<u>Role Playing Game</u>

By Max Santomauro

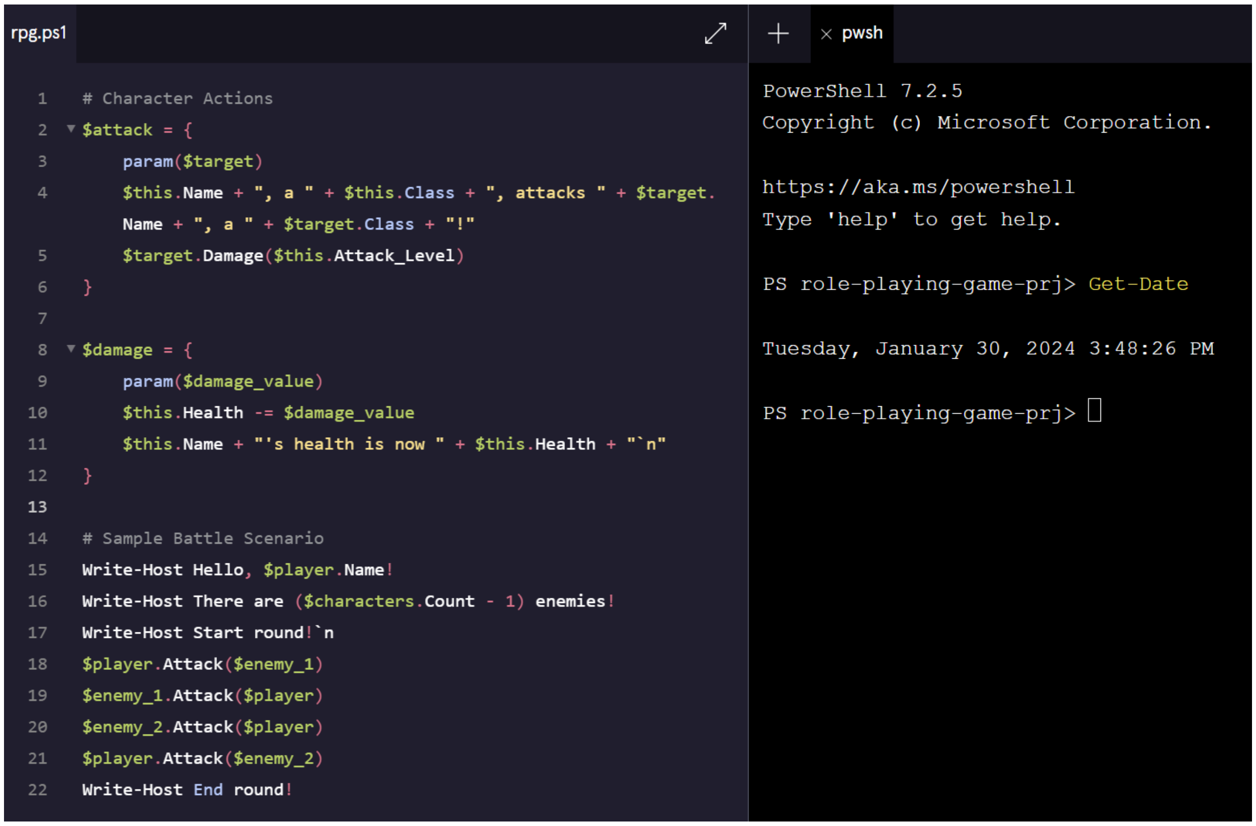
- **Note**: This project is provided by Codecademy (https://www.codecademy.com/)
- **Note**: The environment of this project involves an object-oriented programming language (script **rpg.ps1**) with the use of the PowerShell CLI.

Project Description

Marquis is building a collection of apps and utilities for Windows. He has an unfinished project which he wants you to complete. This project is a small text-based role-playing game.

Marquis wants me to create a few custom character objects for his game. In particular, one character for the player and two enemy characters the computer will control. He has written a few methods for the characters and a sample battle scenario. I need to add the methods and properties to the characters as well.

