Project 3 – Inheritance

Implement Fantasy Combat Game

Goals

- Identify requirements for a program using polymorphism
- Create a program to demonstrate your class hierarchy

Requirements

In this project, you will create a simple **class hierarchy as the basis for a fantasy combat game**. Your 'universe' contains Vampires, Barbarians, Blue Men, Medusa and Harry Potter. Each has characteristics for **attack, defense, armor, and strength points** as follows.

Type	Attack	Defense	Armor	Strength Points
Vampire[1]	1d12	1d6* Charm	1	18
Barbarian[2]	2d6	2d6	0	12
Blue Men[3]	2d10	3d6	3	12 *Mob
Medusa[4]	2d6* Glare	1d6	3	8
Harry Potter[5]	2d6	2d6	0	10/20*Hogwarts

1. Suave, debonair, but vicious and surprisingly resilient!
2. Think Conan or Hercules from the movies. Big sword, big muscles, bare torso.
3. They are small (6" tall), fast and tough. So they are hard to hit and can take some damage. As for the attack value, you can do a LOT of damage when you can crawl inside the armor or clothing of your opponent.
4. Scrawny lady with snakes for hair. They help with fighting. Just don't look at her!
5. Why are you reading this? How can you not know who Harry Potter is?

"3d6" is rolling three 6-sided dice, "2d10" is rolling two 10-sided dice, etc.

*Charm: Vampires can charm an opponent into not attacking. For a given attack there is a 50% chance that their opponent does not actually attack them.

*Glare: If a Medusa rolls a 12 in attack, then the target has looked her in the eyes and is turned to stone. The Medusa wins! If Medusa uses Glare on Harry Potter on his first life, then Harry Potter get back to life.

*Mob: The Blue Men are actually a swarm of small individuals. For every 4 points of damage (round down), they lose one defense die. For example, when they reach strength of 8 they only have 2d6 for defense.

*Hogwarts: If Harry dies (i.e. strength <=0), he immediately recovers and his total strength becomes 20. If he were to die again, then he's dead.

NOTE: The sample creatures are unbalanced intentionally. This will help you in debugging your program! Some will win a lot, and others will lose a lot.

To resolve an attack, you will need to **generate 2 dice rolls**. The attacker rolls the appropriate number and type of dice under Attack. The defender rolls the appropriate number and type of dice under Defense. You will **subtract the Defense roll from the Attack roll**. That is the damage to the defender.

Each class only has its own information or data. When O1 is fighting O2, your program should call O1's attack function. It will return the damage inflicted. Then O2's defense function will take the damage inflicted, roll the specified dice and subtract the damage points from the defense. To apply the damage, you subtract the Armor value. The result is then subtracted from the Strength Points. That value becomes the new Strength Points for the next round. If Strength Points goes to 0 or less, then the character is out of the combat. For example, if one object receives 9 points of damage and rolls 3 for its defense, and has an armor of 4 and strength point of 8, it would take 9 subtract 3, and then 4 for the armor, to receive 2 points of damage, and its new strength point will be 8-2=6.

You need to create a **Creature base class**. Then you will have a subclass for each of these characters. Note that the Creature class will be an **abstract** class. For our purposes right now, each subclass will **vary only in the values in the table**. It is part of your design task to determine what functions you will need.

To play the game, write a menu. Display five fighters by their names and prompt the user to select two fighters to fight one another. Students must account for two fighters of the same type. Randomly select one fighter to attack first. The fighters will take turns fighting each other until one's Strength point is zero or negative. (You do not have to display results of each round of fighting, but you can do that for the purpose of debugging.) Then display the winning fighter to the screen. Ask users to play again or exit the game. This is the **first stage** of a larger project. Please **do not add** any creatures of your own.

You must **include your design** in your reflections document. In that document, you must discuss what the original design is, and how it changes as you worked through the problem. You must also **include a test plan** in that document. The test plan should cover all logic paths (you should have each character type have combat with all character types, including another of its own). The program has a random element. You will need to address that in your test plan. It will also affect debugging. Your design should address this (potential) problem as well. Also, you need to show your test results with a brief discussion. Remember to submit this document as PDF file.

It is not hard, just a lot to think about. The TAs will be asked to grade your project against your design. So please do not just throw together some random stuff. If you give us a random design, you will need to explain each step in how you got to the code submitted. In other words, that will make it much more difficult. So, learn a good habit and think about it before you start coding.

What you need to submit:

- Your program file(s) with the implementation of these five creatures inheriting from a single parent and a makefile to run your program.
- Your reflections document

Suggestion: Create your design before coding anything! You should even be outlining your test plan. The grading is set up to encourage you to develop your program incrementally! Start with the base and Barbarian classes. Create the testing program that runs sample combats for you to test your code. How do you handle random die rolls when testing your code? Then do the others. At each step of the coding process, make notes about what worked, what has been changed from your design. Doing this will make writing the reflections easier.

Grading

- Programming style and documentation (10%)
- In each of these, the virtual attack and defense functions must work correctly:
  - ✓ Create the base class (10%)
  - ✓ Create the Barbarian class (10%)
  - ✓ Create the Vampire class, overload or redefine defense function (10%)
  - ✓ Create the Blue Men class (10%)
  - ✓ Create the Medusa class, overload or redefine attack function (10%)
  - ✓ Create the Harry Potter class (10%)
- Create a test driver program to create character objects which make attack and defense rolls required to show your classes work correctly (20%)
- Reflections document to include the design description, test plan, test results, and comments about how you resolved problems during the assignment (10%)