Introduction to Methods

A method is a sequence of programming statements that does a particular task, and that has a name we can use when we want to invoke that task. You have already encountered a number of built-in methods such as Math.Sqrt( ) that perform common mathematical tasks. In this assignment you will explore how to write and invoke your own methods.

When writing a program we often divide the work that we have to do into smaller tasks. Then each task is written as a method. This is particularly useful when a given task is to be invoked over and over again in a program. Let's take as an example the task of converting a temperature in degrees Fahrenheit to degrees Celsius.

Method names

When you write a method, it is important that you give the method a meaningful name. The name should reflect what it is that the method does. In our example, we are converting Fahrenheit temperatures to Celsius. A useful name then might be `ConvertToCelsius`.

The Method Prologue

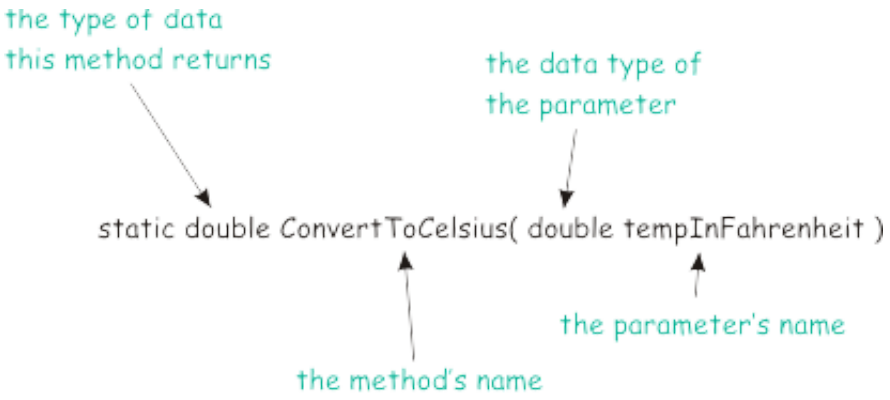
Before writing the method, you need to think carefully about what the purpose of the method is, any data that you have to give to the method so that it can do its work, and what, if anything, that the method returns to you when it is done. This information is written in the **method prologue**. Let's write the method prologue for our conversion program. We will use comments at the beginning of the method to do this. From this point on, every method that you write must have a complete method prologue as illustrated here.

```
// The ConvertToCelsius Method
// Purpose: Converts a Fahrenheit temperature to Celsius
// Parameters: A temperature in degreees Fahrenheit as a
double
// Returns: A temperature in degrees Celsius as a double
```

The Method Outline

Having written the method prologue, we are now ready to write down an outline of how our method will work. This is done by writing down the method declaration, an opening curly brace, and a closing curly brace. The method declaration tells the compiler what the name of the method is, what kind of data the method will return, and what parameters it takes. Then inside of the curly braces, write down a

pseudo-code description of what the method will do. For our ConvertToCelsius method, the method declaration would look like this:



And our completed method outline would be:

```
// The ConvertToCelsius Method
// Purpose: Converts a Fahrenheit temperature to Celsius
// Parameters: A temperature in degreees Fahrenheit as a
double
// Returns: A temperature in degrees Celsius as a double
static double ConvertToCelsius( double tempInFahrenheit)
{
   // Define constants for conversion
   // Subtract 32 from tempInFahrenheit
   // Multiply the result by 5
   // Divide the result by 9
   // Return the result
}
```

Filling in the Code

Now fill in the code that implements the pseudo-code that you have written. Our example will look like this. Notice how nicely documented our method is.

```
// The ConvertToCelsius Method
// Purpose: Converts a Fahrenheit temperature to Celsius
// Parameters: A temperature in degreees Fahrenheit as a
double
// Returns: A temperature in degrees Celsius as a double
static double ConvertToCelsius( double tempInFahrenheit)
{
   Define constants for conversion
   const double TEMP1 = 32.0;
   const double FACTOR1 = 9.0;
   const double FACTOR2 = 5.0;
   // Subtract 32 from tempInFahrenheit
   double newTemp = tempInFahrneheit - TEMP1;
   // Multiply the result by 5
   newTemp = newTemp * FACTOR2;
   // Divide the result by 9
```

```
   newTemp = newTemp / FACTOR1;
   // Return the result
   return newTemp;
}
```

Invoking a Method

To invoke, or call a method, you simply write down the method name and in the parentheses following the method name write down the name of the variable you want to pass to the method to do its work. For example, to invoke the ConvertToCelsius method, I might write

```
double tempF = 27.56;
double tempCelsius = ConvertToCelsius( tempF );
```

The Complete Program

A complete program containing the ConvertToCelsius method and the code that calls it is shown below:

```
using System;

class Program
{
   static void Main()
   {
      double tempF = 27.56;
      double tempC = ConvertToCelsius(tempF);
      Console.WriteLine("{0:F2}", tempC);
      Console.ReadLine();
   }//End Main()

   // The ConvertToCelsius Method

// The ConvertToCelsius Method
// Purpose: Converts a Fahrenheit temperature to Celsius
// Parameters: A temperature in degreees Fahrenheit as a
double
// Returns: A temperature in degrees Celsius as a double
static double ConvertToCelsius( double tempInFahrenheit)
{
   Define constants for conversion
   const double TEMP1 = 32.0;
   const double FACTOR1 = 9.0;
   const double FACTOR2 = 5.0;
   // Subtract 32 from tempInFahrenheit
   double newTemp = tempInFahrneheit - TEMP1;
   // Multiply the result by 5
   newTemp = newTemp * FACTOR2;
   // Divide the result by 9
   newTemp = newTemp / FACTOR1;
   // Return the result
```

```
    return newTemp;
}

}//End class Program
```