Project Start Timestamp and Environment Setup

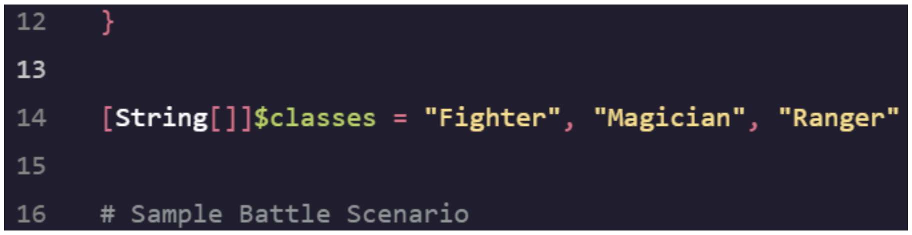


```powershell
# Character Actions
$attack = {
    param($target)
    $this.Name + ", a " + $this.Class + ", attacks " + $target.
    Name + ", a " + $target.Class + "!"
    $target.Damage($this.Attack_Level)
}

$damage = {
    param($damage_value)
    $this.Health -= $damage_value
    $this.Name + "'s health is now " + $this.Health + "`n"
}

# Sample Battle Scenario
Write-Host Hello, $player.Name!
Write-Host There are ($characters.Count - 1) enemies!
Write-Host Start round!`n
$player.Attack($enemy_1)
$enemy_1.Attack($player)
$enemy_2.Attack($player)
$player.Attack($enemy_2)
Write-Host End round!
```

```
PowerShell 7.2.5
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS role-playing-game-prj> Get-Date

Tuesday, January 30, 2024 3:48:26 PM

PS role-playing-game-prj>
```

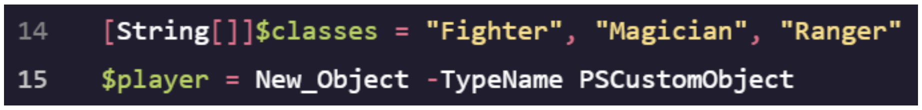
Create Player Character

1. I first created classes for the characters with a strongly typed array of **string** type called **classes**. The items are as follows: **Fighter**, **Magician**, and **Ranger** after the **attack** object. This is seen in line 14 of the script.



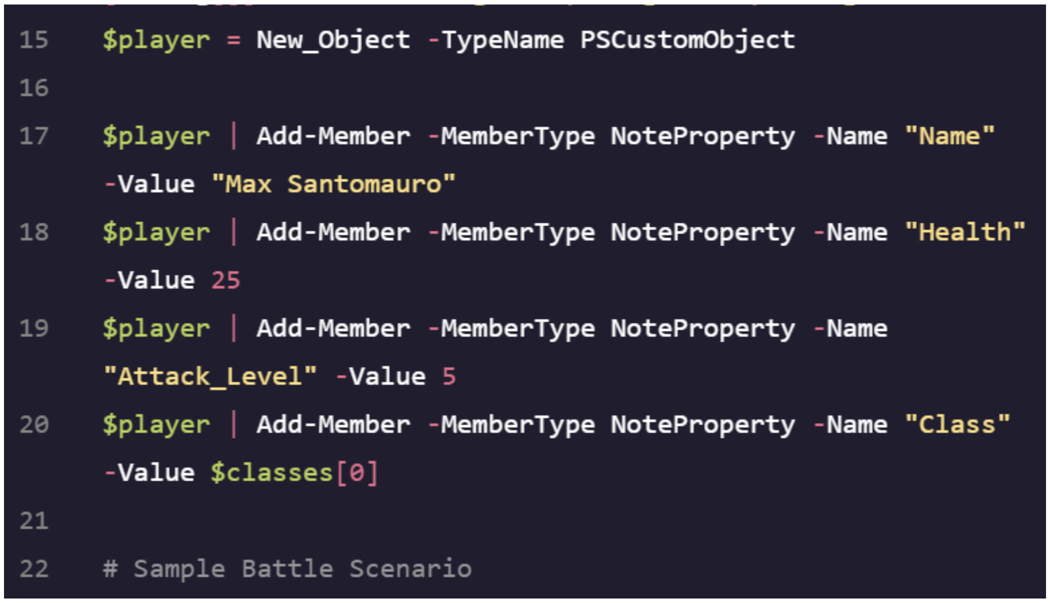
```
12    }
13
14    [String[]]$classes = "Fighter", "Magician", "Ranger"
15
16    # Sample Battle Scenario
```

2. Then I created a custom object called **player** using the **New-Object** cmdlet. This is seen in line 15 of the script.



