C# Intermediate: Classes, Interfaces and OOP

By: Mosh Hamedani

www.programmingwithmosh.com

CLASSES - Exercises

Read this first

I've designed these exercise to help you apply what you have learned in this section. Make sure to do these exercises before moving on to the next chapter if you want to get the most out of this course.

Where are the answers?
I have deliberately decided not to provide the answers. Why? Because I want to make a great programmer out of you, not a lazy programmer who is used to copying/pasting code from various sources, without knowing what is happening!

In the real world, when you are at work, your job is to solve a problem. No one (including myself) knows the answers to real world problems initially. So we need to research, think, try different ideas, and see what works best. And that's exactly the kind of attitude I would like to see in you. If I give you the answers, you're going to get lazy and quickly look at the solution. This way, your programming brain will not be trained and you will always be dependent on other people's answers in the real-world.

So, do your best to solve these problems. They are not excessively hard, and any student who has taken all the lectures in this section should be able to solve these problems with a little effort. If you get stuck along the way, post your question in the discussion area. I'm happy to help you out. But please check the discussion area first to make sure no one has asked the same question before. This way, you'll save both your and my time.

So, let's get started!

Exercise 1: Design a Stopwatch

Design a class called Stopwatch. The job of this class is to simulate a stopwatch. It should provide two methods: Start and Stop. We call the start method first, and the stop method next. Then we ask the stopwatch about the duration between start and stop. Duration should be a value in TimeSpan. Display the duration on the console.

We should also be able to use a stopwatch multiple times. So we may start and stop it and then start and stop it again. Make sure the duration value each time is calculated properly.

We should not be able to start a stopwatch twice in a row (because that may overwrite the initial start time). So the class should throw an InvalidOperationException if its started twice.

Educational tip: The aim of this exercise is to make you understand that a class should be always in a valid state. We use encapsulation and information hiding to achieve that. The class should not reveal its implementation detail. It only reveals a little bit, like a blackbox. From the outside, you should not be able to misuse a class because you shouldn't be able to see the implementation detail.

‘

Exercise 2: Design a StackOverflow Post

Design a class called Post. This class models a StackOverflow post. It should have properties for title, description and the date/time it was created. We should be able to up-vote or down-vote a post. We should also be able to see the current vote value. In the main method, create a post, up-vote and down-vote it a few times and then display the the current vote value.

In this exercise, you will learn that a StackOverflow post should provide methods for up-voting and down-voting. You should not give the ability to set the Vote property from the outside, because otherwise, you may accidentally change the votes of a class to 0 or to a random number. And this is how we create bugs in our programs. The class should always protect its state and hide its implementation detail.

Educational tip: The aim of this exercise is to help you understand that classes should encapsulate data AND behaviour around that data. Many developers (even those with years of experience) tend to create classes that are purely data containers, and other classes that are purely behaviour (methods) providers. This is not object-oriented programming. This is procedural programming. Such programs are very fragile. Making a change breaks many parts of the code.