200 XP

Exercise - Code blocks and variable scope

11 minutes

A code block is one or more C# statements that define an execution path. The statements outside of a code block affect when, if, and how often that block of code is executed at run time. The boundaries of a code block are typically defined by squiggly braces, `{}`.

In addition to their effect on execution path, code blocks can also affect the scope of your variables. The code samples that you examine during this exercise will help you understand the relationship between code blocks and variable scope.

Code blocks impact the scope of a variable declaration

Variable *scope* refers to a variable's visibility to the other code in your application. A locally scoped variable is only accessible inside of the code block in which it's defined. If you attempt to access the variable outside of the code block, you'll get a compiler error.

The remainder of this unit explores the relationship between code blocks and variable scope.

Prepare your coding environment

This module includes hands-on activities that guide you through the process of building and running demonstration 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 a variable inside of a code block

You will begin by looking at the case when a variable is initialized inside a code block.

1. Ensure that you have Visual Studio Code open and Program.cs displayed in the Editor panel.

   > **Note**
   >
   > Program.cs should be empty. If it isn't, select and delete all code lines.

2. Type the following code into the Visual Studio Code Editor:

   ```c#
   bool flag = true;
   if (flag)
   {
       int value = 10;
       Console.WriteLine($"Inside of code block: {value}");
   }
   ```

3. On the Visual Studio Code **File** menu, select **Save**.

   The Program.cs file must be saved before building or running the code.

4. In the EXPLORER panel, to open a Terminal at your TestProject folder location, right-click **TestProject**, and then select **Open in Integrated Terminal**.

   A Terminal panel will open. The Terminal should include a command prompt showing that the Terminal is open to your TestProject folder location.

5. At the Terminal command prompt, to run your code, type **dotnet run** and then press Enter.

   > **Note**
   >
   > If you see a message saying "Couldn't find a project to run", ensure that the Terminal command prompt displays the expected TestProject folder location. For example:

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

You should see the following output:

Output

```
Inside of the code block: 10
```

This is the expected output. But what if you want to access the variable `value` outside of the `if` statement's code block?

Attempt to access the variable outside of the code block in which it was defined

1. In the Visual Studio Code Editor, create a new code line below the `if` statement's code block.
2. Add the following line of code:

```c#
Console.WriteLine($"Outside of code block: {value}");
```

3. Verify that your code looks like the following:

```c#
bool flag = true;
if (flag)
{
    int value = 10;
    Console.WriteLine("Inside of code block: " + value);
}
Console.WriteLine($"Outside of code block: {value}");
```

4. Save your code file, and then use Visual Studio Code to run your code.

   Enter `dotnet run` from the Terminal command prompt to run your code.

5. Notice that when you attempted to run the application, you got a compilation error:

Output

```
error CS0103: The name 'value' does not exist in the current context
```

The problem is that a variable defined in a code block is only accessible (or rather, visible) within that code block. The variable isn't accessible outside of the code block in which it's defined.

A *local variable* is defined as a variable that declared in a method code block.

Move the variable outside

To access a variable from both an outer and inner code block (like the `if` statement's code block), you'll need to move the variable declaration outside of the `if` statement's code block so that all the code has visibility to that variable.

1. Update your code in the Visual Studio Code Editor as follows:

```c#
bool flag = true;
int value;

if (flag)
{
    value = 10;
    Console.WriteLine("Inside of code block: " + value);
}
Console.WriteLine($"Outside of code block: {value}");
```

2. Take a minute to review the updates.

3. Notice that `value` is now declared (but not instantiated) outside the `if` code block.

4. Use Visual Studio Code to save and run your code.

5. Notice that you still get a compilation error.

   This time, when you attempt to run the application, you get the following compilation error:

Output

```
error CS0165: Use of unassigned local variable 'value'
```

This is a simple problem to fix, however, it gives you another insight into working with code blocks.

If the line of code `value = 10;` was outside (above) the `if` statement's code block, the compiler would compile your application, and everything would be fine. However, since that line of code is inside the `if` statement's code block, there's a possibility that the variable will never be assigned a value. In that case, the code statement below the code block that tries to display `value` would encounter an uninitialized variable, which the compiler won't allow.

Initialize the variable with a value

1. To include an initialization as part of the variable declaration, update your code as follows:

```
c#

bool flag = true;
int value = 0;

if (flag)
{
    value = 10;
    Console.WriteLine("Inside of code block: " + value);
}
Console.WriteLine("Outside of code block: " + value);
```

   To fix the "unassigned local variable" issue, you needed to initialize the variable with a value. Your update does this by initializing `value` as part of your variable declaration.

2. Use Visual Studio Code to save and run your code.

3. Notice that now, when you run the application, you see the following output.

```
Output

Inside of code block: 10
Outside of code block: 10
```

Recap

Here are a few important things to remember about code blocks:

- When you define a variable inside of a code block, its visibility is local to that code block and inaccessible outside of that code block.
- To make a variable visible inside and outside of a code block, you must define the variable outside of the code block.
- Don't forget to initialize any variables whose value is set in a code block, such as an `if` statement.

Check your knowledge

1. A developer writes some code that includes an `if` statement code block. They initialize one integer variable to a value of 5 above (outside) of the code block. They initialize a second integer variable to a value of 6 on the first line inside of the code block. The Boolean expression for the code block evaluates to `true` if the first integer variable has a value greater than 0. On the second line inside the code block, they assign the sum of the two values to the first variable. On the first line after the code block, they write code to display the value of the first integer. What is the result when the code statement used to display the first integer is executed? *

- ◯ No error is generated and the integer value is displayed. The value displayed is the sum of the first and second integer.
- ◯ No error is generated and the integer value is displayed. The value displayed is the initialized value from above the code block.
- ◯ An error is generated because the first variable isn't in-scope after the code block.

Check your answers