Week 3: Day 005.1 – C++: Classes and Objects

Create a Visual Studio Solution and Project for Day 005. Compile and test your program for this exercise as you progress through the steps. Fix any issues with each step before moving onto the next!

Create a class named `Player`. Create a `.h` file for the declaration, and a `.cpp` for the implementation.

Add private fields to the `Player` classes that store the player's health, experience level, and rage level. Pick appropriate types for these data members. Add public accessor methods for each of these fields. Also, ensure the get methods maintain `const` correctness.

Add a method to the `Player` class called `PrintStats()` which outputs the current player's statistics (health, experience level and rage level) to the console.

Add a default constructor to the `Player` class that prints out "Player Object Created" to the console, and a destructor that prints out "Player Object Destroyed!".

Add another `.cpp` file, called `main.cpp`. Inside this file, declare the `main` function. In the `main` function, declare a local instance of the `Player` class named `testPlayer`. Remember to `#include "player.h"` at the top of the `main.cpp` file!

Call the `PrintStats` method on the `testPlayer` instance.

Call the Set methods to mutate the `testPlayer` object.

Call the Get methods individually, and print the results return from them in the `main` function.

Finally call the `PrintStats` method again.

Week 3: Day 005.2 – Local vs Freestore Objects:

Add a new project to your Day 005 Visual Studio solution.

Copy the `player.cpp`, `player.h` and `main.cpp` from exercise 1. Ensure you actually copy the files, and do not just accidently create symbolic links to the first project's files!

In the main function, declare a pointer to a Player called `pPlayer`. Assign the pointer the null pointer value.

Next, allocate a Player object on the Freestore with **new**.

Invoke the method of **`pPlayer`** using the **`->`** operator. Again test the functionality of the **`PrintStats`**, and accessor methods.

At the end of the **`main`** function. Deallocate the **`Player`** object from the Freestore using the **`delete`** command.

Contrast the different between a locally stored Player object (**`testPlayer`**), and one stored on the Freestore (***pPlayer**).

Week 3: Day 005.3 – Object Factory:

Add a new project to your Day 005 Visual Studio solution.

Create four new classes in this project:

- **`Entity`**
- **`Zombie`**
- **`HealthPack`**
- **`AmmoPack`**

Give each class appropriate member data, methods, and constructors. Ensure each class has a header file (**.h**) and a source file (**`.cpp`**).

Ensure **`Zombie`**, **`HealthPack`** and **`AmmoPack`** are all subclasses of **`Entity`**. The **`Entity`** class should be abstract.

Add a file called **`main.cpp`**. In this file declare and define a **`main`** function.

In the main.cpp file, declare an enumeration called **`EntityType`** with three elements, **`ZOMBIE`**, **`HEALTHPACK`**, and **`AMMOPACK`**.

Add a function with the signature **`Entity* CreateEntity(EntityType entityType);`** which takes in a enumerated value representing the type of Entity to create. This function must allocate the appropriate **`Entity`** on the Freestore, and then return the pointer to this new **`Entity`** to the calling method.

In the **`main`** function, call **`CreateEntity`** with different values of **`EntityType`**. Store the pointer returned in the **`main`** function, and then manipulate the entity by calling methods that belong to the **`Entity`** via the pointer.

Week 3: Day 005.4 – String Manipulations:

Add a new project to your Day 005 Visual Studio solution.

Using C++, implement the classic word game of Hangman, you do not need to use classes and objects, simple function will do. The purpose is to experiment with using the **`std::string`**.

Use a **`std::string`** to store the secret word, and another **`std::string`** to store the word as the letters are revealed.

Output the game in the following style:

```
  +-----+
  |     |        Secret Word:  _ N I _ E _ _ I _ _
  |     O
  |   \ | /
  |    \|/       Incorrect Guesses:
  |    /
  |   /             A, O, M, B, L, P, D, F, H
  |
  |
--+-----------
```

Week 3: Day 005.5 – STL Containers and Iterators:

Add a new project to your Day 005 Visual Studio solution.

Add a **`main.cpp`** to the project.

Create a **`std::vector`** of **`int`**s. Push a variety of integer values into the container.

Next, iterate through the container and print out the contents of each element in the container.

Then write code to do the same operations with the **`std::stack`**, and **`std::queue`**. Experiment with each type of container.

`#include` the **`algorithm`** header, and try experimenting with the **`std::sort`** and **`std::random_shuffle`** methods on the **`std::vector`** container.