**Comp 1502 Assignment 4 - Troll Hunter Overview**

**Requirements**

* The program will ask the user to enter a file for the game until the user selects a file that is valid
* If a file is not completely valid, then the program will not except it (this required exception handling)
* The users movement has to be limited (so that they can only move to caves and not out of bounds)
* The user and trolls must be able to fight and kill each other/die
* The program must end when the user wins, dies, or chooses to exit the game
* Use polymorphism and inheritance in the program’s design
* Use J-Unit to test sections of the program
* The user has to be given only certain buttons to use when certain movements or actions would not be valid

**Design**

My troll hunter program was primarily focussed on using polymorphism and inheritance to create the weapons, characters, and caves. The weapons are objects that have an abstract parent class called “weapon” that all of the weapon objects are inherited from (into separate classes for each weapon type in the game). The different classes used different constructors from the parent (fist, dagger, sword, axe, and the kill all sword have different parameters). The methods for calculating the damage the weapon could do were over-rid in different ways because each weapon has its damage method calculated in different ways. Character objects were all derived from an abstract class called character. The character object that the user plays as was inherited directly off of the parent character class. This “hero” class holds a weapon object inside of the hero object that the user/hero can use when they fight troll objects. The troll objects were inherited from a troll class which was inherited from the characters parent class. This was done because trolls have different parameters than the hero object does, so it was more logical to me to make a troll class to derive the three specific troll classes off of. The three specific troll classes each were inherited form the troll class and simply over-rid the methods that were used to determine if a troll would attack or defend. This was done because the different trolls attack in different ways, so I was able to use polymorphism to effectively make the trolls attack in different ways. All of my objects tie in with each other because they all interact together. Weapon objects are stored in character objects, which are stored in cave objects. These objects all feed off of each other by passing information between them as the game is played.

**Tests**

I tested all of my weapon objects to ensure that they inflicted the correct amount of damage by using J-Unit automated tests.

To test my battles between a troll object and a user object I set up the “battleTestClass” class to make sure that damage could be inflicted by the weapons from both the user and the trolls when they fight each other. This testing was not automated, but was simply done by running through the class so that I could tell that the fighting was working correctly.

I also set up an automated test for certain parts of the battle sequences with J-Unit because I felt that my battleTestClass class was not a legitimate way to test my battle sequences, it was more of a way for me to make sure that I knew that everything was working with the battles in the ways that I wanted it to (when I used Blu J’s debugger to look through all of the objects in the class).

The first test I made for battles was a test to see if both the hero/user could inflict the correct amount of damage onto each other when they would both attack at the same time.

The second test for battles tested to ensure that no damage would be inflicted upon either character object when they were in a battle sequence together but both chose to defend.

The third battle test was testing to see if the hero/user could actually kill a troll. I gave the hero a kill all sword object so that it would be easy to see if the hero was able to slay the troll in one attack even though the troll was defending.

The last test for battles was testing to see if the troll could kill the user when they both attacked. The troll was given a kill all sword object so that I could see if the troll could kill the user in one big hit.

I also tested the file reading input manually many times. To do this I selected files that were not text files, and I selected text files that contained incomplete troll hunter information.

**Final Thoughts/What I Learned**

Over the course of this assignment I learned how important it is to make code that does not duplicate itself. Instead of copying your code that is extremely similar to work in other sections of your program, it is better to make generalized methods that can be used in multiple sections of your program so that the amount of code you have is decreased , easier to debug, and easier to maintain. I also learnt how to interact with a graphical interface in Java, though I do not feel like I could easily create and use a GUI in another Java program because I do not feel completely comfortable with them yet. I also learned how important it is to design your code correctly the first time you make it and write it. Another thing I learned was that it is also very important to test all of the aspects of your program as you make them so that when you need to use parts of of your program that rely on other parts, your program can actually work without having a million bugs. It is best to work on the features of your program one step at a time so that you can have your program work the way that you want it to.