Cody Hannan
CS162

PROJECT 3 - REFLECTIONS

DESIGN DESCRIPTION

Purpose:

The program we are tasked with making is simple fantasy combat game. The basic function of it is to prompt the user to choose between six different fighters for fighter one and two, who then fight and the winner is displayed to the user.

Requirements:

- A creature base class must be created as well as a subclass for each character. The creature class will be an abstract class.
- Each creature will have characteristics for attack, defense, armor, and strength as shown in the table provided in the posted Project 3 document.
- To resolve an attack, two dice rolls will need to be generated. The attacker will roll some amount and the defender will roll some amount. The final damage value will subtract the defense roll and the armor rating of the defender from the attack roll.
- Vampire will have charm ability that gives it a 50% chance to not be attacked.
- Blue men will lose defense die with every 4 points of damage to their strength.
- Medusa will use glare if a value of 12 is rolled. Glare is an instant kill.
- Harry Potter will be revived with 20 strength if his health is brought down to zero, but this can only happen once.

Design:

Fight Class:

Contains the functions for the where the fight occurs. The constructor will initialize the two fighters and then carry out the battle by calling the attack, defence, and get armor functions in order to calculate damage. The damage will be subtracted from a health variable, which will have been initialized using the getter function for strength.

Creature Class:

The abstract base class. It will contain pure virtual functions for attack and defense, which will be defined in each subclass. This class will also contain protected variables for armor and strength, which will be constructed in the subclasses. There will be getter functions to call the armor and strength values.

Creature Subclasses:

The menu function will display the menu options available to the user.

Menu function::

The menu function will display the menu options available to the user.

Input Validation function:

The inputValidation function will validate user inputs by checking that entered values are in the correct range and of the correct type.

Main function:

The main function will primarily be where the user selects menu options and selects the fighters that will battle.

Pseudocode:

Fight Class:

Fight class constructor that takes user input for which fighter will battle

If fighter1 is Vampire

Allocates memory for a vampire object and assigns it to ptr1

.

.

.

If fighter2 is Vampire

Allocates memory for vampire object and assigns it to ptr2

.

.

.

There are if statements for each class.

Fight class winner function that executes the battle and determines the winner

Get fighter1 health

Get fighter2 health

While fighter 1 and 2 health is more than 0

Randomly choose the target

Output who is attacking to user

Damage=fighter attack - opposing fighter defense and armor

If damage is less than 0

damage=0

Damage is deducted from fighters health

Protected variable of corresponding object is changed

Output value to new value to user

If health is less than 0

Return other fighter

Creature Class:

class Creature

{

Protected variables for armor and strength

Public:

Constructor takes values to init armor and strength

Pure virtual functions for attack and defense

Getter functions for armor and strength

Setter function for strength

Menu function:

Outputs menu options available to user:

1. Battle.
2. Exit program.

Input Validation function template:

Function call template is inputValid(int min, int max, string “variable”)

If variable == “some type of input that is needed”

Cin >> User inputted value

while(cin fails or input is outside of max and min range)

If cin failed

Notifies user to enter the proper value type.

Cin >> user inputted value

Else if value is outside of range

Notifies user to enter a value within max and min range

Cin >> user inputted value

If variable == “some other type of input that is needed”

.

.

.

Main function:

While option doesn’t equal 2

Menu function is called

User is prompted for selection

If selects 1

Ask user to choose their fighters

List the available fighters

User is prompted for fighter one and fighter two

Fight object is created

Outputs winner to the user

TEST PLAN / RESULTS

Case	Input	Function	Expected Outcome	Actual Outcome
Vampire vs other fighters	User selects option 1 and chooses fighters	fight.winner()	Outputs Fight progression and winner. Charm should occasionally activate	Outputs Fight progression and winner. Charm occasionally activated
Barbarian vs other fighters	User selects option 1 and chooses fighters	fight.winner()	Outputs Fight progression and winner	Outputs fight progression and winner
BlueMen vs other fighters	User selects option 1 and chooses fighters	fight.winner()	Outputs Fight progression and winner. Defense should lower as strength decreases.	Outputs Fight progression and winner. Defense lowered as strength decreased.
Medusa vs other fighters	User selects option 1 and chooses fighters	fight.winner()	Outputs Fight progression and winner. Glare should occasionally activate	Outputs Fight progression and winner. Glare occasionally activated.
HarryPotter vs other fighters	User selects option 1 and chooses fighters	fight.winner()	Outputs Fight progression and winner. Should revive when health reaches 0	Outputs Fight progression and winner. Didn't revive when health reached 0
Fighters vs themselves	User selects option 1 and chooses fighters	fight.winner()	Should work the same. Fighters are differentiated by fighter one: and fighter two:	Worked the same. Fighters are differentiated by fighter one: and fighter two:

REFLECTIONS

This assignment wasn't too bad. The main issue was getting the child classes to all interact with each other properly. My initial problems came from figuring out how to keep track of health for the fighters while they battled. At first, I thought I'd use a friend function so that the fight class would have access to the protected variables strength and armor in the creature class, but I was unable to get idea working properly so I abandoned it. I instead opted for using getter functions to store the health in a separate variable within the fight class. This worked pretty well until I got to the abilities for Harry Potter and the Blue Men. Their abilities have to do with their strength values and since I wasn't able to reference those values directly I had some problems. I was able to get the Blue men ability to work by using a setter value for strength. As health changed I would use the setter function to change the corresponding creature classes protected strength variable. However, with Harry's ability it didn't work as well. I think it has moreso to do with the while loop and at what order events within the battle occur. Overall, I felt better about this assignment. Probably because it didn't have as much to do with memory allocation, which has proven to be a weakness of mine.