Assignment 2: Iteration and Selection Algorithms

Loops, if-else, switch

1. Objectives

The main objectives of this assignment are:

- To exercise with conditional and iteration algorithms
- To learn how to use string.Format
- To work with methods having a parameter and a return type.

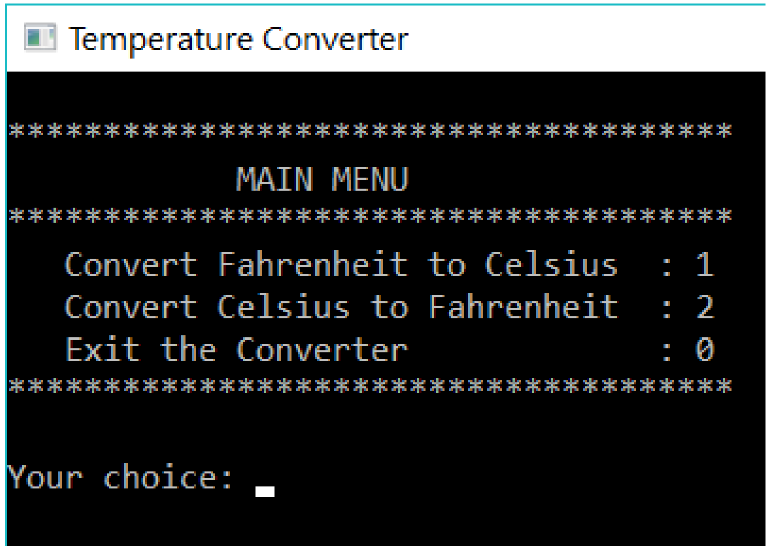
We are going to write many methods in this assignment. The rule to remember is to write a method for every task you would like the objects of a class to be able to do. Methods can be void or returning a value. Void methods perform tasks when they are called but it does not send any value back to the caller. It can have parameters if it needs information that is not available or accessible to the method. Returning value methods work exactly the same way with one exception and that is, it must always send a value back to the caller through a return statement.

Note: This assignment is graded as Pass (G) or Resubmit. If you aim to qualify for a VG, you need to do the VG-Assignment instead.

2. Description

In this assignment, we are going to write a program that displays a list of temperatures in Celsius converted to Fahrenheit and vice versa. A menu is provided for the user to choose the type of conversions: The menu should repeat until the user chooses 0 to exit.

When option 1 is chosen, the program calculates and displays a list of values between 0 and **212** degrees in Fahrenheit converted to Celsius degrees as shown in the next image When option 2 is selected, the program lists values


Temperature Converter

```
**************************************
            MAIN MENU
**************************************
   Convert Fahrenheit to Celsius   : 1
   Convert Celsius to Fahrenheit   : 2
   Exit the Converter              : 0
**************************************


Your choice: _
```

