Advanced Scripting   
Classes

Last Updated: 3/6/2024 10:50 PM Version 1  
Document Prepared for: CYBER360 Student

# Name Click here to enter name ID Click here to enter id

# Instructions

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

# Overview

In this exercise, you will create a simple *class*, suitable to import information from our metals file. This class will provide the “*abstract*” blueprint specification for *objects* that “*encapsulate*” appropriate data types for the metal’s properties.

# Requirements

* PowerShell
* Class sample files, from the **psfiles.zip** archive file.
* Recommended: VS Code.

# Task 1—Explore the metals data

## Steps

1. Remember where you unzipped the contents of the psfiles.zip archive.  
   You’ll must first **Set-Location** (or **cd** ) to that path, then to the data subfolder where you found the **Metals.csv** file.
2. Import the data.   
   **import-csv Metals.csv|format-table**
3. You may have already worked with this data before in another exercise. Examine the columns and complete the following table, with column names and the corresponding data types that would be the most appropriate for storing that column’s data.

|  |  |
| --- | --- |
| Column | Data Type |
| Click or tap here to enter text. | Click or tap here to enter text. |
| Click or tap here to enter text. | Click or tap here to enter text. |
| Click or tap here to enter text. | Click or tap here to enter text. |
| Click or tap here to enter text. | Click or tap here to enter text. |

# Task 2—Create A Metal Class

Create a class to hold the metal data. I’ll get you started. You will need to complete the properties.

## Steps

1. Create a file named **ImportMetals.ps1**.
2. To start your new script, create a class to hold metal data that contains a type restricted property for each column in the Metals file. Each property name must match the column name:  
   **Class Metal{**   
    **[string]$Symbol**   
   #Add other Properties as appropriate   
   **}**
3. Next in the script, let’s create instances of the class. Learn three ways to do so:
   1. One way: call the **new** method:   
      $m1=[Metal]::New()
      1. Set some properties:   
         $m1.Symbol='Au'   
         $m1.Name='Gold'   
         $m1.MeltingPoint=1945   
         $m1.SpecificGravity=19.3
      2. Take a look at the results:   
         $m1
   2. Another way: use the **New-Object** cmdlet:   
      $m2=New-Object Metal   
      $m2.Symbol='Ag'   
      $m2.Name,$m2.MeltingPoint,$m2,SpecificGravity='Silver',1762,10.6   
      $m2
   3. A third way: initialize by *casting* a hashtable into an object:   
      $m3=[Metal]@{Symbol='Sn';Name='Tin';MeltingPoint=450;SpecificGravity=7.3}   
      $m3
4. Save and execute your script. Its output should look pretty much like this:   
   A screen shot of a black background

   Description automatically generated   
   (Fix your script if it’s not yet working.)
5. Now let’s add two more lines to the end of your script.
   1. Import the data and cast to our new Metal type. (Again, use the appropriate path to the file.)   
      $Metals=Import-Csv Metals.csv | ForEach-Object {[Metal]$\_}
   2. View the data type of the first element in our new **$Metals** array variable:   
      $Metals[0].GetType().FullName
6. Save and execute your updated script. If you encounter errors, fix any bugs or typos.
   1. Look at the last line of the output. What’s the full name of the first element’s data type? Click or tap here to enter text.
   2. Determine: do the rest of the elements of the array contain the same data type? Click or tap here to enter text.
7. Copy your completed script here:   
   Click or tap here to enter text.

# Deliverable

Upload this document with completed answers to I-Learn Canvas.