Braeden Kuether
August 6, 2018

Project 3 Reflection Document

Design -

This design looks similar to the previous project in that we will be using one main class, that the rest of the classes will inherit. In this case, instead of a zoo it's a fantasy fighting game. So, for this project we will have variable in a common character class, like strength, armor, attack, defense, lives in some cases and more. Moreover, we will have to use some specific functions to classes like the hogwarts function for Harry Potter, or the mob function for the Blue Men. In those cases, I will declare a virtual function in the overall Character class, and then implement each function separately in their classes. Same thing goes for the attack rolling function as well as the defense rolling function. However, in the case of simple getter and setter functions, those will be implement into the overall Character class and .cpp file. On of the tricky things I see going into the project is balancing how the hogwarts and charm functions will work with the glare function. I suspect that in each case I will have to alter the code slightly to get the results I'm looking for.

Specifics -

For the attack roll functions, I will implement each separately in their own classes, and use the rand function to return a random value of "dice". I will implement a similar system for defense roll functions. In special cases like the mob function, I will check for the strength of the Blue Men at the beginning of every turn, and alter the "defense" number depending on the strength. Based on that, the defense roll function will have different rand statements depending on the "defense" variable specific to the Blue Men. In the case of the Hogwarts function, I will set a "life" variable specific to Harry Potter. When the hogwarts function is called, it will check to make sure that Harry Potter is still on his first life, and then increment the life variable inside the function so it cannot be re called.

Test Tables:

Function	Test Value	Expected Outcome	Observed Outcome
input_validation()	a	This is expected to reprompt because it is not an integer.	Function reprompts.
input_validation()	1	This should return true.	Returns true.
input _validation()	1.0	This is expected to reprompt because the check function will find the period, which is	Function reprompts.

		not a 0-9 integer.	
input_validation	20	This should reprompt the user.	Function reprompts.
hogwarts()	Strength = 0 Life = 1	This should print a statement and increase life and strength to 20.	Prints statement, increments life value, increases strength to 20.
hogwarts()	Strength = 5 Life = 1	The function will simply return.	Does nothing.
hogwarts()	Strength = 0 Life = 2	The function will simply return.	Does nothing, because life value out of bounds.
charm()	Rand var = 0	This input will simply return, default to damage calculation.	Does, nothing. Calculates damage.
charm()	Rand var = 1	Will print a statement, and “block” all damage.	Prints a statement, damage for the turn is set equal to 0.
glare(int)	Input = 8	This will simply return	Function returns, calculates normal damage.
glare(int)	Input = 12	This will print a statement, and then set the opponent’s strength to 0.	This will print a statement, and then set the opponent’s strength to 0.
mob()	Strength = 12	This should not alter the defense roll function.	Nothing changes with defense roll function.
mob()	Strength = 7	Defense roll maxed out at 2 dice with 6 sides instead of 3.	Defense roll produces numbers in the 0-12 range only.
mob()	Strength = 3	Defense roll maxed out at 1 dice with 6 sides instead of 3	Defense roll produces numbers in the 0-6 range only.

Reflection: This project was a bit easier than the previous project given how much more experience I have with classes in general, and with things like separating files and using include statements. Some of the problems I encountered throughout my program were things like memory leaks, not compiling, and the specialty function not working properly. In the case of the memory leaks, I added delete statements right before the function returns, meaning that the objects I create at the beginning of functions were deleted. Next, In the case of my program not compiling, I had to go through all of the individual files and check the header guards, that there were semicolons after classes in the .hpp files and that I had all of the correct include

statements. Some of the specialty functions like the hogwarts were not working properly until I implemented the life variable to make sure that it could only be called once.

Class Hierarchy:

Character:
Variables like strength, defense, attack, lives, ect.

Inheriting Character:

Vampire:
Specialty function charm()

Medusa:
Specialty function glare()

Blue Men:
Specialty function mob()

Barbarian

Harry Potter:
Specialty function hogwarts()