Methods

**Defining Methods**

A method is a code block that contains a series of statements. A program causes the statements to be executed by calling the method and specifying any required method arguments. In C#, every executed instruction is performed in the context of a method. The Main method is the entry point for every C# application and it is called by the common language runtime (CLR) when the program is started.

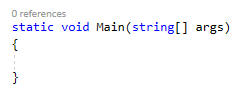
To use a method, you need to:

* Define the method
* Call the method

When you define a method, you basically declare the elements of its structure. The syntax for defining a method in C# is as follows −

**<Access Modifier>** **<Return Type>** **<Method Name>** **(Parameters)**  
   {  
        **method body**  
   }

An example would be**:**

   
These parts are defined as:

* **Access Modifier** − This determines the visibility of a variable or a method from another class.
* **Return type** − A method may return a value. The return type is the data type of the value the method returns. If the method is not returning any values, then the return type is void.
* **Method name**− Method name is a unique identifier and it is case sensitive. It cannot be same as any other identifier declared in the class.
* **Parameter list**− Enclosed between parentheses, the parameters are used to pass and receive data from a method. The parameter list refers to the type, order, and number of the parameters of a method. Parameters are optional; that is, a method may contain no parameters.
* **Method body** − This contains the set of instructions needed to complete the required activity.

These method blocks allow us to group our code together so that a single activity can be found in a single place. There are a few conventions though that we need to follow to make our code easier for others to understand and meet industry standards.

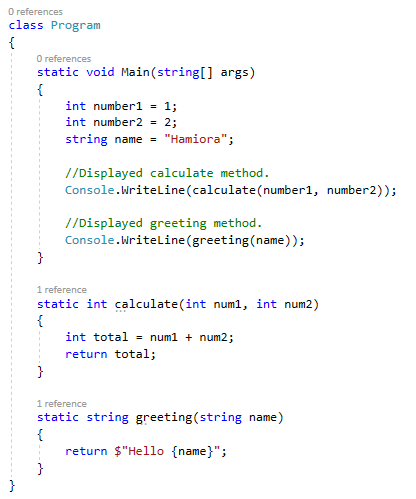
* A method should only perform a single function. If you method does more than one thing then maybe it should more than one method.
* The name of a function should explain what the function does.
* The function should return the value rather than print it to the screen. This is in case you wish to use your method elsewhere or in a different platform

**Returning a value**

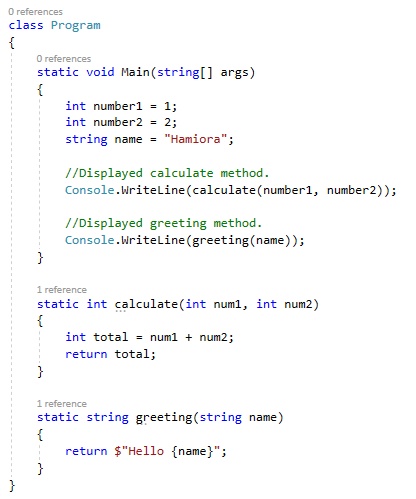
If we look at what we've done so far, our main method has had a return type of VOID. this means that this method doesn't return anything when it called. If we want our methods to return a value we need the return type set to what ever type of data we want to return is. As an example if we wanted our method to return a int number the return type would need to be an int.

e.g:

In this example the method will only be able to return an int as that's what has been specified by the return type in the method declaration. This returned variable can be used as we need as it can be displayed or even used in further methods. An example of this would be:



In this example the method is called twice and both times it displays the result that is being returned from the calculate method.



**Parameters**

Using parameters we are able to use the same method, but allow for different results with the output depending on the parameters its been passed. This would be helpful if you want to print a greeting with a custom name, or calculate a number but you do not know what the input values are. Examples of each are:

When calling methods from anywhere you need to ensure that the access modifier allows you to access that method where ever you are (SCOPE) and also that you pass the correct variables according to the method declaration. If you don't do this correctly you will receive an error.

**Exercises**

**1.**Write a method that changes the sign of a number passed to it. The method needs to returns a number with the opposite sign.

**2**. Write a method that accepts a number representing a distance in miles from the user. The function needs to return a number representing the distance in kilometers. You then need to display the original value and the result to the user.

**Conversion:** **1 kilometer is 0.62137 miles**

**3.**Write two methods. Each method accepts a parameter representing a circle’s radius from the user.

− One method needs to return a number representing the circumference of the circle.

− The other method returns a number representing the area of the circle.

**Formula: C = 2 π r        A = π r2       Use an approximation for pi: π = 3.1416**

Display both of these results to the user.

**4.**Write a method that accepts a value, ie. a parameter is passed to the method that represents a day number. The method returns the day name. You need to use an array of names which is used to hold all of the week days. **Don't use if or switch statements.**