Justin Hammel
CS-162
Spring 2017
Project 4 Design and Reflection

1. Define what the program should do.
    a. Purpose: Create a program that will utilize the creature classes created in project 3 to compete in a tournament.
        i. Linked lists and queues will be used to store the rosters of fighters for the user and cpu teams.
        ii. There will be a linked list to store the pointers for the creatures that loses fights.
        iii. Following the tournament a winner will be determined based on the point values assigned to the creatures each player defeated. A set number of points will be assigned to each creature that will be tallied up at the end of the fight (when one player has no creatures left).

    b. Input:
        i. Human player will have to input the size for the teams to fight, size_t teamSize
        ii. Player will then have to select the creatures to be placed on the teams from the available menu choices, Creature* creatChoice
            1. Human player inputs team lineup first
            2. Multiples are allowed
            3. After creature is selected the player must provide a name for each of the creatures, string* creatName
        iii. After the match is over there will be an option to ask the user if they would like to play again

    c. Processing:
        i. Creature fight(), controls the flow of combat
        ii. Tournament score(), will keep a tally of the score for the current match
        iii. Queue Linked structure class is a FIFO linked structure that will control the creature lineup
            1. Creatures added by player starting at the head of the list moving to the back
            2. Creatures are moved to the back of the list following a win
        iv. A linked list structure will hold the loosing fighters
            1. Creatures for this list will be added to the front of the list

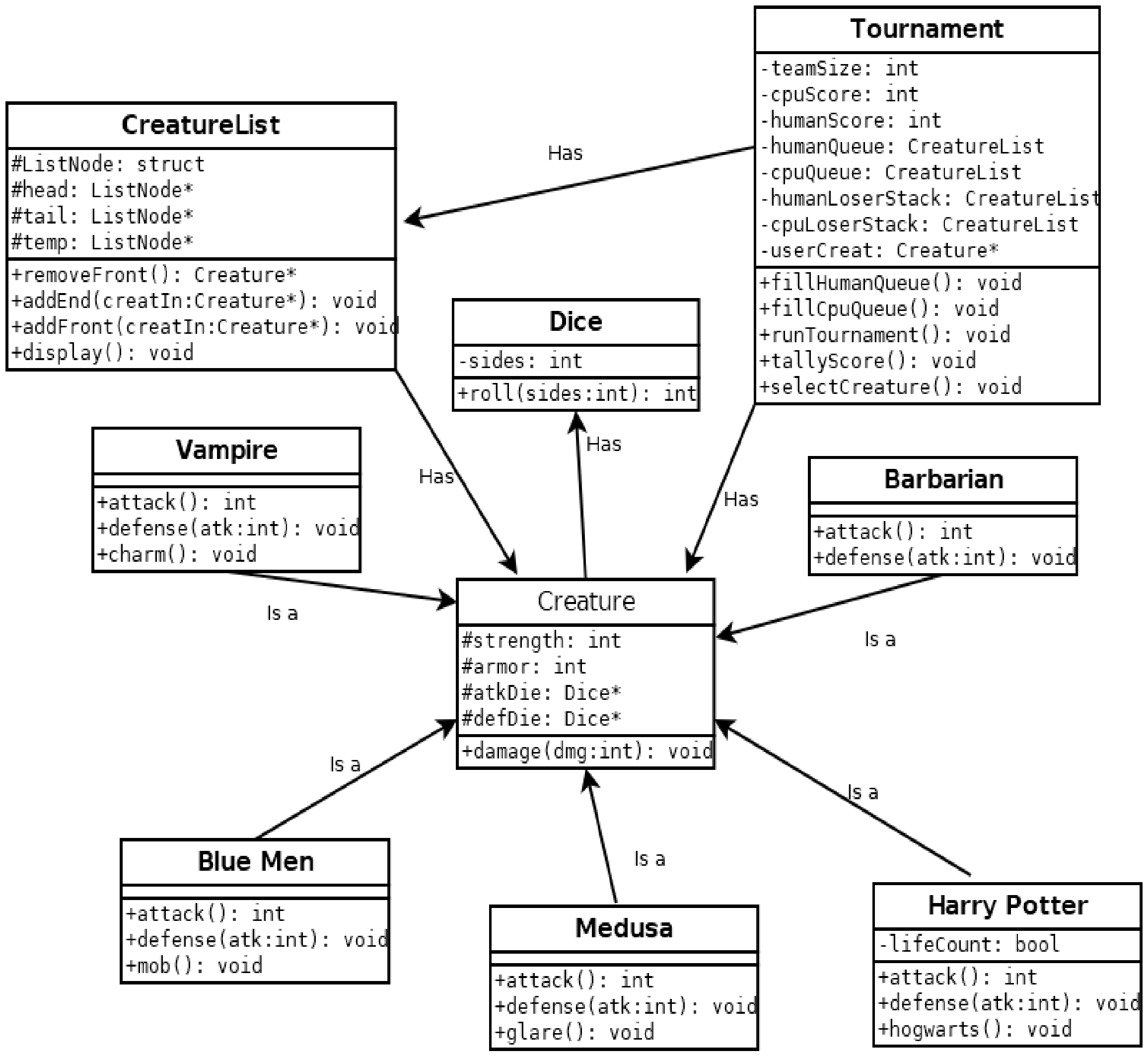
    d. Output:
        i. Output will prompt user for input when needed
            1. Team size, creature selection, creature name, would you like to play again?
        ii. Following each round of combat the results of the fight will be displayed
            1. Will show the types of creatures that fought and who won