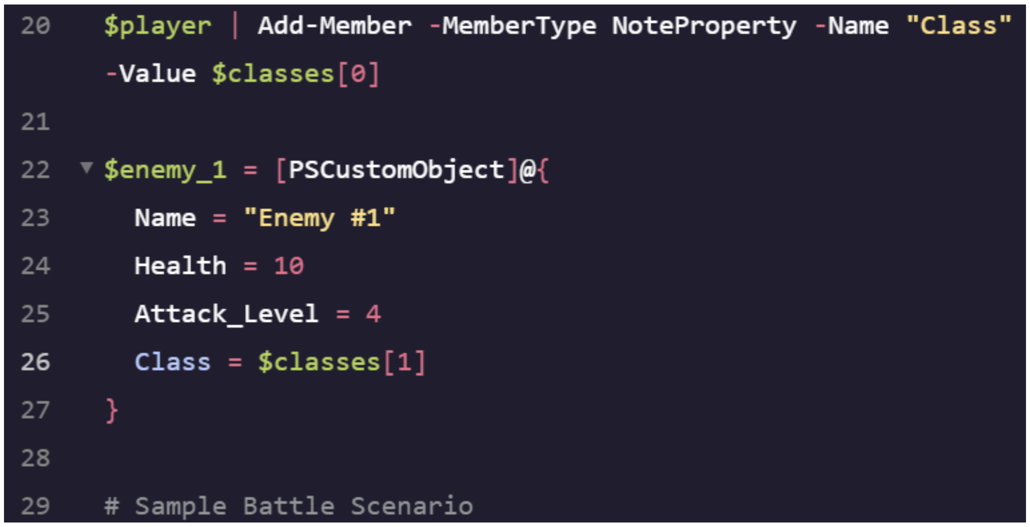
```
14    [String[]]$classes = "Fighter", "Magician", "Ranger"
15    $player = New_Object -TypeName PSCustomObject
```

3. Next I added some properties to this custom object that describes the player character to **player** with the **Add-Member** cmdlet. The **values** I added are: **Name**: (string of my choosing), **Health: 25**, **Attack_Level**: **5**, **Class**: the first item in the **classes** array (index 0). This is seen in lines 17 through 20 of the script.



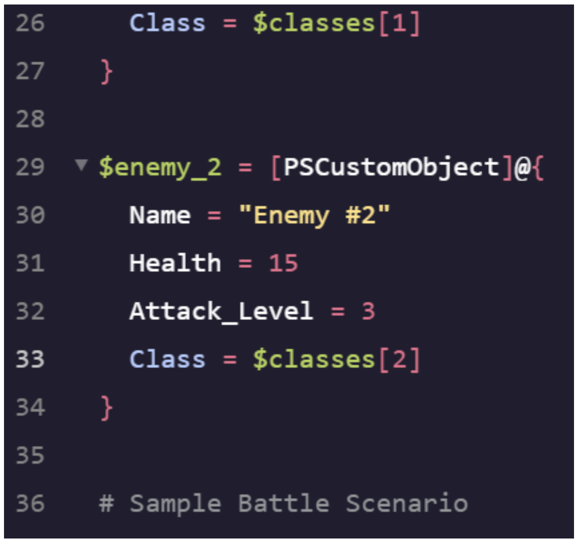
```
15    $player = New_Object -TypeName PSCustomObject
16
17    $player | Add-Member -MemberType NoteProperty -Name "Name"
      -Value "Max Santomauro"
18    $player | Add-Member -MemberType NoteProperty -Name "Health"
      -Value 25
19    $player | Add-Member -MemberType NoteProperty -Name
      "Attack_Level" -Value 5
20    $player | Add-Member -MemberType NoteProperty -Name "Class"
      -Value $classes[0]
21
22    # Sample Battle Scenario
```

4. I also created custom objects called **enemy_1** for enemy characters using a hashtable and assigned it the following properties: **Name** : **“Enemy #1”**, **Health** : **10**, **Attack_Level** : **4**, **Class**: the second item in the **classes** array (Index 1). This is seen in lines 22 through 27 of the script.



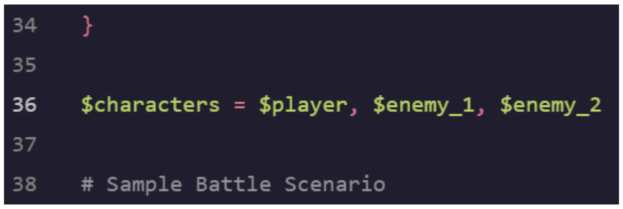
```
20    $player | Add-Member -MemberType NoteProperty -Name "Class"
      -Value $classes[0]
21
22  ▾ $enemy_1 = [PSCustomObject]@{
23      Name = "Enemy #1"
24      Health = 10
25      Attack_Level = 4
26      Class = $classes[1]
27    }
28
29    # Sample Battle Scenario
```

5. I created a second object called **enemy_2** for enemy characters using a hashtable and assigned it the following properties: **Name** : **“Enemy #2”**, **Health** : **15**, **Attack_Level** : **3**, **Class**: the third item in the **classes** array (Index 2). This is seen in lines 29 through 34 of the script.
    a. **Note**: copy/pasting and modifying custom object **enemy_1** for custom object **enemy_2** makes this process easier.



