

✓ 100 XP

Exercise - Create effective code comments

13 minutes

In this exercise, you'll add notes to your code and temporarily disable certain lines of code from compilation. Then you'll look at how the C# compiler understands whitespace and how to use whitespace to increase the readability of your code.

What is a code comment?

A code comment is an instruction to the compiler to ignore everything after the code comment symbols in the current line.

c#

// This is a code comment!

This may not seem useful at first, however it's useful in three situations:

- When you want to leave a note about the intent of a passage of code. It can be helpful to include code comments that describe the purpose or the thought process when you're writing a particularly challenging set of coding instructions. Your future self will thank you.
- When you want to temporarily remove code from your application to try a different approach, but you're not yet convinced your new idea will work. You can comment out the code, write the new code, and once you're convinced the new code will work the way you want it to, you can safely delete the old (commented code).
- Adding a message like `TODO` to remind you to look at a given passage of code later. While you should use this judiciously, it's a useful approach. You may be working on another feature when you read a line of code that sparks a concern. Rather than ignoring the new concern, you can mark it for investigation later.

ⓘ Note

Code comments should be used to say what the code cannot. Often, developers update their code but forget to update the code comments. It's best to use comments for higher-level ideas and not to add comments about how an individual line of code works.

Prepare your coding environment

This module includes exercises that guide you through the process of building and running sample code. You are encouraged to complete these activities using Visual Studio Code as your development environment. Using Visual Studio Code for these activities will help you to become more comfortable writing and running code in a developer environment that's used by professionals worldwide.

1. Open Visual Studio Code.

You can use the Windows Start menu (or equivalent resource for another OS) to open Visual Studio Code.

2. On the Visual Studio Code **File** menu, select **Open Folder**.

3. In the **Open Folder** dialog, navigate to the Windows Desktop folder.

If you have a different folder location where you keep code projects, you can use that folder location instead. For this training, the important thing is to have a location that's easy to locate and remember.

4. In the **Open Folder** dialog, select **Select Folder**.


If you see a security dialog asking if you trust the authors, select **Yes**.

5. On the Visual Studio Code **Terminal** menu, select **New Terminal**.

Notice that a command prompt in the Terminal panel displays the folder path for the current folder. For example:

```
dos
```

```
C:\Users\someuser\Desktop>
```

 **Note**

If you are working on your own PC rather than in a sandbox or hosted environment and you have completed other Microsoft Learn modules in this C# series, you may have already created a project folder for code samples. If that's the case, you can skip over the next step, which is used to create a console app in the TestProject folder.

6. At the Terminal command prompt, to create a new console application in a specified folder, type **dotnet new console -o ./CsharpProjects/TestProject** and then press Enter.

This .NET CLI command uses a .NET program template to create a new C# console application project in the specified folder location. The command creates the CsharpProjects and TestProject folders for you, and uses TestProject as the name of your `.csproj` file.

7. In the EXPLORER panel, expand the **CsharpProjects** folder.

You should see the TestProject folder and two files, a C# program file named Program.cs and a C# project file named TestProject.csproj.

8. In the EXPLORER panel, to view your code file in the Editor panel, select **Program.cs**.
9. Delete the existing code lines.

You'll be using this C# console project to create, build, and run code samples during this module.

10. Close the Terminal panel.

Create and use code comments

In this task, you will try creating and removing various types of code comments.

1. In the Visual Studio Code Editor panel, enter the following code:

```
c#  
  
string firstName = "Bob";  
int widgetsSold = 7;  
Console.WriteLine($"{firstName} sold {widgetsSold} widgets.");
```

2. To modify your code with code comments and revisions, update your code as follows:

```
c#
```

```
string firstName = "Bob";  
int widgetsPurchased = 7;  
// Testing a change to the message.  
// int widgetsSold = 7;  
// Console.WriteLine($"{firstName} sold {widgetsSold} widgets.");  
Console.WriteLine($"{firstName} purchased {widgetsPurchased} widgets.");
```

3. Take a minute to review your comments and code updates.

Notice that the code comments are used to document the potential change being made, and to temporarily disable the old message as you test the new message. Your next step will be to test your update. If you're satisfied with the new code, you can safely delete the old code that was commented out. This is a safer, more methodical approach to modifying working code until you're convinced that you're ready to permanently remove it.

4. On the Visual Studio Code **File** menu, click **Save**.
5. In the EXPLORER panel, to open a Terminal at your TestProject folder location, right-click **TestProject**, and then select **Open in Integrated Terminal**.
6. At the Terminal command prompt, type **dotnet run** and then press Enter.