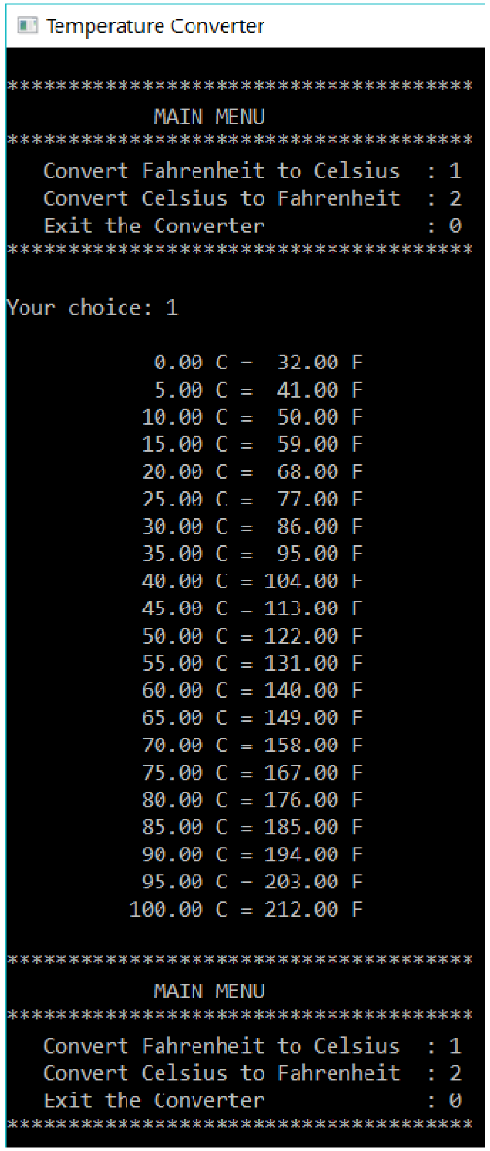
from 0 to **100** Celsius converted to Fahrenheit degrees. In the demo below, the results are calculated with intervals of 5 for option 1 and 4 for option 2 to make the list shorter on the Console window. You can try any interval 1, 4 or 5. Use the conversion formulas:

```
F = 9/5 * C + 32
C = 5/9 *(F - 32)
```

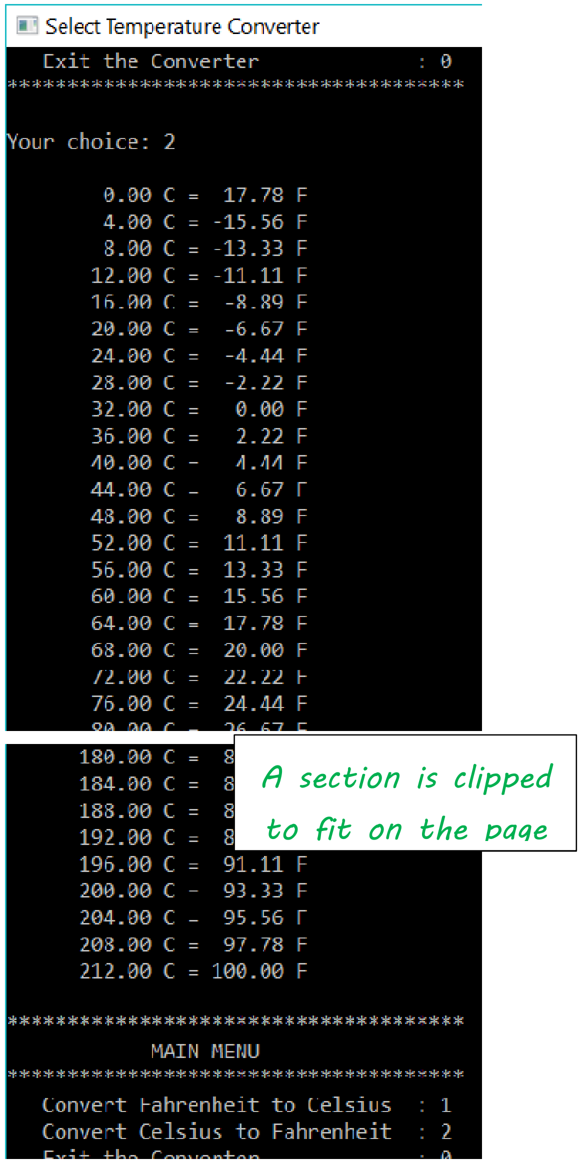
where F = Fahrenheit and C = Celsius.

Note: The images are screen shots of a running session, and placed side by side in above to save space. When running the program, they are displayed one a time on the Console window.

