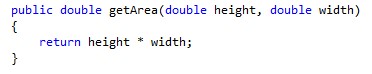
1050 Programming Logic

Lab 5 (23 points total)

Name: Matthew Mosinski

1. Identify and describe the following elements of the method header given the following method called getArea(). You must identify where they are in the method header example, other possible values and what the significance of each is (what does it mean?) (12 points – 2 points each)



1. Scope

The *scope* of the method in the given example is public. It is the first declaration in the method header. Known as an access modifier, this tell the program where the method can be called. The *public* access modifier means that the method can be called and used within the same class again or by another class that calls it. A *private* scope limits the method to only being called by code within the same class

* 1. Static vs. Non-Static

The method header in the given example is *non-static*. Unless the method header includes the word *static*, it is given that the method is *non-static*. If the method was declared as *static*, the method and the method, cannot be instantiated. *Non-static* differs in that it can be instantiated. If a method not static, an object must be instantiated to call the method from another class.

* 1. Return Type

The *return type* of the method in the given example is a *double* integer value. *Double* included in the parameters therefore, it does not need to be specified in the return type. *Double* is used to get a more precise area because it allows the system to compute larger numbers or floating-point number which include decimal numbers. *Int* could also be used instead of *double*, however it will cause an error if any franctional numbers are included.

* 1. Method Name (Identifier)

The name of the method in the given example is *getArea*(). This could be name to anything the user wants it to be. The significance of naming the method *getArea*, is so the reader or writer simply knows that this method is get the area based on the parameters.

* 1. Parameters

The parameters of the method are *double height* and *double width*, located inside the parenthesis following the method. These parameters tell the console that we will be using *height* and *width* and to use floating-point integers when dealing with *height* and *width.* These parameters could be changed based on the area of the shape or, like a method, whatever the user wants it to be.

* 1. Method Body

The method body in the given example is located inside of the curly braces following the method header. In this case the method body is *return (height \* width),* which will return the area of a 4-sided shape. If void was included I the method header, the console would not expect a return.

1. Explain the difference between a user-defined method and a method that is provided with a framework. What should we consider when creating a user-defined method? (3 points)

A user-defined method is a method that is created by the user and a framework method is one that is provided by Visual Studio. When creating a user-defined method, the writer should pick a simple name to what the method is doing, such as *getArea,* hopefully making it obvious to whoever reads or uses the code can know that this method is going to output the area of a shape. WriteLine is an example of how simple a method should be name. *“Don’t reinvent the wheel.”*

1. Discuss the difference between a static and non-static method (2 points)

There are many differences between *static* and *non-static* methods. *Static* and *non-static* methods are inversely related. *Static* methods call can be called by using the class name, where object must be created in *non-static* methods, where the user can create new keywords and variable for the object(s), in order to call the method. *Static* classes always contain static methods, whereas *non-static* does not. An easy way to remember the difference is by referred to *non-static* methods as instance methods, because they have the be accessed from an instance.

1. Use the attached code. Note: you will have to extract the code and open it in Visual Studio before starting. Add a method to the Dog class called bark(). It should have the following characteristics: (3 points)

* 1. Zero parameters
  2. No return value
  3. Should execute Console.WriteLine("{0} is Barking...", name);

1. Add a method to the Dog class called doTrick(). It should have the following characteristics: (3 points)

* + Should accept a single string parameter called trickName
  + No return value • Should execute:

Console.WriteLine("{0} is so smart! {0} is doing a(n) {1}", name, trickName);