Design

1. Character Class (Abstract)
    a. Protected Variables
        i. Attack
        ii. Defense
        iii. Armor
        iv. Strength points
    b. Protected functions (virtual)
        i. Attack
            1. Override for each subclass
                a. Return damage pts rolled (attacked)
        ii. Defense
            1. Override for each subclass
                a. Takes damage from attacker, calculates damage inflicted, subtract from defender strength points
                b. Damage = attacker's roll – defender's roll – defender's armor
2. Vampire subclass
    a. Attack
        i. One rand() from 1-12
    b. Defense
        i. One rand() from 1-6
    c. Armor
        i. 1
    d. Strength points
        i. 18
    e. Special
        i. Charm
            1. 50% chance opponent doesn't attack
                a. Rand() from 1-2
                    i. If 1 proceed w/ attack
                    ii. Else if 2 skip opponent attack
3. Barbarian subclass
    a. Attack
        i. Two rand() from 1-6
    b. Defense
        i. Two rand() from 1-6
    c. Armor
        i. 0
    d. Strength points
        i. 12
    e. Special
        i. None
4. Blue men subclass
    a. Attack

i. Two rand() from 1-10
b. Defense
i. Three rand() from 1-6
c. Armor
i. 3
d. Strength points
i. 12
e. Special
i. Mob
1. Adjust Defense based on strength points
a. If strength is > 8, defense is 3d6
b. Else if strength is > 4 && strength < 8, defense is 2d6
c. Else if strength is < 4, defense is 1d6

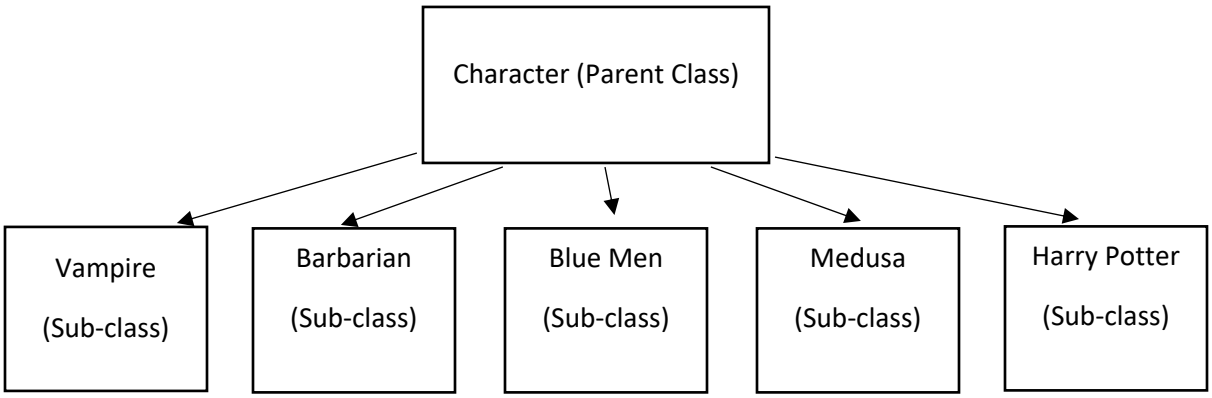
5. Medusa subclass
a. Attack
i. Two rand() from 1-6
b. Defense
i. One rand() from 1-6
c. Armor
i. 3
d. Strength points
i. 8
e. Special
i. Glare
1. If attack rand() == 12
a. Damage is == Enemy strength points
i. Enemy health is taken to zero

6. Harry Potter
a. Attack
i. Two rand() from 1-6
b. Defense
ii. Two rand() from 1-6
c. Armor
iii. 0
d. Strength points
iv. 10
e. LifeCounter
a. Set to 1, decremented if Hogwarts is used
f. Special
v. Hogwarts
1. If strength points hit 0 and LifeCounter != 0;
a. Strength == 20
b. LifeCounter = 0

Game class

1. Ask user to choose which characters to battle
    a. Have pointer for Character A + Character B
        i. When choice is made, create new character and set pointer to it
2. Rounds loop
    a. If one character's strength points < 0, exit loop, end battle
        i. Else keep looping
    b. Display Function (inside rounds loop)
        i. Call get functions for each character
            1. Character type
            2. Armor pts
            3. Strength pts
            4. Attack roll value
            5. Defense roll value
            6. Total damage
            7. Defenders strength points after damage
3. After battle ends ask user if they want to keep playing

Class Hierarchy



Test Table

Test Case	Input	Expected	Observed
User enters option to play again or quit	Enters "1" or "2"	If "1" plays again If "2" exits program	"1" played again "2" exited program
User enters invalid character selection	Enters value != int, or not within range	Tells user entry is invalid, tries again	Tells user entry is invalid, tries again

Character stats are appropriately displayed	Play round	Health displayed before is changed after damage taken	Health displayed before was changed after damage taken
Charm ability is appropriately executed	Play vampire class	When charm is cast, opponent doesn't attack	When charm was cast, opponent didn't attack
Glare ability is appropriately executed	Play Medusa class	When glare is activated, damage done automatically kills opponent	When glare was activated, damage done killed opponent instantly
If player dies after first attack, round ends	Play characters until first attack kills opponent	After attacker kills opponent, round ends, doesn't get turn	After attacker killed opponent, round ended, didn't get turn
If Harry Potter dies, comes back once, with 20 strength	Play Harry Potter class	Harry Potter comes back once hp is zero	Harry Potter came back ONCE with 20 hp
If Blue men strength is diminished, defense goes down	Play Blue men class	Blue men class, roll smaller numbers below 8 and 4 strength pts	Blue men class, rolled smaller numbers below 8 and 4 strength pts

Reflection

My design plan initially felt very bare-bones, but there was a lot I couldn't account for until I saw how the characters interacted with each other. The abstract class was initially missing get functions and a variable for type. I didn't realize the importance until I began defining the display function and needed a way to display the values of each character without accessing them directly. Once I was able to display the stats I was better able to see the outcome of the battles between characters. It was then that I encountered a bug that increased the hp of characters on some attacks. I figured out that after calculating damage taken, if the armor or defense was higher than the attack, it would subtract a negative number from the strength points, essentially adding it. I decided to add for loops for each character to prevent negative hp and negative damage taken values.

One of the biggest hurdles I came across was figuring out how to appropriately output when charm occurs. I originally set it to set damage taken to zero if triggered, and the numbers came out fine, but the program was still printing that the opponent attacked. I decided I needed to create a bool variable to flag when charm was activated. Then I needed a get function to get the value of that bool. The problem was that not all characters had charm abilities, and I needed the program to only get it when the function was accessed by a vampire object. I decided that I needed a virtual function, but one that didn't need to be overridden by the child classes that don't need it, and for them to still become concrete. So instead of creating a pure virtual function, I created a virtual function and defined it to always return a bool set to false. If called by any class other than vampire, it would return false for the charm bool. This virtual function would then be overloaded by the vampire class to modify the charm bool as needed.