Programming Assignment 4.1

**Assignment Description**

Learning about abstraction and developing an **Apple** class (in the exercises) really made you hungry -- but for eggs, not apples.

In this assignment, you'll implement an **Egg** console app class.

**Why do we care?**

Because you should get more practice developing constructors, properties, and methods in your own console app class.

**Getting Started**

Download the appropriate zip file for your OS below and unzip the file somewhere on your computer.

**Windows Programming Assignment 4.1 Materials** ZIP File

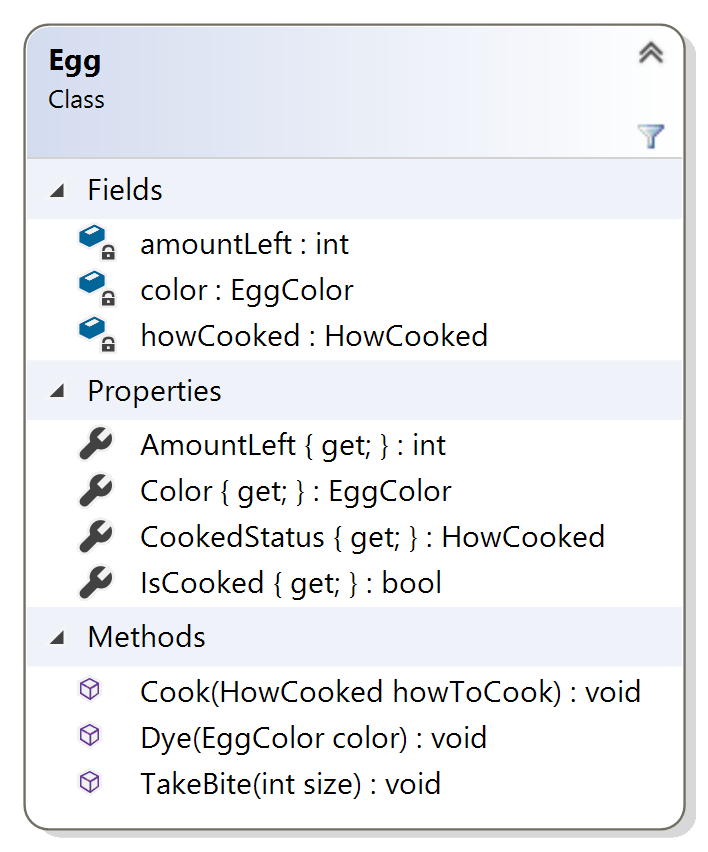
**Mac Programming Assignment 4.1 Materials** ZIP File

Open the project in Visual Studio.

**Important:** You MUST only add code as indicated by the comments in that file. If you don't, you're virtually guaranteed to fail all the test cases in the automated grader.

**Requirements**

For this assignment, you need to implement an **Egg** class as shown in the class diagram below:



The Visual Studio project I provided to you contains the **Egg** class (with the constructor, properties, and method bodies unimplemented) and a **Program** class that tests the **Egg** class. You shouldn't change the Program.cs file at all (if you do, you'll break the automated grader), all your work in this assignment is in the Egg.cs file.

I've also included two enumerations in the project: one for the color of the egg and one for how the egg is cooked. You shouldn't change those files either.

The constructor (not shown in the class diagram, but included in the **Egg** class I provided) should initialize the **amountLeft** and **color** fields based on the constructor parameters. I've already initialized the **howCooked** field in the code I provided.

**Hint:** You'll have to use **this** to set the color properly because the field and the parameter are both called **color**. Don't change the field or parameter name, learn how to use **this**!

The **AmountLeft** property should return the value of the **amountLeft** field.

The **Color** property should return the value of the **color** field.

The **CookedStatus** property should return the value of the **howCooked** field.

The **IsCooked** property should return true if the egg is cooked and false if the egg isn't cooked. You should be able to figure out how to use the value of the **howCooked** field to return the appropriate **bool** from this property..

The **Cook** method changes the **howCooked** field using the **howToCook** parameter. Once an egg has been cooked, it can't be cooked again.

The **Dye** method changes the **color** field using the **color** parameter. Only white eggs can be dyed, and they can only be dyed blue.

**Hint:** you'll have to use **this** to set the color properly because the field and the parameter are both called **color**. Don't change the field or parameter name, learn how to use **this**!

The **TakeBite** method changes the **amountLeft** field based on the **size** parameter. The value of the **amountLeft** field should never go below 0.

The best approach is to implement the constructor and the **AmountLeft** and **Color** properties first (to pass test case 1), then implement the **Cook** method and the **CookedStatus** property (to pass test case 2), and so on.

**Required Output Format**

The Program.cs file I provided handles all the output correctly. It simply prints Passed or FAILED for each test case that's selected by the user input.

**Running Your Code**

Because of the code I included to work with the automated grader on Coursera, when you run your program the command prompt window will open and it will sit there doing nothing. To make your code run, type in a test case number (1 through 10, inclusive) and press the <Enter> key; your code should then run so you can check your output.

For example, your input could be

**3**

to run the third test case.

You can actually run your code again if you want to by typing in a test case number and pressing the <Enter> key again. When you’re ready to stop running your code, type q (for quit).

