Justin Hammel
CS-162
Spring 2017
Project 3 Design and Reflection

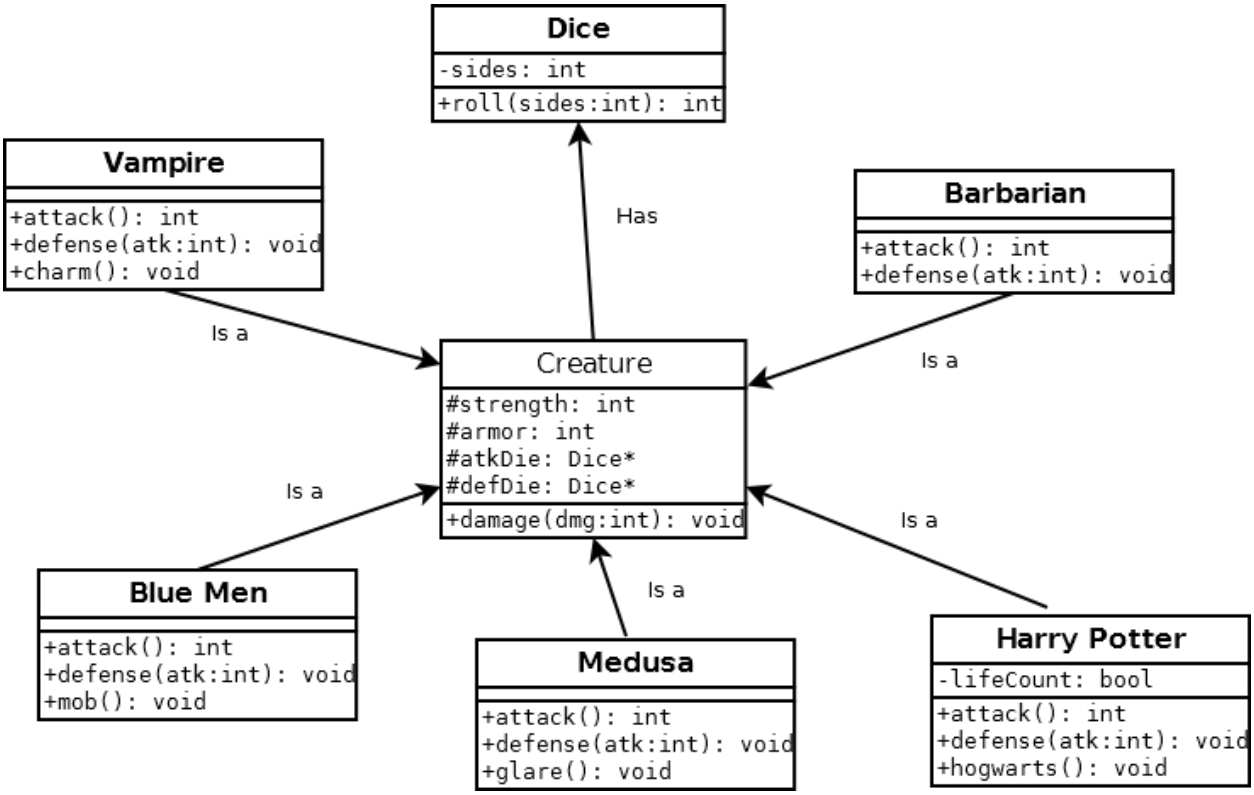
1. Define what the program should do.
    a. Purpose: Create the class hierarchy for a fantasy combat game. This portion of the project involves implementing the characters for the game. There are 5 derived types of characters that will be derived from a creature base class. The program will allow the distinct types of creatures to fight each other based on rules particular to their specific creature class.
    b. Input:
        i. There will initially be a main menu requiring input based on the types of creature available to fight:
            1. The user will have to select two of the five available creatures to fight in the program: Vampire, Barbarian, Blue Men, Medusa, or Harry Potter
            2. Input will be based on the user selecting the integer printed next to the creature names when the menu is displayed
    c. Processing:
        i. Random function will need to select which of the two creatures will attack first
        ii. Random rolls for each of the different dice type need to be generated: 12 sided, six sided, and 10 sided
        iii. Attack function passes damage based on die roll to defender's defense function
        Defenders die roll subtracted from the attack roll
        Remainder will be the damage inflicted which will finally have the defenders armor value subtracted resulting in the damage dealt to the defender
        iv. Strength points for each creature are monitored with the damage from attacks being subtracted each round. When a creature reaches 0 strength they are defeated.
        v. Within each class with special powers there will need to be a checker to control the special powers of that class
            1. Vampire – 50% of attacks do not apply due to charm power
            2. Blue Men – Adjust defense of creature for every 4 damage taken (rounded down) lose 1 die of defense
            3. Medusa – when attack is a 12 medusa automatically wins combat. Harry Potter ability still applies. Vampires charm can negate glare.
            4. Harry Potter can resurrect once if health reaches 0. When this happens his new strength becomes 20. After second loss he is dead.