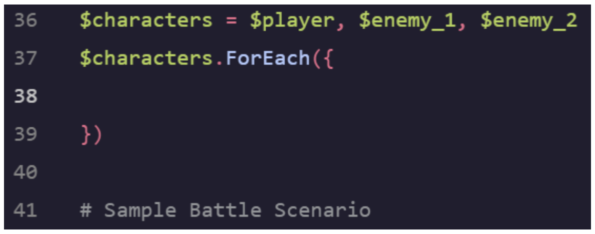
```
26      Class = $classes[1]
27    }
28
29  ▾ $enemy_2 = [PSCustomObject]@{
30      Name = "Enemy #2"
31      Health = 15
32      Attack_Level = 3
33      Class = $classes[2]
34    }
35
36    # Sample Battle Scenario
```

6. In order to keep track of all the characters, I created an array with all the custom objects by assigning them to the array **characters**. This is seen in line 36 of the script.



```
34   }
35
36   $characters = $player, $enemy_1, $enemy_2
37
38   # Sample Battle Scenario
```

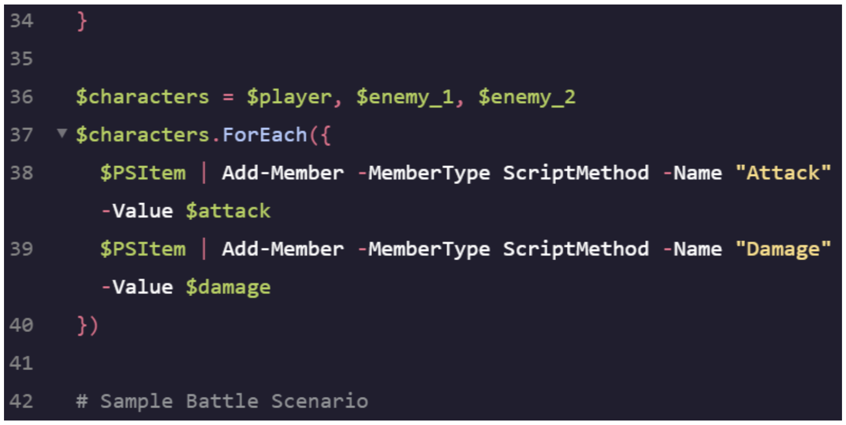
7. I iterated over the **characters** array to add methods to all of the custom objects with the **ForEach** method and with empty curly braces. This is seen in lines 37 through 39 of the script.



```
36   $characters = $player, $enemy_1, $enemy_2
37   $characters.ForEach({
38
39   })
40
41   # Sample Battle Scenario
```