As can be noticed from the above sample, the Main Menu repeats itself after each operation and stops when the user inputs 0, whereby 'When a 0 value is given, the Main Menu loop terminates and so does the application. At this point, the job of the **TemperatureConverter** object is finished and the execution goes back to where the object was initiated and called (Main method)


Temperature Converter

```
****************************************
            MAIN MENU
****************************************
   Convert Fahrenheit to Celsius   : 1
   Convert Celsius to Fahrenheit   : 2
   Exit the Converter              : 0
****************************************

Your choice: 1

            0.00 C =  32.00 F
            5.00 C =  41.00 F
           10.00 C =  50.00 F
           15.00 C =  59.00 F
           20.00 C =  68.00 F
           25.00 C =  77.00 F
           30.00 C =  86.00 F
           35.00 C =  95.00 F
           40.00 C = 104.00 F
           45.00 C = 113.00 F
           50.00 C = 122.00 F
           55.00 C = 131.00 F
           60.00 C = 140.00 F
           65.00 C = 149.00 F
           70.00 C = 158.00 F
           75.00 C = 167.00 F
           80.00 C = 176.00 F
           85.00 C = 185.00 F
           90.00 C = 194.00 F
           95.00 C = 203.00 F
          100.00 C = 212.00 F

****************************************
            MAIN MENU
****************************************
   Convert Fahrenheit to Celsius   : 1
   Convert Celsius to Fahrenheit   : 2
   Exit the Converter              : 0
****************************************
```

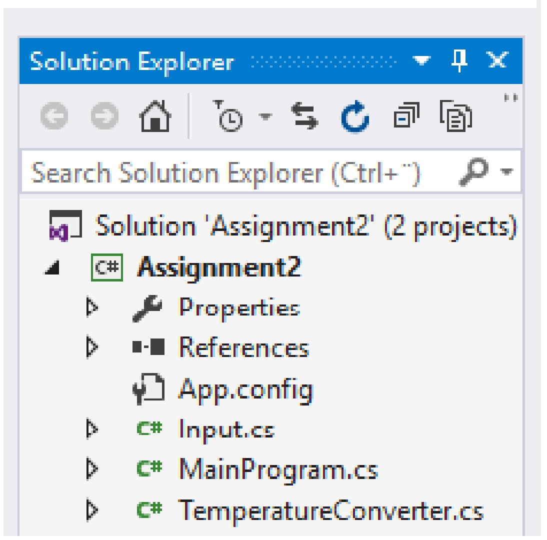

Select Temperature Converter

```
   Exit the Converter              : 0
****************************************

Your choice: 2

        0.00 C =  17.78 F
        4.00 C = -15.56 F
        8.00 C = -13.33 F
       12.00 C = -11.11 F
       16.00 C =  -8.89 F
       20.00 C =  -6.67 F
       24.00 C =  -4.44 F
       28.00 C =  -2.22 F
       32.00 C =   0.00 F
       36.00 C =   2.22 F
       40.00 C =   4.44 F
       44.00 C =   6.67 F
       48.00 C =   8.89 F
       52.00 C =  11.11 F
       56.00 C =  13.33 F
       60.00 C =  15.56 F
       64.00 C =  17.78 F
       68.00 C =  20.00 F
       72.00 C =  22.22 F
       76.00 C =  24.44 F
       80.00 C =  26.67 F
      180.00 C =  8
      184.00 C =  8
      188.00 C =  8
      192.00 C =  8
      196.00 C =  91.11 F
      200.00 C =  93.33 F
      204.00 C =  95.56 F
      208.00 C =  97.78 F
      212.00 C = 100.00 F

****************************************
            MAIN MENU
****************************************
   Convert Fahrenheit to Celsius   : 1
   Convert Celsius to Fahrenheit   : 2
   Exit the Converter              : 0
```

A section is clipped to fit on the page

3. The Project

Copy the **Input.cs** class from the code sample available in Module 2, and change its namespace to your application's namespace. You can then use its method(s) to read numeric values from the Console.