Here's what running the code multiple times with different inputs should look like (though none of your test cases should fail!). The first line is the test case number, the second line is your output line, and so on:

**1**

**Passed**

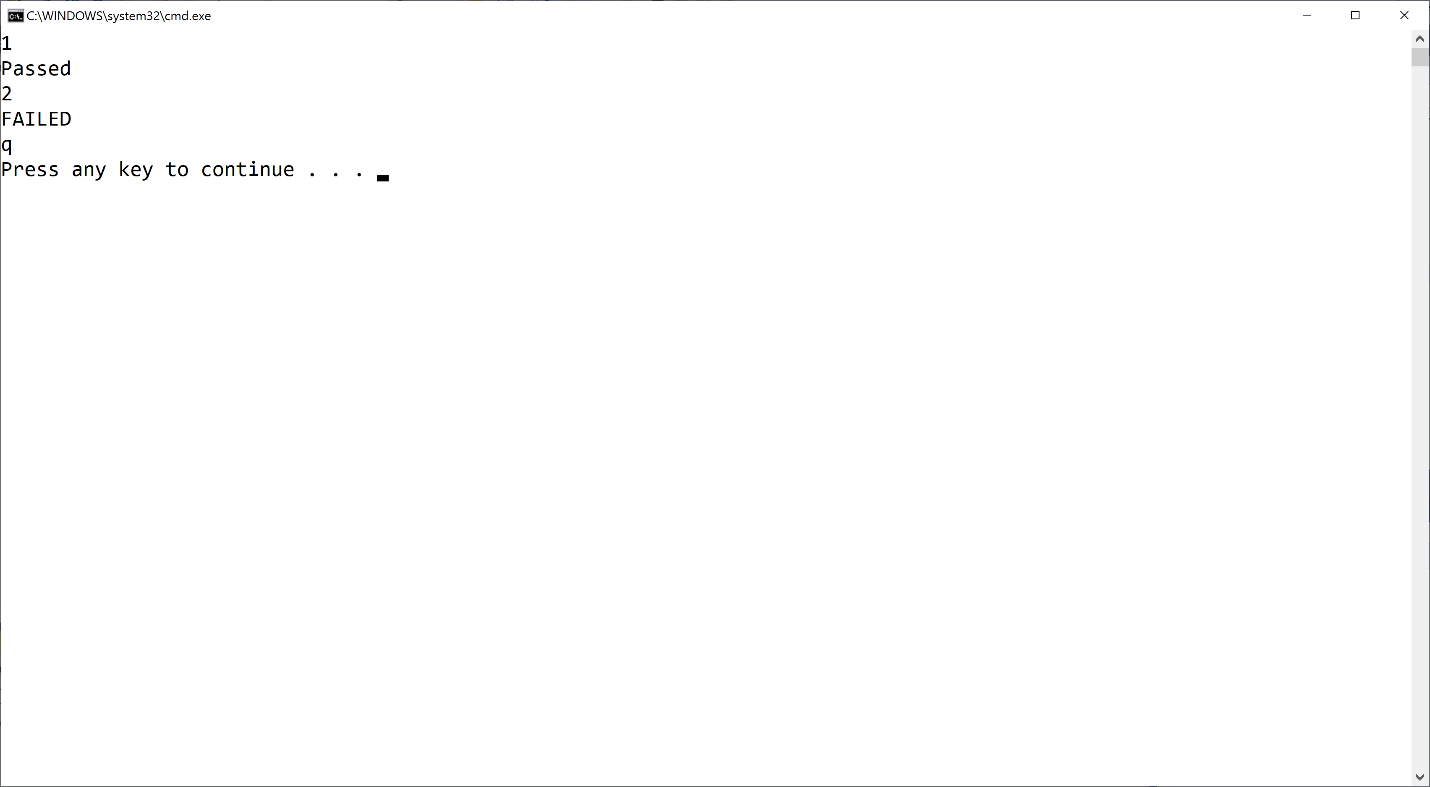
**2**

**FAILED**

**q**

**Important Note:** The Coursera formatting makes it look like there's a blank line between each of the lines above, but there's not. The above output should be exactly 5 lines of input/output.

The image below shows my console window when I run the code multiple times as described above:



If your output doesn't match the image above EXACTLY (no extra words, characters, spaces, or blank lines) you'll fail all the test cases in the automated grader. Of course, you shouldn't get any FAILED output when you run your code!

Mac users: You may or may not have an extra blank line after the q. That's fine.

**Test Case Inputs**

The automated grader uses the set of test cases that are built into the Program.cs file I provided to you.

**Common Problems**

Historically, lots of people post about grading errors on this assignment (because their code is wrong). Here are a couple things you should check:

1. You have to start with the Visual Studio project I gave you in the assignment materials and add your code to the appropriate places in the Egg class. The code I gave you includes the appropriate structure for the automated grader to work. If you don't do this, you'll almost definitely fail all the test cases in the automated grader
2. I give you an example in this assignment (and all assignments) for what output you should get when you run the code multiple times. Be sure to try that example; if your code doesn't generate output exactly as shown in the example, your code isn't working properly

**Submitting Your Solution**

Go to the LMS and submit your project to **Assignment 4.1**