Richard Moot
CS162- 400
November 8, 2015

Assignment 3 – Polymorphism & Inheritance

Design

For this project, I want to keep the inheritance between classes minimal, since I don't want the class to become too vertical and cause conflicts. The Creature abstract class will be the main class that all other classes inherit from with a separate Combat class that is intended to be the driver for testing everything and negotiating combat between all of the creatures.

Creature Class -> All other classes will be the main way for everything to work and each individual creature will be responsible for how they are different from the main creature class. The main methods that will be shared amongst all classes will be the following: attack(Creature), defense(), and takeDamage(). With these methods, I should be able to successfully interact between all of the classes.

The combat class will be responsible for driving the program and handling the battles between creatures. The battles will happen simultaneously, so each turn will evaluate the damage that was done by each character and applied to one another (no turn based implementation). The combat will terminate once one of the creatures no longer has any strength left.

For the Goblin's special attack, I will create and additional bool member variable that should allow me to track whether an Achilles has been applied to any enemy. It makes more sense for the Goblin to remain aware of this rather than every other class, since it would only be relevant when a Goblin is in combat.

For the Shadow, I plan on overriding their takeDamage() method to handle when a dodge should be taking place. In the event that a dodge takes place, it would simply not apply any damage to their strength and then return.

Test Plan

My test plan will be evaluating that specific parts of the implementation are taking place, sometimes ignoring certain rules in order to make the testing a little bit easier.

1. Our first step of testing is normal combat simulation and ensuring that combat terminates once one of the creatures no longer has any strength left. This will be done between the Barbarian and ReptilePeople, since the combat should last fewer round and be easier to test. It is most likely the Barbarian would lose most of the time, since they lack armor.
2. Our second step of testing would be to make sure that the Achilles ability is being triggered. For this we will let combat continue (ignoring strength dropping below 9) until we have the Achilles applied to one of the other creatures. This will also be done against The Shadow since the amount of rounds for getting an Achilles will usually be enough to see a dodge being demonstrated.
3. Our third step in testing would be to have two goblins fight each other to ensure that we don't have an Achilles being triggered. For this, we will have 12 rounds of combat (ignoring strength) and see if a twelve was rolled by either and that no Achilles was triggered.
4. Our final step of testing would be to just make sure that Blue Men is working properly, which we will do against another Barbarian. This is just to ensure that we have a correct implementation of the class.

Test Results

1. This was straightforward and had combat consistently ending upon one of the creatures going to or below zero strength points.
2. This had elicited quite a bit of my code, since I sometimes had an Achilles being triggered when no 12 was rolled, which clued in that I was calling my attack() method too many times, so I was getting silent 12's rolled. I was able to correct this and had each phase of testing terminate once an Achilles was activated

3. This took place after correcting the errors from 2 and prevented any complications with the results of testing. The two would combat and roll 12's and never have an Achilles be triggered
4. This was exactly as number 1, and no odd behavior took place in these results.

Reflections

Being one of the more involved assignments, this has me learning a new level of organization. I had never been more thankful for using some version management to save my progress at intervals where I had my program in a working state. The most difficult portion for me was getting values to change correctly and consistently, which ultimately led me to using pointers to my creatures so that I could be sure that their strength values we actually being changed.

I ended up having to make quite a few changes from my original design in order to make things work. Although I kept the attack, defense, and takeDamage methods, I had added a few methods that were used for returning values. I had added a data member that would initialize the creatures type, so that it would be possible to know what type a creature was (which was useful for a goblin v goblin situation).

I also added the methods for returning strength and armor for each character so that it would be possible to know how much strength was left as well as properly evaluate how much damage should have been done to a creature.

I decided to have my Combat class simply handle all combat related issues, and even handle special cases of The Shadow and Goblins by announcing when they had used one of their specials. This allows me to have all of my test cases to be pushed into Main by just have a couple while conditional or for loops to call the battlePhase method to make combat take place.

Ultimately, this was quite eye opening as to what is possible with polymorphism and why inheriting your classes correctly is so important. You can then just pass in a super class and handle way more data types as a result of it.