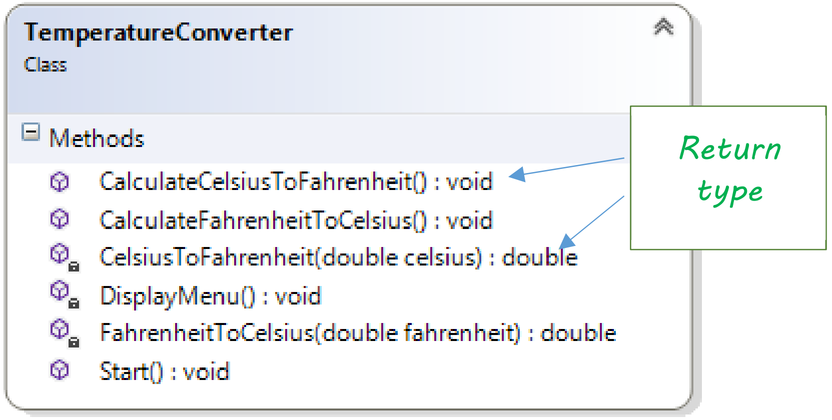


Solution Explorer

Search Solution Explorer (Ctrl+¨)

```
Solution 'Assignment2' (2 projects)
  Assignment2
    Properties
    References
    App.config
    Input.cs
    MainProgram.cs
    TemperatureConverter.cs
```

You can design the **TempeatureConverter** class by yourself but to give you an idea of the fields and methods that you may need to write, a list of the members of the class are show in the figure below. . Notice that no instance variable is required.



```
TemperatureConverter
Class

Methods
  CalculateCelsiusToFahrenheit() : void
  CalculateFahrenheitToCelsius() : void
  CelsiusToFahrenheit(double celsius) : double
  DisplayMenu() : void
  FahrenheitToCelsius(double fahrenheit) : double
  Start() : void
```

The **Start**() method in the **TemperatureConverter** class is called by the Main methods in the **MainProgram** class.

This method in turn calls the methods **DisplayMenu** (same class) and then depending the user's choice, calls either the method **CalculateCelsiusToFahrenheit**, or **CalculateFahrenheitToCelsius**.

The above two methods call **CelsiusToFahrenheit** and **FahrenheitToCelsius** to convert a value from Celsius to Fahrenheit and from Fahrenheit to Celsius, respectively.

4. Requirements

3.1 Requirements for a pass grade

3.1.1 The application must at least have two classes, one containing the **Main** method and one for the calculation of temperatures (**TemperatureConverter**).

3.1.2 The Main method should be kept short; it should only create an instance of the **TemperatureConverter** class and call a starting method (as in the previous assignment).

3.1.3 All class fields (if any) should be declared as private.

3.1.4 Control of the user input must be done so the values provided are as expected. For example, the menu choice must be an integer 0, 1, or 2, and nothing else.

3.1.5 Compile, run, and test your project before submission .The project must compile without error and the program should work satisfactorily.

If you need help, a step-by-step guidance and some examples are available in the Assignment page in the module.

3.2 Optional part - Displaying results in columns

This part is optional and can be skipped. The requirements listed below are in addition to the above.

3.2.1 Show the results distributed over a number of columns as the picture below that shows the results listed results in three columns.

3.2.2 Comment your methods using three slashes "///". Click on the line just above the method definition in your code file, and write three slashes. VS will then create a skeleton for the comments.