   d. Output:
      i. Initially a menu will output with the name of the creatures for the user to select to fight
      ii. The results of each round of fighting will be displayed on the screen
         1. This will include round number, attack, defense, armor, final strength, and if any specials were used
         2. At the end of the fight display the winning creature to the screen and a prompt to play again, or exit

2. Program design
   a. Menu creation and display will be handled by my menu class
      i. Update menu options text document to reflect creature choices
      ii. Integer values will be displayed next to creature names for the choices
   b. Prompt will display with the menu offering the user the option of selecting two creatures to fight or exiting the game
      i. Program will loop until two creatures have been chosen, or the option to exit is chosen
      ii. Creature counter will determine when two creatures have been generated for combat
   c. After two creatures are chosen the program will randomly pick first attacker using rand()
   d. Die rolls will be handled by a set of functions based on the type of die needed
   e. Virtual function for attack will be called by the first creature to attack
      i. This will call the appropriate function for the type of die needed
      ii. Attack dice result will be passed to the virtual defense function of the defending creature
      iii. Defense dice will be rolled by the defending creature
      iv. Loss of strength is based on atk die result – (defense roll + armor)
      v. If creature reaches ≤ 0 strength the opposing creature wins
         1. First time harry potter reaches ≤ 0 strength his strength will be changed to 20. Following second drop to 0 or less he loses.
      vi. Powers for other creatures will be activated as previously mentioned in the design plan.

Justin Hammel
CS-162
Spring 2017
Project 3 Design and Reflection

3. Class Diagram



4. Testing Plan

TEST	INPUT VALUES	DRIVER FUNCTIONS	EXPECTED RESULTS	OBSERVED OUTCOMES
CREATURE CREATION	User selects two barbarians to fight, this will be done prior to generating any other class	Creature and Barbarian class constructors and virtual fight function	Creatures should fight according to the rules of the game	Game plays as expected. Correct values are displayed for attack, defense, damage, and new strength values.
VAMPIRE CHARM	Barbarian fights Vampire	Vampire charm power test	Charmed attacks from the1 Barbarian should not succeed	Charm power causes damage for attacker to be set to 0.
BLUE MEN SWARM	Barbarian fights Blue Men	Adjust defense based on strength	1 die of defense removed per 4 damage taken (rounded down)	Had to change the test to Blue Men fighting another Blue Men to even the odds. Initially Blue Men mob ability was not functioning properly. Dice were not being removed after damage was dealt. Changes will be discussed in the reflection portion.
MEDUSA GAZE	1. Medusa fights Barbarian	Medusa gaze power	If the result of Medusa attack is a 12 should auto win combat	Medusa gaze power works as expected against Barbarian.

MEDUSA GAZE CONT.	2. Medusa fights Vampire		against Barbarian. Against Vampire charm ability should negate gaze.	After tweaking the code to make Medusa perform gaze more often Vampire charm power trumps gaze as expected.
HARRY POTTER POWER	Harry Potter fights Medusa	Harry Potter Hogwarts	after reaching 0 strength or being affected by Medusa gaze the first time Harry Potter should have strength adjusted to 20. If he loses a second time then the game should end.	Used same Medusa tweak as above to make gaze power more frequent to test against Harry Potter power. Harry Potter's power functions as expected against the Medusa Gaze.

5. Reflection

I was rather pleased with how this project turned out. I think I have finally figured out how to get rid of my memory leaks and it only took me 6 weeks. The biggest thing I found out this week regarding memory management was that using one delete on two different pointers in one line (e.g. delete ptr1, ptr2) does not work. On top of that, it also leads to weird memory leaks as it only affected half of my pointers. It took me a while to track down the problem, but I finally figured it out.

The size of this program meant that I had many changes that needed to be made throughout the implementation and testing process. One of the first major changes I had to make involved creating a separate class to handle the actual combat. This made since to help take out a substantial portion of code that would have otherwise cluttered up my main function.

Originally after I first decided to implement the Fight class I was planning on passing the individual Creature pointers into the combat method. This would have made things difficult when trying to keep tabs of whose turn it was. To fix this problem I decided to go with an array of pointers to the creatures instead. Since I knew how many creatures would be in combat at a time using this scheme was very easy to implement.

Originally when trying to deal with either the Medusa gaze power, or the Vampire charm power I was not very clear on how I would implement them. After thinking about it I opted to use a method that either boosted the damage when the gaze power was activated, or in the case of the charm power set the attackers damage to zero. For the purposes of this project I think this works well, but there could be issues with the Medusa power if the damage is not sufficient to win combat.

The most difficult power for me to get working was the mob ability for the Blue Men. At first I thought to just have the defense dice removed by checking the strength of the Blue Men creature, but this did not work as I expected. The way my turn counter was functioning when the mob ability would adjust the number of dice available 1 turn after the damage was dealt. In the end, I went with a counter method to tally up the damage dealt to the defending creature. This value then ended up controlling how many defense dice the creature would have to roll with next turn.