Assignment 3 – Inheritance

**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 your 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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Attack</th>
<th>Defense</th>
<th>Armor</th>
<th>Strength Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gollum</td>
<td>1d6 *Ring</td>
<td>2d6</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Barbarian</td>
<td>2d6</td>
<td>2d6</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>Baba Yaga</td>
<td>1d6 *Soul</td>
<td>1d10</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>Blue Men</td>
<td>2d10</td>
<td>3d6*</td>
<td>3</td>
<td>12 *Mob</td>
</tr>
<tr>
<td>Unicorn</td>
<td>1d20+2d6*</td>
<td>2d6 *Unicorn</td>
<td>3</td>
<td>18</td>
</tr>
</tbody>
</table>

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

*Soul- Baba Yaga feeds on the pain and suffering of others. When she makes a successful attack her attack roll is applied to her opponent, but she gains points equal to the points lost by her opponent. For example, she makes an attack and 3 points of damage are inflicted. She increases her strength points by 3 also. Her total can exceed 12. She wears no physical armor but has many spells and enchantments to protect her.

*Unicorn- Unicorns are peaceful creatures. They will only defend until they received damage. The following turn they will attack with their horn (1d20) and their hooves (2d6). Unless they face another Unicorn. They are very territorial and will immediately fight another unicorn to the death.

*Ring- Gollum has the one ring. It can make him invisible. That makes him harder to hit. It also allows him to occasionally surprise his opponent. There is a 5% chance he makes an attack with 3d6.

*Mob- The Blue Men are actually a swarm of small picties. For every 4 points of damage (round down) they lose 1 defense die. Such as, when they reach strength of 8 they only have 2d6 for defense.

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 subtract the Defense roll from the Attack roll. That is the damage.

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, then roll the specified dice and subtract the points for 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. If it receives 8 points of damage and rolls 3 for it’s defense and has an armor of 3 it would take 8 subtract 3 and then 3 for the armor to receive 2 points of damage.

You need to create a Creature class. Then you will have a subclass for each of these characters. Note that the Creature class will be an abstract class with both the virtual attack and virtual defense functions. You will never instantiate one. 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. The only values that can change are the Strength Points and the number of dice for the Gollum and Hydra special abilities.

This is the first stage in what will be a larger project. Please do not add any creatures of your own.
You must complete your design document. In that document and 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.

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. Slavic (Russian) witch.
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. 😊 And yes they are the Nac Mac Feegle of Discworld. I just wanted a shorter name for your project.
4. He is small, not fast, and not strong.

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.

Grading:

- programming style and documentation (10%)
- In each of these the virtual attack and defense functions must work correctly-
  - create the abstract base class and the Barbarian class (15%)
  - create the Gollum class including the modified attack function(10%)
  - create the Baba Yaga class (10%)
  - create the Blue Men class (10%)
  - create the Unicorn class including the modified defense function(10%)
- create a test driver program to create character objects which make attack and defense rolls required to show your classes work correctly (15%)
- 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.