You should see the following output:

Output

Bob purchased 7 widgets.

Again, if you're satisfied with your updates, delete the old code that's commented out.

7. Delete the code comments.

Your code should match the following:

c#

```
string firstName = "Bob";  
int widgetsPurchased = 7;  
Console.WriteLine($"{firstName} purchased {widgetsPurchased} widgets.");
```

8. To apply a block comment that comments out multiple lines, update your code as follows:

c#

```
/*  
string firstName = "Bob";  
int widgetsPurchased = 7;  
Console.WriteLine($"{firstName} purchased {widgetsPurchased} widgets.");  
*/
```

Block comments are great if you need to write a long comment or remove many lines of code. Block comments use a `/*` at the beginning of the code and a `*/` at the end. Using a block comment is the quickest and easiest way to disable three or more lines of code.

9. Replace your existing code with the following:

```
c#  
  
Random random = new Random();  
string[] orderIDs = new string[5];  
// Loop through each blank orderID  
for (int i = 0; i < orderIDs.Length; i++)  
{  
    // Get a random value that equates to ASCII letters A through E  
    int prefixValue = random.Next(65, 70);  
    // Convert the random value into a char, then a string  
    string prefix = Convert.ToChar(prefixValue).ToString();  
    // Create a random number, pad with zeroes  
    string suffix = random.Next(1, 1000).ToString("000");  
    // Combine the prefix and suffix together, then assign to current OrderID  
    orderIDs[i] = prefix + suffix;  
}  
// Print out each orderID  
foreach (var orderID in orderIDs)  
{  
    Console.WriteLine(orderID);  
}
```

ⓘ Note

There are many C# concepts in this code listing that may be new to you. It's not necessary to understand what the code is doing in order to appreciate how comments can help readers understand the purpose of the code.

10. Take a minute to see if you can figure out the purpose of the code.

Given the comments, you might be able to figure out what the code is doing (assuming the comments accurately describe the current state and were updated as the code was updated). But can you guess why this code exists? Wouldn't it be helpful if there was some explanation at the top of the code file that provided some context and described its purpose?

11. Consider how you would improve the comments.

Notice that there are two main problems with these comments:

- The code comments unnecessarily explain the obvious functionality of individual lines of code. These are considered low-quality comments because they merely explain how C# or methods of the .NET Class Library work. If the reader is unfamiliar with these ideas, they can look them up using learn.microsoft.com or IntelliSense.
- The code comments don't provide any context to the problem being solved by the code. These are considered low-quality comments because the reader doesn't gain any insight into the purpose of this code, especially as it relates to the larger system.

12. Remove the existing comments.

Your code should match the following:

c#

```
Random random = new Random();
string[] orderIDs = new string[5];

for (int i = 0; i < orderIDs.Length; i++)
{
    int prefixValue = random.Next(65, 70);
    string prefix = Convert.ToChar(prefixValue).ToString();
    string suffix = random.Next(1, 1000).ToString("000");

    orderIDs[i] = prefix + suffix;
}

foreach (var orderID in orderIDs)
{
    Console.WriteLine(orderID);
}
```

Notice that the code is already less cluttered.

13. To add a comment that explains the higher-level purpose of your code, update your code as follows:

```
c#

/*
    The following code creates five random OrderIDs
    to test the fraud detection process. OrderIDs
    consist of a letter from A to E, and a three
    digit number. Ex. A123.
*/
Random random = new Random();
string[] orderIDs = new string[5];

for (int i = 0; i < orderIDs.Length; i++)
{
    int prefixValue = random.Next(65, 70);
    string prefix = Convert.ToChar(prefixValue).ToString();
    string suffix = random.Next(1, 1000).ToString("000");

    orderIDs[i] = prefix + suffix;
}

foreach (var orderID in orderIDs)
{
    Console.WriteLine(orderID);
}
```

A comment's usefulness is subjective. In all matters related to code readability, you should use your best judgment. Do what you think is best to improve the clarity of your code.

Recap

The main takeaways from this exercise:

- Use code comments to leave meaningful notes to yourself about the problem your code solves.
- Don't use code comments that explain how C# or the .NET Class Library works.
- Use code comments when temporarily trying alternative solutions until you're ready to commit to the new code solution, at which point you can delete the old code.
- Never trust comments. They may not reflect the current state of the code after many changes and updates.

Module complete:

Unlock achievement
