Kimberly Tom

CS 162

Project 4

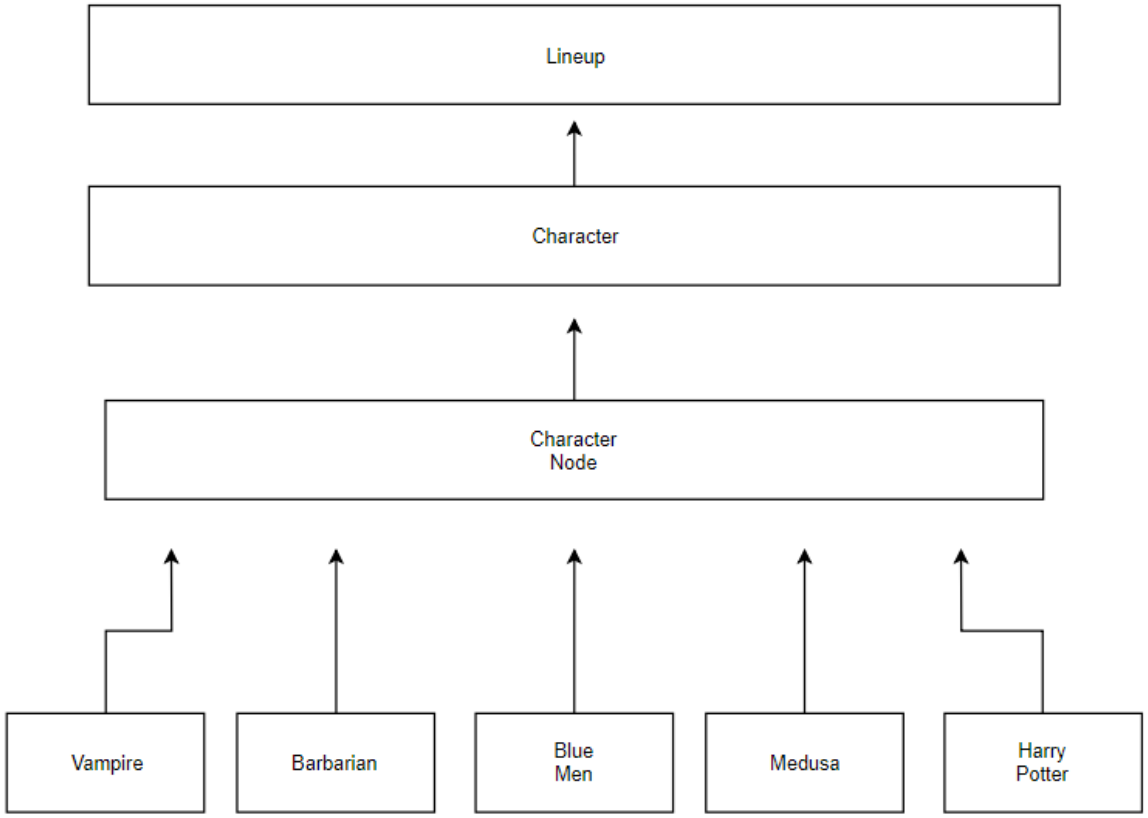
Design for Fantasy Combat Tournament

1. Main
    a. Main is where we instantiate the menu
        i. Key functions and data members
            1. hold menu function
            2. set random time
2. Character base class – abstract class
    a. attack function - virtual
    b. attack die roll function
    c. defense die roll function
    d. defense function - virtual
    e. armor function
    f. damage taken function
    g. strength points function - virtual
    h. recovery function
    i. return character type function - virtual
3. Vampire – derived class of character
    a. attack roll function uses 1d12
    b. defense roll function uses 1d6
        i. charm ability
    c. attack function takes dice roll and implements attack
    d. defense function takes dice roll
    e. armor function initialized at 1
    f. strength points initialized at 18
    g. recovery function returns half of total starting HP, or full recovery if goes above starting HP
    h. get character type function
4. Barbarian - derived class of character
    a. attack roll function uses 2d6
    b. defense roll function uses 2d6
    c. attack function takes dice roll and implements attack
    d. armor function initialized at 0
    e. strength points initialized at 12
    f. strength points initialized at 18
    g. recovery function returns half of total starting HP, or full recovery if goes above starting HP
5. Blue Men - derived class of character
    a. attack roll function uses 2d10

