**Topic 4: Overloading Methods**

Top of Form

Bottom of Form

**Content**

**Reading**

[Reading](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286950_1)

Methods

* + Microsoft:docs.microsoft.com
    - [Methods](https://docs.microsoft.com/en-us/dotnet/csharp/methods)
  + GeeksForGeeks: geeksforgeeks.com
    - [Methods](https://www.geeksforgeeks.org/c-sharp-methods/)
    - [Method Overloading](https://www.geeksforgeeks.org/c-sharp-method-overloading/)
  + TutorialsPoint: tutorialspoint.com
    - [Methods](https://www.tutorialspoint.com/csharp/csharp_methods.htm)
    - [Method Overloading](https://www.tutorialspoint.com/What-is-overloading-in-Chash)
    - [Method Overloading](https://tutorialspoint.dev/language/c-sharp/c-method-overloading) (on tutorialspoint.dev)
  + W3Schools: w3schools.com
    - [Methods](https://www.w3schools.com/cs/cs_methods.asp)
    - [Method Overloading](https://www.w3schools.com/cs/cs_method_overloading.asp)

**Overloading Methods**

[Overloading Methods](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286950_1)

Methods break down code into smaller, easier to maintain modules. A method with no return type executes statements and ends. A value-returning method executes statements and returns a value to the method call. A method call is the statement that invokes, or initiates the method code at that point in the program.

It is nice to have method called getUserInputInt() in your Main method for several reasons.

* + Readability-your Main method is smaller, easier to read (and thus maintain)
  + Robustness and Elegance-you can maintain error and exception handling in the method
  + Modularity-you can reuse the method getUserInputInt() more easily that copy paste in the same program and in other programs

You have already used methods. Here is an an example of a method call:

public static void Main(string[] args)

{

String fullName = "Ayah Morgana";

String middleName = fullName.Substring(5); // Call to method Substring(int) from Class String

Console.WriteLine(middleName);

}

You have already used other methods from the String class, here are the signatures of a few methods

* + String Substring(int index, int length) // This is an overloaded method (more on that later!)
  + bool Equals(String)
  + int CompareTo(String)

**Overloading Methods**

[Overloading Methods](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286950_1)

You have seen another example of method overloading in learning methods:

https://youtu.be/Cb2rBNywhr0

using System;

namespace MethodsOverload

{

class Program

{

private static void DisplayMessage()

{

Console.WriteLine("Print a message");

}

private static void DisplayMessage(String msg)

{

Console.WriteLine(msg);

}

private static void DisplayMessage(String name, int score)

{

Console.WriteLine("Nice job {0:G} on your score of {1:G}", name, score);

}

public static void Main(string[] args)

{

DisplayMessage();

DisplayMessage("Methods are fun!");

// Method call using literals

DisplayMessage("Mohammed", 90);

// Method call using variables

String name = "Ayah";

int score = 80;

DisplayMessage(name, score);

}

}

}

All the methods have the same name, but different argument lists. They have different method signatures.

[**MethodOverload**](https://dmacc.blackboard.com/webapps/assignment/uploadAssignment?content_id=_7286971_1&course_id=_102593_1&group_id=&mode=view)

Write a Console App to compute average of three numbers for four different numeric data types float, decimal, int and double. You will need to write four separate methods.   
  
For each of the four methods:

* + Name your method Average()
  + Determine the return type for the method (HINT: It might be different that data type you are averaging)
  + Include the argument list with 3 numbers as parameters
  + Compute the average of the three
  + Avoid Magic Numbers!
  + Return the average
  + Add a comment in the method body to explain the return type you selected

In the Main() method

* + Declare 3 variable of each type following naming conventions (DO NOT ASK FOR USER INPUT)
  + Declare 4 variables to store the result of each average following naming conventions
  + Make a method call to each Average() for the 4 data types, storing the result in the corresponding variable
  + Print the results for each (with enough information for the user to understand the output)

Submit your Program.cs file with the Academic Honesty Header included.