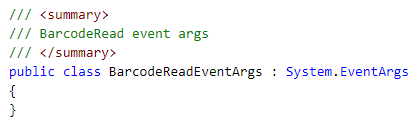
{

uberHealth = \_characterHealth \* 1000;

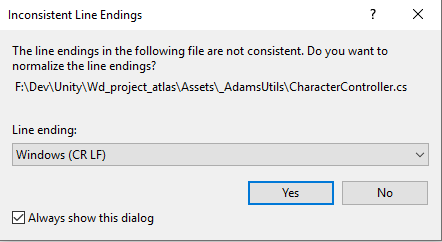
}

return uberHealth;

}

* Prefix interfaces with I, and name interfaces as nouns, noun phrases or adjectives.  
  
* Use singular names for enums, except for bit flags.  
  
* Use the EventArgs suffix for classes that contain information about an event:  
  

## Line Ending Conventions:

Use windows-based CR LF for line endings:  


## Layout Conventions:

Good layout uses formatting to emphasize the structure of your code and to make the code easier to read.

* Use the default Code Editor settings (smart indenting, four-character indents, tabs saved as spaces). For more information, see Options, Text Editor, C#, Formatting.
* Write only one statement per line.
* Write only one declaration per line.
* If continuation lines are not indented automatically, indent them one tab stop (four spaces).
* Add at least one blank line between method definitions and property definitions.
* Use parentheses to make clauses in an expression apparent, as shown in the following code:  
  
* Always vertically align opening and closing curly brackets on a separate line:

public bool IsCharacterDead()

{

if(characterHealth <= 0)

{

return true;  
}

else

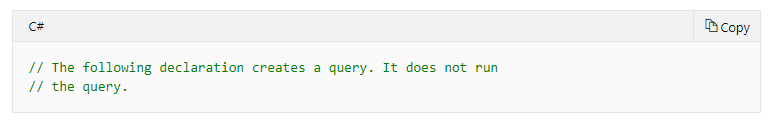
{

return false;

}

}

## Commenting Conventions:

* Place the comment on a separate line, not at the end of a line of code.
* Begin comment text with an uppercase letter.
* End comment text with a period.
* Insert one space between the comment delimiter (//) and the comment text, as shown in the following example.  
  
* Do not create formatted blocks of asterisks around comments.
* Comment code where the intention of the code could be misinterpreted in order to clarify the intended purpose of the code.