    b. defense roll function uses 3d6
    c. attack function takes dice roll and implements attack
    d. armor function initialized at 3
    e. strength points initialized at 12
        i. decrease defense die for every 4 points loss
    f. strength points initialized at 18
    g. recovery function returns half of total starting HP, or full recovery if goes above starting HP
6. Medusa - derived class of character
    a. attack roll function uses 2d6
        i. if rolls a 12, instant win
    b. defense roll function uses 1d6
    c. attack function takes dice roll and implements attack
    d. armor function initialized at 3
    e. strength points initialized at 8
    f. strength points initialized at 18
    g. recovery function returns half of total starting HP, or full recovery if goes above starting HP
7. Harry Potter - derived class of character
    a. attack roll function uses 2d6
    b. defense roll function uses2d6
    c. attack function takes dice roll and implements attack
    d. armor function initialized at0
    e. strength points initialized at 10, then 20 for second life
    f. strength points initialized at 18
    g. recovery function returns half of total starting HP, or full recovery if goes above starting HP
8. Menu
    a. Menu class displays options to the user
        i. select how many fighters on each team
        ii. function call to create the queue for each team based on how many fighters
        iii. function to create the loser list queue
        iv. function call to gameplay simulation
        v. option to play again after tournament ends
        vi. exit
9. Gameplay
    a. asks user what character to put on team and to name the character and adds to team queue for each team
    b. game function to play game
        i. character 1 attack function
            1. attack roll function
        ii. character 2 defense function
            1. defense roll function
        iii. character 2 attack function

            1. attack roll function
        iv. character 1 defense function
            1. defense roll function
    c. display results of which character won and lost
    d. restores health of character who won the round
    e. moves losing character to loser list, and winning character to back of team queue
    f. starts next round with the next character in line on each team and repeats until one team has no more characters with health left
10. characterNode
    a. takes in user's desired character choice and user's desired name for character
        i. creates new character object that takes in the nickname as parameter to be stored
11. lineup
    a. destructor sets head an tail to NULL
    b. checks to see if queue is empty (bool)
    c. destructor deletes the queue
    d. add new character
        i. add as only item in queue if empty
        ii. otherwise, adds to back
    e. get front of queue
        i. returns NULL if empty or returns the character at the head of queue
    f. remove front
        i. removes the first item in the queue, unless empty
    g. add to back of queue
        i. adds to the back of the queue, if empty, then this new item becomes the only item in queue
    h. add to front of queue
        i. if nothing in queue, it becomes only item in queue
        ii. if there is something in the queue, then adds to front of queue
    i. print list
        i. if not empty, this prints the queue from newest to oldest
12. Input Validation
    a. validation for start menu
    b. validation for fighter count
    c. validation for character selection
    d. validation for user's desired name of character
    e. validation for whether user wants to see loser list

Hierarchy



Test Plan and Results

TEST CASE	INPUT VALUES	EXPECTED OUTCOMES	OBSERVIED OUTCOMES
start menu, start	1	starts game	starts game
start menu. quit	2	quits game	quits game
start menu letter/number combinations	1a -2g b	invalid, prompt user for selection	invalid, prompt user for selection
enter number fighters team 1	1 7 10	valid, continue to ask for team 2 fighter amount	valid, continue to ask for team 2 fighter amount

enter number fighters team 1	-1 0 11 6d	invalid, ask user to enter valid integer between 1 and 10	invalid, ask user to enter valid integer between 1 and 10
enter number fighters team 2	1 5 10	valid, continue to character selection	valid, continue to character selection
enter number fighters team 2	-5 0 12 3s	invalid, ask user to enter valid integer between 1 and 10	invalid, ask user to enter valid integer between 1 and 10
character selection team 1	2, “Bobby” 4, “Tanya” 1, “6588” 4 “Tanya2”	valid, displays character selection	valid, displays character selection
character selection team 1	6 0 10 v 2d	invalid, ask for integer between 1 and 5	invalid, ask for integer between 1 and 5
character name selection for team 1	“ “ “” “1111111111111111 11111111111111”	invalid, ask for name length between 1 and 20	invalid, ask for name length between 1 and 20
character selection team 2	1 “Sarah “ 3 “4Fred” 1 “Ned” 1 “ned”	valid, displays character selection	valid, displays character selection
character selection team 2	7 f 8c 0 -3	invalid, ask for integer between 1 and 5	invalid, ask for integer between 1 and 5
character name selection for team 2	“ “ “” “222222222222222 222222222222221”	invalid, ask for name length between 1 and 20	invalid, ask for name length between 1 and 20
simulation with 2 chars on team 1 and 3 chars on team 2	team 1 1 “Fred” 1 “Johnny” team 2 1 “Anna” 2 “Sarah” 4 “Pat”	displays round results and final results and winner. combined score of both teams should be no more than 4, asks user if they want to see loser list	displays round results and final results and winner. combined score of both teams should be no more than 4, asks user if they want to see loser list
simulation with 1 chars on team 1 and 1 chars on team 2	team 1 5 “Harrrry” team 2	displays 1 round of results and final results and winner, asks user if	displays 1 round of results and final results and winner, asks user if