iii. When the game is over (one player has no creatures remaining) the winner of the tournament will be displayed based on the highest points

2. Program design
    a. Main menu with the following options:
        i. Start game
        ii. Play again after tournament ends
        iii. Exit
    b. First prompt following start of game asks for team size, int
    c. Creature menu will display following team size input
        i. Creature menu will include just the creature names from project 3
        ii. After a creature is selected the user will be prompted to select the creatures name as a string
        iii. Creature* will be stored in a queue style doubly linked list structure
            1. There will be 2 queue style lists generated, plyCreatures, cpuCreatures
        iv. Following the lists being filled the program will traverse the lists from the front to back sending creatures to the fight function
            1. When a fighter loses they are placed into a linked list structure that adds new Creature* to the tail of the list
            2. Winning creatures are added to the end of their respective queues
            3. Results of the fights will display on the screen
            4. Game ends when one of the players lists become empty
                a. Head = null
            5. Following individual fights score will be tallied for the winning player
                a. This will be handled by Tournament score() method
                b. There will be a score value for each player within the Tournament class
            6. Following the end of the game the winner based on points will be displayed
            7. If menu option was made to play again after then the game will loop back to the first menu

Justin Hammel
CS-162
Spring 2017
Project 4 Design and Reflection

3. Class Diagram



4. Testing Plan

TEST	INPUT VALUES	DRIVER FUNCTIONS	EXPECTED RESULTS	OBSERVED OUTCOMES
ADDING TO QUEUE	Select 1 of each creature to add to each team with a team size of 5 chosen.	fillHumanQueue() fillCpuQueue() addend()	Run prior to fights being run. Will use display() to verify that the queue is building properly.	Ran into issues with my destructor in the Tournament class. Deleting of Creature pointers was causing the data in the linked list to be deleted as well. After getting rid of delete calls in the Tournament class Creature pointers were properly added to the queue.
FIGHTING BATTLE	One of each creature on each team of 5.	removeFront() tallyScore() addFront()	When creatures lose a fight they should be added to the loser stack for their respective team. Score should count two points for each creature defeated. Winnning fighters should return to their queue after	Once the battle portion was added more memory leaks started to appear. This will be discussed in further detail in the reflection portion.

			healing.	

5. Reflection

This program gave me more trouble when trying to track down and solve memory leaks. Originally I think I fell into the same trap that most other people did when trying to implement linked list with dynamically allocated objects. I was trying to manage the pointers and dynamically allocated memory in both the calling section of the program and the linked list.

At first I had calls to delete my creature pointers in the destructor for the Tournament class. This would occur when I was trying create new Creature objects and pass them as pointers to the linked list. Following passing them I would then delete the pointer I had just created. I found out this was wrong when I was unable to print any of the data for these creatures due to read access violations trying to access the creature pointers that I thought were still part of their linked list nodes. This led me remove all the deletes that were being called for my creature pointers in the Tournament which worked until I tried to implement the actual combat portion.

I tried to follow some of the advice to correct my memory leaks that were suggested in Piazza, but I was unsuccessful. There still seem to be quite a few leaks when the combat takes place. The fact that the numbers fluctuate as much as they do leads me to believe that the number of rounds that the game plays is causing based on how long it drags out. Most of my changes from when I started designing revolved around trying to isolate the memory leaks, but none of them seemed to succeed.

Following the advice of some of the other students I tried to make changes to my CreatureList destructor and the removeFront() method to fix some of the memory leaks. This involved making sure my removeFront() method was not deleting any of the creature pointers, but rather leaving them to the destructor to take care of in the end. This way the data for the creature would still be available to add back to either the queue or the stack when needed. Most likely I will have to take this matter to one of the group study hours and hope I can get further clarification.