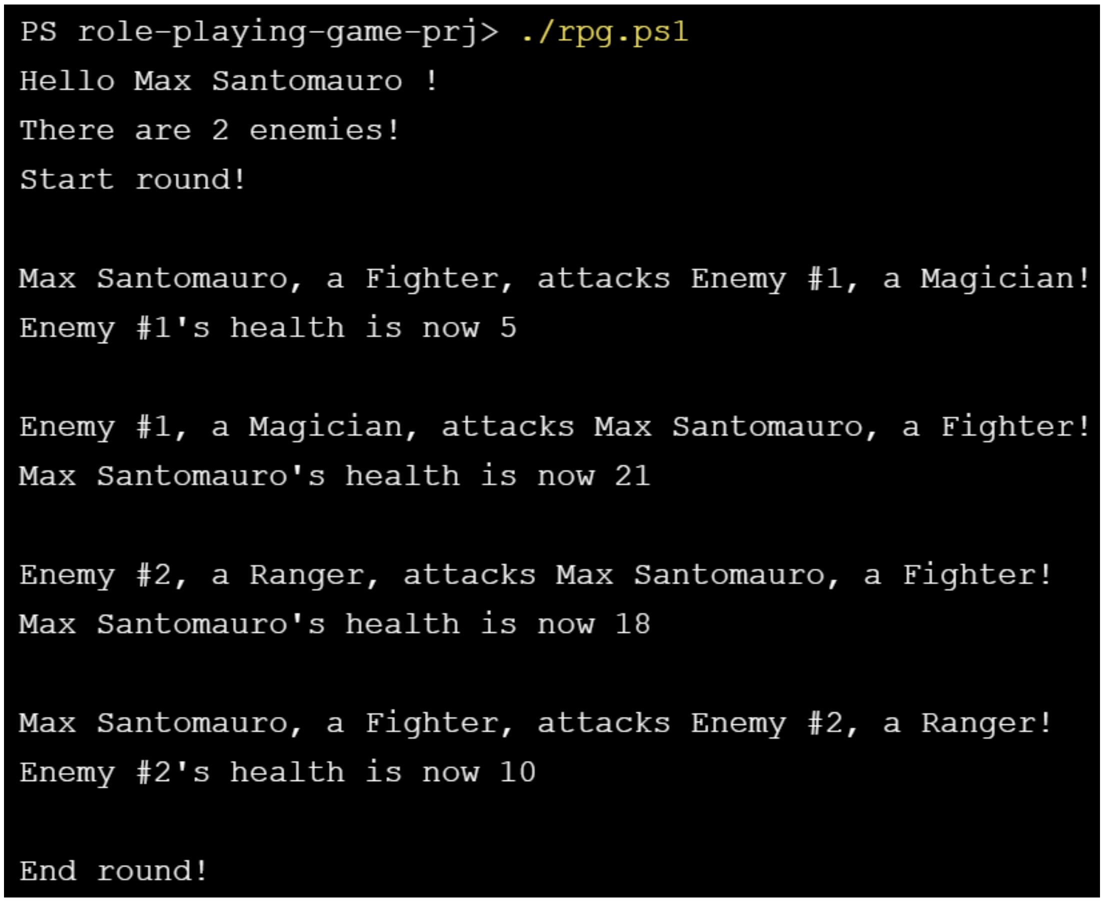
8. Marquis wrote two **ScriptBlock's** he wants me to add to each character object: **attack** and **damage** (which can be observed in lines 2 through 12). I piped **Add-Member** to **PSItem** to add methods to all the character objects I created inside the curly braces of the **ForEach** method. The added methods are:
   a. **Attack** with the value of the **attack** ScriptBlock.
   b. **Damage** with the value of the **damage** ScriptBlock.

   Changes shown from line 37 to line 40.



```
34   }
35
36   $characters = $player, $enemy_1, $enemy_2
37   $characters.ForEach({
38     $PSItem | Add-Member -MemberType ScriptMethod -Name "Attack"
       -Value $attack
39     $PSItem | Add-Member -MemberType ScriptMethod -Name "Damage"
       -Value $damage
40   })
41
42   # Sample Battle Scenario
```

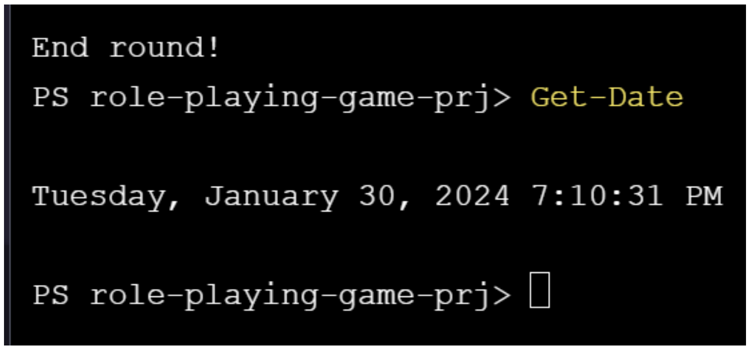
9. This is the result of the script when called in the PowerShell terminal:



```
PS role-playing-game-prj> ./rpg.ps1
Hello Max Santomauro !
There are 2 enemies!
Start round!

Max Santomauro, a Fighter, attacks Enemy #1, a Magician!
Enemy #1's health is now 5

Enemy #1, a Magician, attacks Max Santomauro, a Fighter!
Max Santomauro's health is now 21

Enemy #2, a Ranger, attacks Max Santomauro, a Fighter!
Max Santomauro's health is now 18

Max Santomauro, a Fighter, attacks Enemy #2, a Ranger!
Enemy #2's health is now 10

End round!
```

   a. **Note**: I corrected a mistake in line 15 in which the **New-Object** cmdlet was incorrectly types as New_Object .

Project End Timestamp



```
End round!
PS role-playing-game-prj> Get-Date

Tuesday, January 30, 2024 7:10:31 PM

PS role-playing-game-prj> 
```