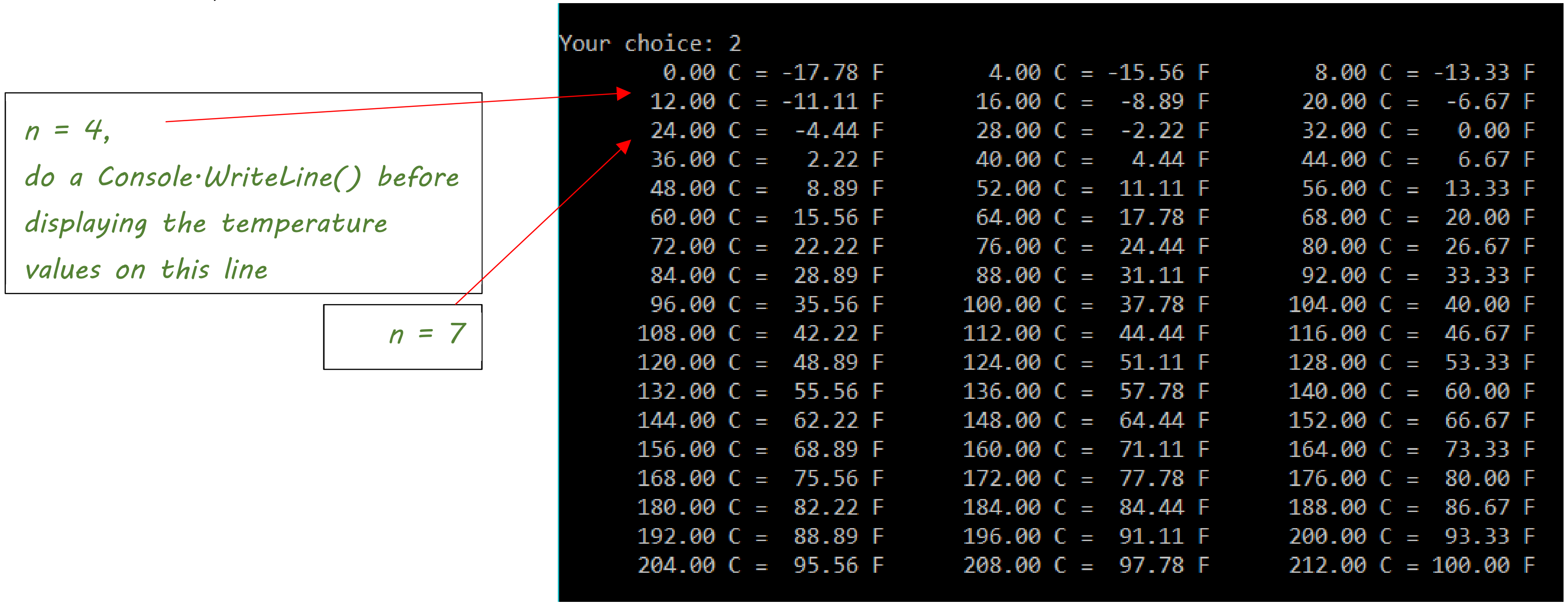
3.2.3 Use proper names for variables, methods, other identifiers as well as for your project and your application.

To divide the results into columns, you simple format a string with the number of values equal to number of columns. The values are formatted by using the string.Format with a certain width (to get spaces between the values. To determine when to cause a "carriage return, i.e. Console.WriteLine()), you can use the % operation:

```
if ( (counter % n ==  0) && counter > n)
```

where n = number of times. In the figure below n = 3;

The above if-statement with n = 3 gives a true value when counter is 3, 6, 9, 12, 15 and so on. Display the values using Console.Write and whenever the if-state results into a true value, call Console.WriteLIne(). The variable counter is a local variable that should be incremented by one every time a column is complete. In other words, you need one loop counter for iterations of all temperature values and one for the values of n (3, 6, 9, 12, 15, etc.)

