## **Introduction**

### **Basics**

C# is a statically-typed language, which means that everything has a type at compile-time. Assigning a value to a name is referred to as defining a variable. A variable can be defined either by explicitly specifying its type, or by letting the C# compiler infer its type based on the assigned value (known as *type inference*). Therefore, the following two variable definitions are equivalent:

int explicitVar = 10; *// Explicitly typed*

var implicitVar = 10; *// Implicitly typed*

Updating a variable's value is done through the = operator. Once defined, a variable's

type can never change.

var count = 1; *// Assign initial value*

count = 2; *// Update to new value*

*// Compiler error when assigning different type// count = false;*

C# is an **[object-oriented language](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/object-oriented-programming" \t "https://exercism.org/tracks/csharp/exercises/_blank)** and requires all functions to be defined in a *class*. The class keyword is used to define a class. Objects (or *instances*) are created by using the new keyword.

class Calculator

{

*// ...*

}

var calculator = new Calculator();

A function within a class is referred to as a *method*. Each method can have zero or more parameters. All parameters must be explicitly typed, there is no type inference for parameters. Similarly, the return type must also be made explicit. Values are returned from methods using the return keyword. To allow a method to be called by code in other files, the public access modifier must be added.

class Calculator

{

public int Add(int x, int y)

{

return x + y;

}

}

Methods are invoked using dot (.) syntax on an instance, specifying the method name to call and passing arguments for each of the method's parameters. Arguments can optionally specify the corresponding parameter's name.

var calculator = new Calculator();var sum\_v1 = calculator.Add(1, 2);var sum\_v2 = calculator.Add(x: 1, y: 2);

Scope in C# is defined between the { and } characters.

C# supports two types of comments. Single line comments are preceded by // and multiline comments are inserted between /\* and \*/.

## **Instructions**

Lucian's girlfriend is on her way home and he hasn't cooked their anniversary dinner!

In this exercise, you're going to write some code to help Lucian cook an exquisite lasagna from his favorite cook book.

You have four tasks, all related to the time spent cooking the lasagna.

### **1. Define the expected oven time in minutes**

Define the Lasagna.ExpectedMinutesInOven() method that does not take any parameters and returns how many minutes the lasagna should be in the oven. According to the cooking book, the expected oven time in minutes is 40:

var lasagna = new Lasagna();

lasagna.ExpectedMinutesInOven();*// => 40*

### **2. Calculate the remaining oven time in minutes**

Define the Lasagna.RemainingMinutesInOven() method that takes the actual minutes the lasagna has been in the oven as a parameter and returns how many minutes the lasagna still has to remain in the oven, based on the expected oven time in minutes from the previous task.

var lasagna = new Lasagna();

lasagna.RemainingMinutesInOven(30);*// => 10*

### **3. Calculate the preparation time in minutes**

Define the Lasagna.PreparationTimeInMinutes() method that takes the number of layers you added to the lasagna as a parameter and returns how many minutes you spent preparing the lasagna, assuming each layer takes you 2 minutes to prepare.

var lasagna = new Lasagna();

lasagna.PreparationTimeInMinutes(2);*// => 4*

### **4. Calculate the elapsed time in minutes**

Define the Lasagna.ElapsedTimeInMinutes() method that takes two parameters: the first parameter is the number of layers you added to the lasagna, and the second parameter is the number of minutes the lasagna has been in the oven. The function should return how many minutes you've worked on cooking the lasagna, which is the sum of the preparation time in minutes, and the time in minutes the lasagna has spent in the oven at the moment.

var lasagna = new Lasagna();

lasagna.ElapsedTimeInMinutes(3, 20);*// => 26*