Advanced Scripting   
Class Methods

Last Updated: 11/9/2021 11:25 AM Version 1  
Document Prepared for: CIT361 Student

# Name Zach Lucas ID 895649438

# Instructions

Save a copy of this document. Answer all questions directly in this document. You will save and upload this completed document as your homework submission.

# Overview

PowerShell classes support methods as well as properties. In this exercise you will create a class with several methods.

# Requirements

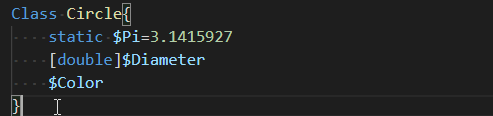
PowerShell

# Setup

# Task 1—Creating a Circle class to work with

Unlike PowerShell functions methods must declare a return the

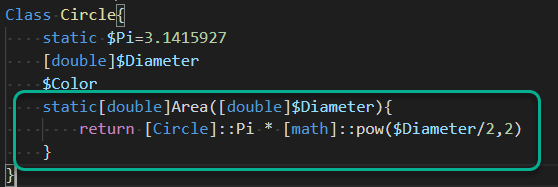
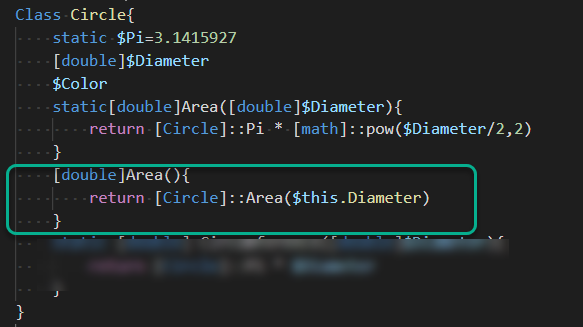
## Steps

1. In VSCode or the PowerShell ISE create a file named Methods.ps1 with the following code to start with:  
   
2. Run your code.
3. Create an instace of Circle to verify it works.
4. Set the Diameter of the circle to some value then interrogate the object to make sure it works.
5. Set the color to some value then interrogate the object to make sure it works.

# Task 2—Adding Methods to a Class

There are two types of methods instance and static. Static methods do not require an instance of the class to be used. Notice you created a static property in the class named $pi it can also be accessed without an instance of the class. Static methods can only access static properties. Remember, each time you make changes you will need to run your code again to see the changes.

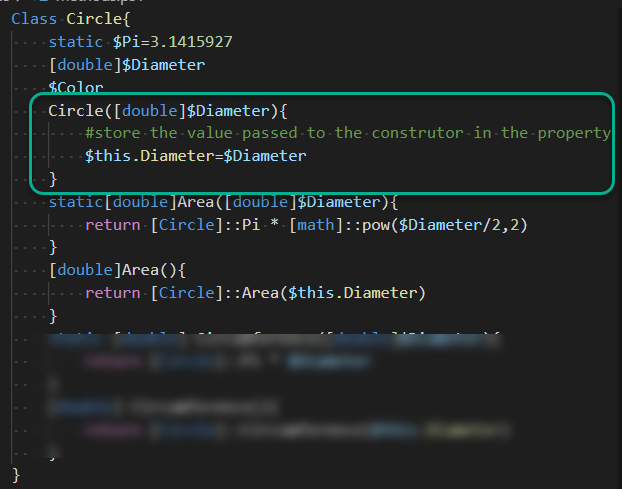
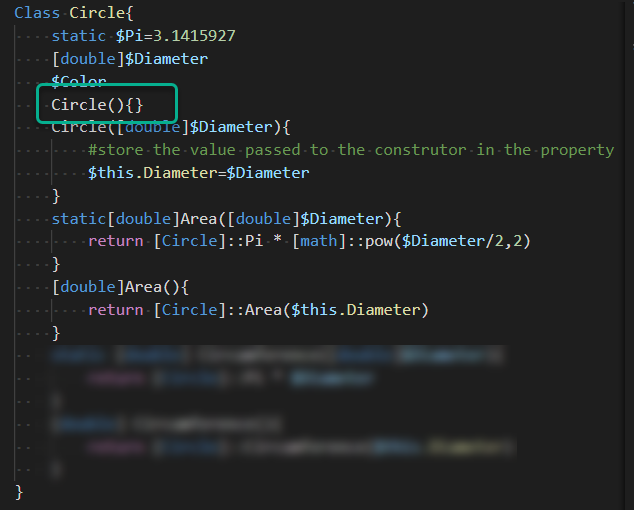
## Steps

1. Create a static method to calculate the area of the circle. You will need to pass the Diameter of the circle to the method since the method cannot access the diameter property of the class since it is not static Add the following code to your class.  
   
2. Use the new method to calculate the area of a circle that is 2.5 in diameter  
   [Circle]::Area(2.5)
   1. What is the area?4.90873859375
3. To calculate the Circumference of a circle the calculation is Pi \* Diameter. Create a static method to calculate the Circumference like you did the Area.
   1. Test it. What is the Circumference of a Circle that has a diameter of 3.7
4. Now you will create an instance method to calculate the Area. For this method you will not need to pass the Diameter since it will be in the diameter property. Also since we already have a method that calculates the are we will just us it rather than rewriting the calculations. Remember when you access a property from a method in a class you need to use the $this object. Enter the following code:  
   
5. Test the new method, first create an instance then set the Diameter.  
   $c=[circle]::new()  
   $c.Diameter=8  
   $c.Area()
   1. What is the Area? 50.2654832
6. Create an instance method to calculate the Circumference.

# Task 3—Creating Constructors

The constructor is a method that is called when an object is created. The constructor is just a method by the same name as the class.

## Steps

1. Create a constructor that allows you to pass a value for diameter in when creating the class. Add the following code.  
   
2. Run the code and test it  
   $c=[circle]::new(12)  
   $c
3. Now try to create an instance without passing the diameter  
   $c=[circle]::new()
   1. What was the result?
4. Make a constructor that does not accept any arguments. Add the following code:  
   
5. Test it  
   $c=[circle]::new()
   1. Did it work? Yes, has a diameter of 0
6. Add a constructor that accepts a diameter and color.
   1. Test your code.

# Wrap-up

Paste your complete code here

class Circle {  
 static $Pi=3.1415927  
 [double]$Diameter  
 [string]$Color  
 Circle(){} #constructor that doesn't accept any arguments  
 Circle([double]$Diameter, [string]$Color){ #$<variable>=[circle]::new(<diameter>, '<color>')  
 $this.Diameter=$Diameter  
 $this.Color=$Color  
 }  
 static[double]Area([double]$Diameter){ #calculates area  
 return [Circle]::Pi \* [math]::pow($Diameter/2,2)  
 }  
 [double]Area(){ #instance method of area  
 return [Circle]::Area($this.Diameter)  
 }  
 static[double]Circumference([double]$Diameter){ #caluclates circumference  
 return [Circle]::Pi \* $Diameter  
 }  
 [double]Circumference(){ #instance method of circumference  
 return [Circle]::Circumference($this.Diameter)  
 }  
}

# Deliverable

Upload this document with completed answers to i-learn.