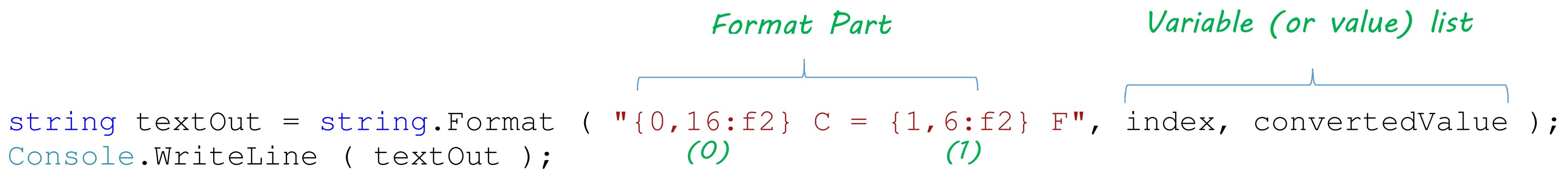

n = 4,
do a Console.WriteLine() before displaying the temperature values on this line
n = 7

```
Your choice: 2
     0.00 C = -17.78 F        4.00 C = -15.56 F        8.00 C = -13.33 F
    12.00 C = -11.11 F       16.00 C =  -8.89 F       20.00 C =  -6.67 F
    24.00 C =  -4.44 F       28.00 C =  -2.22 F       32.00 C =   0.00 F
    36.00 C =   2.22 F       40.00 C =   4.44 F       44.00 C =   6.67 F
    48.00 C =   8.89 F       52.00 C =  11.11 F       56.00 C =  13.33 F
    60.00 C =  15.56 F       64.00 C =  17.78 F       68.00 C =  20.00 F
    72.00 C =  22.22 F       76.00 C =  24.44 F       80.00 C =  26.67 F
    84.00 C =  28.89 F       88.00 C =  31.11 F       92.00 C =  33.33 F
    96.00 C =  35.56 F      100.00 C =  37.78 F      104.00 C =  40.00 F
   108.00 C =  42.22 F      112.00 C =  44.44 F      116.00 C =  46.67 F
   120.00 C =  48.89 F      124.00 C =  51.11 F      128.00 C =  53.33 F
   132.00 C =  55.56 F      136.00 C =  57.78 F      140.00 C =  60.00 F
   144.00 C =  62.22 F      148.00 C =  64.44 F      152.00 C =  66.67 F
   156.00 C =  68.89 F      160.00 C =  71.11 F      164.00 C =  73.33 F
   168.00 C =  75.56 F      172.00 C =  77.78 F      176.00 C =  80.00 F
   180.00 C =  82.22 F      184.00 C =  84.44 F      188.00 C =  86.67 F
   192.00 C =  88.89 F      196.00 C =  91.11 F      200.00 C =  93.33 F
   204.00 C =  95.56 F      208.00 C =  97.78 F      212.00 C = 100.00 F
```

5. Help and Guidance:

5.1. String.Format (or string.Format),

The object string has several useful methods that you use to manipulate text. Use String.Format (or string.Format), if you need to insert the value of an object, variable, or expression into another string. For example, you can insert the value of a double type into a string to display it to the user as a single string. You can combine several values to display in the same string and you can format each value using formatting rules that are available in C# for different value types.
The string.Format ( ) method takes an argument which is a combination of a format part and a list variables. These are passed to the method inside the parentheses.( ). The formatting part is to be enclosed inside quotation marks, followed by a comma and then a list of variables, which you refer to in the formatting part. Here is an example:



```
string textOut = string.Format ( "{0,16:f2} C = {1,6:f2} F", index, convertedValue );
Console.WriteLine ( textOut );
```

where *index* is an integer and *convertedValue* is a double. What happens in the above statement is that `"{0,16:f2}` will be replace by the current value of *index* and `{1,6:f2}` will be replaced by the value of the variable *convertedValue*. The index-variable is the first variable in the list and therefore a 0 is used at the beginning of the expression `"{0,16:f2}` The variable convertedValue is second variable in the list.

The expression "`{0,16:f2}` has these parts:

- 0 the position of the variable in the list to be used.
- Comma is a separator
- 16 is the number of characters within which the value of index is to be aligned.
- :f2 is the number of decimal positions to which the number is to be rounded off (rounding upwards). The letter “f” stands for floating-point and 2 sets the number of decimal positions.

If index has a value 2, it will be displayed as 2.00, right-adjusted within a 16-character width. Therefore, the method will be adding 12 spaces to the left of 2.00. The resulting substring will be displayed 12 spaces from the left.

