Advanced Scripting   
Inheritance and Enumerations

Last Updated: 11/9/2021 8:14 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

Now you will use inheritance to create a new class called column that will be based on the circle class. You will add a Hight property and an Area method

# Requirements

PowerShell

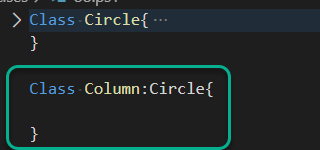
Methods.ps1 from previous lab

# Setup

Create a new file called oo.ps1 and copy the contents of your Methods.ps1 file from the previous exercise.

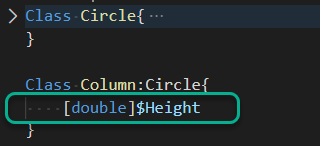
# Task 1—Create a new class than inherits from Circle

## Steps

1. Add the following code to your file  
   
2. Run your code.
3. Create a new instance of Column, and look at the results  
   $c=[Column]::new()  
   $c
4. It should look just like your circle class. Look at the members  
   $c|get-Member
5. Look at the static members  
   $c|get-Member -Static

# Task 2—Add a property

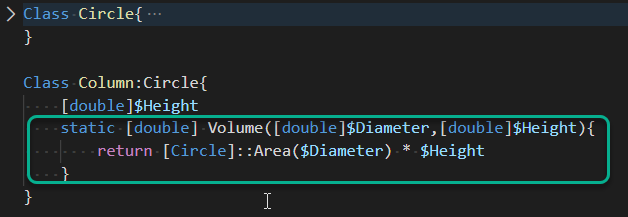
## Steps

1. Add a new height property to your column class  
   
2. Create an instance. Notice the new property  
   [Column]:new()

# Task 3—Add a Method

Now you will provide support for a Volume method. You will create a static method and an instance method.

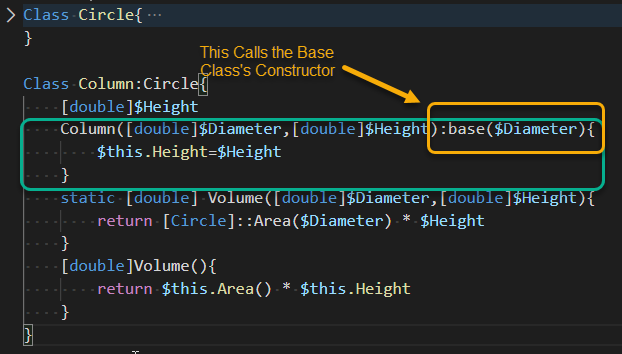
## Steps

1. The Volume of a column is the same as the Area of a Circle times it’s height, so all we need to do is call the base class method to calculate the Area, then times it by the height. To do so we will need to pass two arguments, Diameter and Height. Add the following code.  
   
2. Use the new class to calculate the volume of a column  
   [Column]::Volume(2,2)
   1. What is the Volume?
3. Now add an instance method to calculate the Volume. It will require no arguments since it will use the properties for the Height and diameter.
   1. Test it till it works.

# Task 4—Add a constructor

Adding constructors is pretty much the same as the base class, however you may wish to call the base classes constructor from your inherited class.

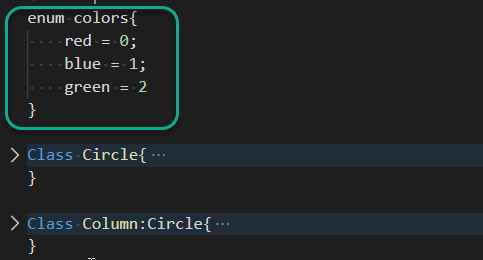
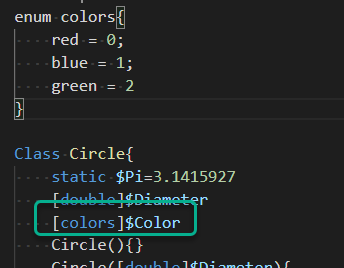
## Steps

1. Add a constructor that accepts both a Diameter and Height. It will call the base constructor and pass the Diameter to it  
    
2. Create an instance of the class passing in the Diameter and Height  
   [column]::new(4,10)
   1. Notice both the Diameter and Height have values

# Task 5—Creating an Enumeration

Enumerations are a useful programming tool that allows you to use a descriptive word to represent a numeric value. You will create an enumeration for colors then use it in your circle class.

## Steps

1. Create the following enumeration at the top of your file above the Circle class  
   
2. Run your code then look at the enum  
   [enum]::GetNames([colors])
3. Create a variable of type colors and set it to red  
   [colors]$v='red'
4. Interrogate  
   $v
   1. What is the Value of $v?Red
5. Change $c to blue  
   $v='blue'
6. See if you can convert your variable to an int  
   [int]$v
   1. Did it convert? Yes to 1
   2. What is the value if it converted?
7. Set the value to ‘yellow’  
   $v='yellow'
   1. Was it successful? There was an error
8. Set the value to 2  
   $v=2  
   $v
   1. Was it successful? Yes
   2. What was returned? green
9. Are the enums case sensitive? No
10. Change the datatype of the Color property of circle to [colors].  
    
11. Create a new instance of circle or column.
12. What is the color of the circle? red Why? It defaults to 0

# Wrap-up

Copy the contents of your oo.ps1 script here

enum colors {  
 red = 0;  
 blue = 1;  
 green = 2;  
}  
  
class Circle {  
 static $Pi=3.1415927  
 [double]$Diameter  
 [colors]$Color  
 Circle(){} #constructor that doesn't accept any arguments  
 Circle([double]$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)  
 }  
}  
  
class Column:Circle{  
 [double]$Height  
 Column([double]$Diameter,[double]$Height):base($Diameter){  
 $this.Height=$Height  
 }  
 static [double] Volume([double]$Diameter,[double]$Height){  
 return [Circle]::Area($Diameter) \* $Height  
 }  
 Column(){}  
 [double]Volume(){ #instance method of volume  
 return $this.Area() \* $this.Height  
 }  
}

# Deliverable

Upload this document with completed answers to i-learn.