**Assignment 2 – Classes and Objects**

Submit to MUOnline as a compressed (.zip) file containing your code project.

For this particular assignment, **no user input is necessary**. Simply demonstrate that your classes and functions work, including with bad values (like bob.SetHealth(-100) should clamp his health to 0). You may hardcode the names, numbers, etc. (only in main() when you create the objects). But if you want to add user input for fun, by all means go ahead! 😊

|  |  |  |
| --- | --- | --- |
| 1. | Create a class representing a **Character** in a game. At a minimum a character should have a name and numeric values for health, stamina, magic, and attack power.  All variables should be private. A character should also have public functions for getting (or accessing) all of their member variables, and functions for setting (or ‘mutating’) their member variables. Setter functions should validate that the result is valid (for instance setting their hp below 0 shouldn’t be allowed).  The character should have a **default** constructor that initializes all their member variables to appropriate values, and **optionally** accepts a string for setting the character’s name as soon as they are created. | 10% |
|  |  |  |
| 2. | Create a class representing an **Ability**. It should have a name, and numeric values for how much it changes a character’s health, stamina, and magic, and numeric values for how much magic and stamina it costs. All variables should be private with public get/set functions. Negative values should be allowed for their effects (for instance, if you name a variable ‘healthEffect’, a ‘First Aid’ Ability would have a positive ‘healthEffect’ amount, but a ‘Fireball’ Ability would have a negative ‘healthEffect’ amount. Their costs should never be less than 0 or greater than 100. A constructor should initialize members appropriately. | 10% |
|  |  |  |
| 3. | Add a member variable of type Ability to the Character class. Add a member function **SetAbility** that takes 1 Ability object and stores it in the character, and an overload of **SetAbility** that takes the name and other properties of an ability (instead of 1 ability object that already has all that data) and sets it on the Character’s ability. A character will just have 1 ability; having more than 1, choosing which 1, acquiring them at run-time, etc. will be challenges for another day…  Example of how you would call these 2 overloads:  //Overload 1  Character superman;  Ability heatVision;  heatVision.SetHealthEffect(-5);  **superman.SetAbility(heatVision);**//copies heatvision ability into superman’s ability.  //Overload 2:  Character dragon;  **dragon.SetAbility(“Fire Breath”, -10, -10, 0);**//Sets dragon’s ability with the name “Fire Breath”, -10 health effect, -10 stamina effect, and 0 magic effect. | 20% |
|  |  |  |
| 4. | Add a function **IsAlive** to the character class that returns whether or not the character is still alive based on their health variable.  Add a function **Display** to both the Character class and the Ability class, to conveniently print out their information. The Character’s Display() should indicate whether the character is still alive.    Add a function **Attack** to the character class so it can attack and do damage to another character with their ‘attackPower’ value (Hint: make sure it doesn’t attack a copy of another character…). Include some console output so it says something like ‘Bob hits Modern Sexy Vampire for 10 damage!’  Finally, add a function **UseAbility** to the character class, so that the character can use their ability. It should take 1 parameter of the **Character** type, and effect that Character (which may be the same character using the ability; i.e. ‘Bob heals Bob for 5 hp’. It should use the properties of the ability (like how much ‘healthEffect’ it does) and change the appropriate values of the character. For instance,  Ability fireSpell;  fireSpell.SetHealthEffect(-5);  character.SetAbility(fireSpell); **character.UseAbility(vampireCharacter);**  would result in vampireCharacter losing 5 health. Include some console output so it says something ‘Fireball cast on Modern Sexy Vampire. Modern Sexy Vampire’s stats are now: (and then call the Display function for the character)’. | 10%  10%  10%  10% |
|  |  |  |
| 5. | Remember that making classes just creates data types we can use in a program. If main() is empty, we still have no ‘program’.  So create a program that instantiates some Characters (at least 2) and give them abilities that do different things. Test all the member functions of part 3 and 4 above. Have the 2 characters attack each other and use both characters’ abilities. Display() their info before and after each attack to show how the objects are changing. Show at least one dying to test the IsAlive function. | 20% |