Assignment 3 – Inheritance And Polymorphism

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

You will submit a reflections document with every assignment. This is you chance to explain why you did what you did. Or why something you tried didn't work. You can explain what you did to test your program to make sure it works. You can also document features or bugs that inhibit the program in some way.

You will create a simple class hierarchy as the basis for a fantasy combat game. Your 'universe' contains the following creatures. Each will have characteristics for attack, defense, armor, and strength points.

Type	Attack	Defense	Armor	Strength Points
Vampire[1]	1d12	1d6 *Charm	3	8
Medusa[2]	2d6 *Glare	1d6	3	8
Bubba[3]	2d6	2d6	0	12

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

*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. **You do NOT need to modify the attacking Creature to implement this change.**

*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! **You do NOT need to modify the attacking Creature to implement this change.**

To resolve an attack you will need to generate 2 dice rolls. The attacker rolls the appropriate number and type of dice under Attack. That amount of damage is passed to the defender in the defender object's defense function. The defender rolls the appropriate number and type of dice under Defense. You subtract the Defense roll from the Attack roll. That is the damage. 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.

HINT: Carefully think through how the attack and defense functions will interact. Something like this:
```
hits = Creature1.attack();
Creature2.defense(hits);
```

You need to create a Creature class. Call it what you choose. Then you will have a subclass for each of these characters. The parent class will be an **abstract class**. Each subclass will vary only in the values in the table. Since each starts with the same data elements you will only need one constructor. It is part of your design task to determine what functions you will need. Definitely attack() and defense()? Incorporate them into your class hierarchy.

To manage the combat you will need two pointers to Creature, say fighter1 and fighter2. You will instantiate each to the appropriate derived class object. All that your main function should do is to call the attack and defend functions. All other actions should be enclosed in each derived class.

You must complete your design document. It must include a class hierarchy. In your reflections you can discuss how the original design may have changed as you worked through the problem. You must also submit a test plan. The test plan should cover all logic paths. So you should have each character type have combat with all character types (including another of its own). Remember to submit these documents as PDF files.

To test your classes create a program that instantiates 2 creatures of the type specified by the user. Conduct rounds of combat until only one has strength points remaining. Can you have a draw? That is can both run out of strength points in the same round? Look at your design for the answer. ☺

It is not hard, just a lot to think about. The sooner you start the better it will be! The TAs will be asked to grade your project against your design so please do not just throw together some random stuff so you have a file to submit. No, you are not required to implement only the design you submit. BUT, your reflections will need to explain the difference. So the old adage garbage in, garbage out will not apply here. 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. ☺

HINT: This 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. It is not hard but you need to think about it.

What you need to submit:

> Your program file(s) with the implementation of these five creatures inheriting from a single parent.
> Your design document (including the class hierarchy)*
> Your test plan*
> Your reflections document- including the design and test documentation

1. Really? You don't know what a vampire is? ☺ Although think Bram Stoker, not the modern variety.

2. Scrawny lady with snakes for hair. They help with fighting. Just don't look at her!

3. You know him. The big good-hearted friend who sometimes says too much. ☺

======

Grading:

- programming style and documentation (10%)
- In each of these the virtual attack and defense functions must work correctly-
  - create the base class and the Barbarian class (20%)
  - create the Vampire class including the modified attack function(15%)
  - create the Medusa class (15%)
- 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 (20%)

Suggestion: 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 toe test your code. How do you handle random die rolls when testing your code? Then do the others.

Hint: Create your design before coding anything! You should even be outlining your test plan. At each step of the process make notes about what worked, what changed. Doing this as you progress will make writing the reflections easier.