In the same way, the expression `{1,6:f2}` instructs the computer to take the variable number 1 from the list, i.e. ***convertedValue***, round it up to two decimal places and display the result within a 6-character-width, right adjusted.
To left-adjust a substring, you need to add a minus sign before the width, {0, -16:f2}. There is no way for center adjusting.

All other characters and words that are not inside the curly braces are considered as constants and will be displayed as they are.

Important notes:

1. Console.WriteLine is designed to work in the same way as string.Format and you don't need to use string.Format. Instead of preparing a formatted string (textOUt in the code example above) and then using Console.WriteLine, you can write a formatted text to the Console directly:

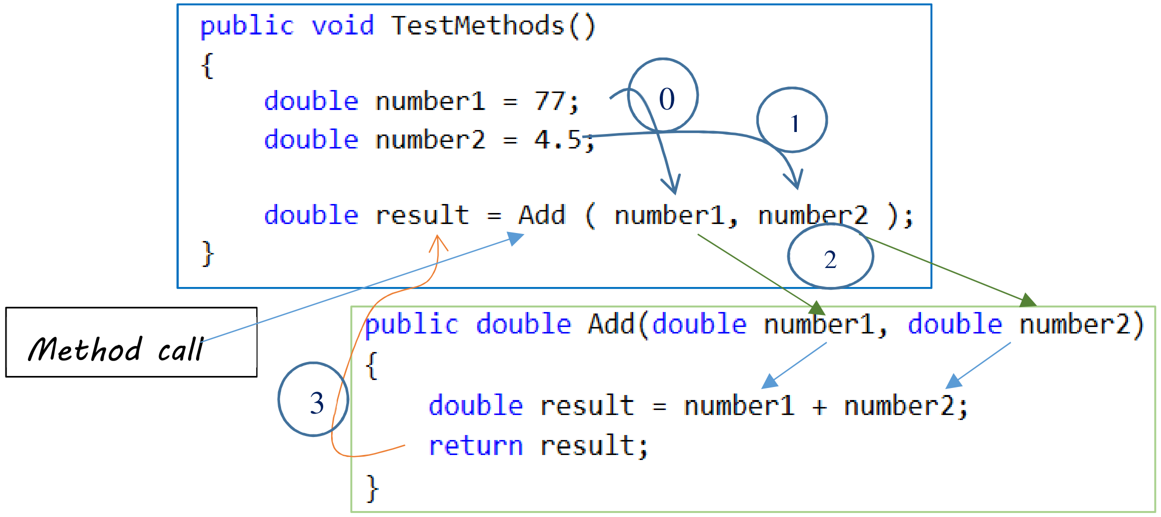
   ```
   Console.WriteLine ( "{0,16:f2} C = {1,6:f2} F", index, convertedValue );
   ```

   The good thing about string.Format is that you can use it at any time; you can use it with Console, with Windows Forms controls to display texts in components other than a Console, and whenever you would need to have a formatted string.

2. To get straight columns when you have more lines using same formatting, you should use a font like “Courier New” which uses the same width for all characters. Such a font draws a space, a small ‘i’ and a big ‘W’, with the same number of pixels on the Console (or other components).

5.2. Method with parameters and a return value

In code snippet that follows, you can see a method that has two parameters (arguments). The parameters are to receive input values from the caller method. The method then sends back a value, the result of the calculation, to the caller through the return statement.



```
public void TestMethods()
{
    double number1 = 77;
    double number2 = 4.5;

    double result = Add ( number1, number2 );
}
```

```
public double Add(double number1, double number2)
{
    double result = number1 + number2;
    return result;
}
```

Note that variables number1 and number2 in the method **Add** are not the same variables as in the caller method **TestMethods** although they have same names. They could have any other names.

More help? See the mode on the Its L for the help document and exercises.

6. Submission

Submit your assignment in the same way as the previous one.

Good Luck.
Farid Naisan