	4 “Med”	they want to see loser list	they want to see loser list
simulation with 10 chars on team 1 and 10 chars on team 2	team 1 1 “BB” 2 “Jill” 3 “Tony” 4 “Clay” 5 “ 222” 4 “B4” 3 “Nancy” 2 “Sara” 1 “Jim” 2 “Reed” team 2 5 “Kerry” 4 “Cory” 3 “33” 4 “Hii” 4 “brian” 5 “tara” 1 “Sam” 2 “Ned” 2 “Larry” 3 “laurie”	displays rounds and final results, displays winner, asks user if they want to see loser list	displays rounds and final results, displays winner, asks user if they want to see loser list
after final results, display loser list	1	show loser list, then asks if user wants to play again	show loser list, then asks if user wants to play again
after final results, don’t display loser list	2	asks if user wants to play again	asks if user wants to play again
after final results, when loser list is displayed	0 -1 3	invalid, ask user to select integer between 1 and 2 for selection	invalid, ask user to select integer between 1 and 2 for selection
after 1 simulation, play again, run another simulation	1	asks user to input how many characters for each team	asks user to input how many characters for each team
run 2 simulations	team 1 1 “Fred” 1 “Johnny” team 2 1 “Anna” 2 “Sarah” 4 “Pat” 2 don’t show loser list 1 play again	ones 1 simulation and shows final results, runs 2nd simulation and shows loser list of only the second round	ones 1 simulation and shows final results, runs 2nd simulation and shows loser list of only the second round

	team 1 5 “Harold” team 2 3 “Fiona” 1 show loser list 2 quit		
after simulation, quit	2	program quits	program quits
check for memory leaks	valgrind ./project4	no memory leaks	no memory leaks

Reflection

My program does not differ much from my design. I basically kept it very similar to the Project 3 design, but added characterNode struct and lineup class. I also added two more virtual functions to the character class, a function that returns the character type and a function that restores a character’s strengthpoints. One thing I did change was that I ended up creating a default constructor that takes in the two character counts together to be used to initialize the loser list queue.

A major problem I had was my program was exiting after the user enters in the name of their final character for team 2. I was calling a getFront function to obtain the first monster in each queue. My program was running the getFront function but it was exiting after that. I spent a lot of time thinking there must be something wrong with that function. I then ran valgrind and I was getting memory leak issues when that function was ran. I finally determined that it was because I was creating my list of characters for each team twice (2 times for each team), so I was running into problems. To solve this problem, I fixed my create team functions. I was already creating a team in gamePlay.hpp, so I was creating it twice. To solve this, I instead created a temporary pointer to hold the newly created

team, then I set that newly created team equal to the already created team 1 and team 2 pointers. This solved my problem and the program was not crashing at getFront anymore.

Another problem I had was I was confused how to have a winning character on a team gain it's strength back without going over their total strength points. At first, I just had the character recover fully, but then this wasn't really implementing an interesting function. I therefore decided that the character can gain half of whatever their full strength points are, and then use an if statement to say if their new total strengthpoints is above their maximum strength points, just set the strengthpoints to the maximum.

A third problem I had was with my lineup class. I had to keep looking through my functions in the lineup class to make sure there were no errors. It is really easy to mess up on the logic of queues and I had to keep checking to make sure things were done in the right order, or else the program would not work correctly.

This project, although builds off project 3, helped ingrained the idea of polymorphism's importance. I added two more virtual functions to the character class to meet the goals of project 4. I added a recovery function to recover a winning character's strength points and also added a get character type function so I can display the character type before the name of the character when printing information on the round and results. I learned how to have queues interact with each other. I also learned more about moving, adding, and removing items from queues and what types of situations they can be useful for.