**Part 69 - Making method parameters optional by specifying parameter defaults**

In this video, we will discuss making **method parameters optional by specifying parameter defaults**.  
  
**This method allows us to add any number of integers**  
public static void AddNumbers(int firstNumber, int secondNumber,   
    int[] restOfTheNumbers)  
{  
    int result = firstNumber + secondNumber;  
    foreach (int i in restOfTheNumbers)  
    {  
        result += i;  
    }  
  
    Console.WriteLine("Total = " + result.ToString());  
}

**If we want to add 5 integers** - 10, 20, 30, 40 and 50. We call the method as shown below.  
AddNumbers(10, 20, new int[]{30, 40, 50});  
  
**At the moment all the 3 parameters are mandatory**. If I want to add just 2 numbers, then I can invoke the method as shown below. Notice that, I am passing an empty integer array as the argument for the 3rd parameter.  
AddNumbers(10, 20, new int[]{});  
  
**We can make the 3rd parameter optional by specifying a default value of null** for the 3rd parameter.   
public static void AddNumbers(int firstNumber, int secondNumber,  
    int[] restOfTheNumbers = null)  
{  
    int result = firstNumber + secondNumber;  
  
    // loop thru restOfTheNumbers only if it is not null  
    // otherwise you will get a null reference exception  
    if (restOfTheNumbers != null)  
    {  
        foreach (int i in restOfTheNumbers)  
        {  
            result += i;  
        }  
    }  
    Console.WriteLine("Total = " + result.ToString());  
}  
  
**Since we have specified a default value for the 3rd parameter**, it is optional. So, if we want to add just 2 numbers, we can use the function as shown below.  
AddNumbers(10, 20);  
  
**Optional parameters must appear after all required parameters**  
The following method will not comiple. This is because, we are making parameter "a" optional, but it appears before the required parameters "b" and "c".  
public static void Test(int a = 10, int b, int c)  
{  
    // Do something  
}  
  
**The following method will compile**, as optional parameter "a" is specified after all the required parameters ("b" & "c").  
public static void Test(int b, int c, int a = 10)  
{  
    // Do something  
}  
  
**Named Parameters**  
In the following method, parameters "b" & "c" are optional.  
public static void Test(int a, int b = 10, int c = 20)  
{  
    Console.WriteLine("a = " + a);  
    Console.WriteLine("b = " + b);  
    Console.WriteLine("c = " + c);  
}  
  
When we invoke this method as shown below, "1" is paased as the argument for parameter "a" and "2" is passed as the argument for parameter "b" by default.  
Test(1, 2);  
  
My intention is to pass "2" as the argument for parameter "c". To achieve this we can make use of named parameters, as shown below. Notice that, I have specified the name of the parameter for which value "2" is being passed.  
Test(1, c: 2);