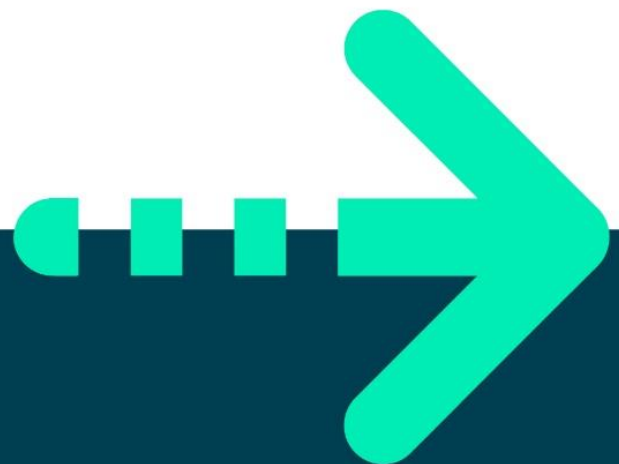




LAB 3, C# – INTRODUCTION TO METHODS

FUNDAMENTALS





Lab 3, C# – Introduction to methods

Objective

The objectives of this practical session are as follows.

- To be able to write and invoke methods with varying number of parameters, some of which return a value.
- To accept user input in response to a prompt and process that data further including converting it to a different type of data.
- You'll also create and use a new class

Part 1 – Authoring a helper method

Step by step.

1. Launch Visual Studio and create a new **Desktop->Console App** project. Please refer to lab1's instructions if you need help.

2. Name this project as **Lab03**.

3. Add a new method in the **Program** class as `public static int GetInt(string prompt)`

This method has a `string` parameter called *prompt*, which it displays before getting an integer input from the user. It then returns an `int`.

Tip: to get keyboard input use the `Console.ReadLine()` method

4. Create another method as `public static string GetString(string prompt)`.
5. This method is similar to the `GetInt()` method except it returns a string
6. Call both methods in the **Main()** method and then print the result to test your code. For example, try getting someone's name and age;



Part 2 – Performing data conversions

The scenario is going to mimic a serving line at a lunch hall in that we are going to prompt the user to answer certain questions. What would you like as a main dish? Then how many Roast Potatoes? How many Brussels Sprouts? Then display what their lunch is.

Step by step.

1. Create a method called **TheLunchQueue**. In the *Program* class.
Tip: `public static void TheLunchQueue()`
2. Call the `GetString()` method to display the following
What main dish would you like (Fish, Burgers or veg) ?
And get the answer into a variable called `mainCourse`.
3. Use the `GetInt()` method to display the following prompts and capture the values in suitable variable names.

How many roast potatoes would you like?
How many Brussel sprouts would you like?

Display the description for producing a bill. Something like:
Hello, your lunch is xx with yy roast potatoes and zz Brussel sprouts.

Replacing xx, yy and zz with your actual values of course!
4. Test your code by calling **TheLunchQueue() method** from `Main()`.

Part 3 - Weight Conversions

1. Create a method as
`public static void ConvertInputToStonesPounds(int pounds)`.
2. Ask the user for a total weight in pounds in `Main()` and pass the result to the above new method.
3. Display the result (stones & pounds) in the new method.

Note: there are 14 pounds in a stone.
Tip: Use division (/) and modulus (%)
4. Create another method as
`public static void ConvertKgsToStonesPounds(int kg)`.
 - a. Ask the user for a weight in kilograms.
 - b. Convert the weight and display it in stones and pounds
Hint: 1 kilo = 2.20462 pounds
Tip: convert the Kg to pounds and then call
`ConvertInputToStonesPounds(int kg)`
5. Test your code at each stage



Part 4 – Move your code to a separate class

Does every method have to be in the Program class?

In this part you'll create a new class and move all the code to that class.

6. Create a new Class called **Lab3Exercises** without a `main()` method in this project.

Tip: Right mouse click on the project name (not the Solution) and then select the Add->Class... menus

7. Cut all the code outside of the Program's `Main()` method and paste them inside the *Lab3Exercise* class.

8. Remove the **static** word from every method in *Lab3Exercise* class.

We'll discuss static method at a later date. The only reason why every method was static was because `Main()` is a **static** method, but we are now free of `Main()`!

9. Back in the `Main()` method, create an instance of *Lab3Exercises* class and use it to call the methods.

```
Lab3Exercises myLab3 = new Lab3Exercises();
```

10. At the start of each method call (in main) add "**myLab3**"

For example:

instead of **TheLunchQueue()** type **myLab3.TheLunchQueue()**

11. Run the application to make sure everything works.

**** End ****

