Joel Huffman
2/25/2018
Project 4 Reflection

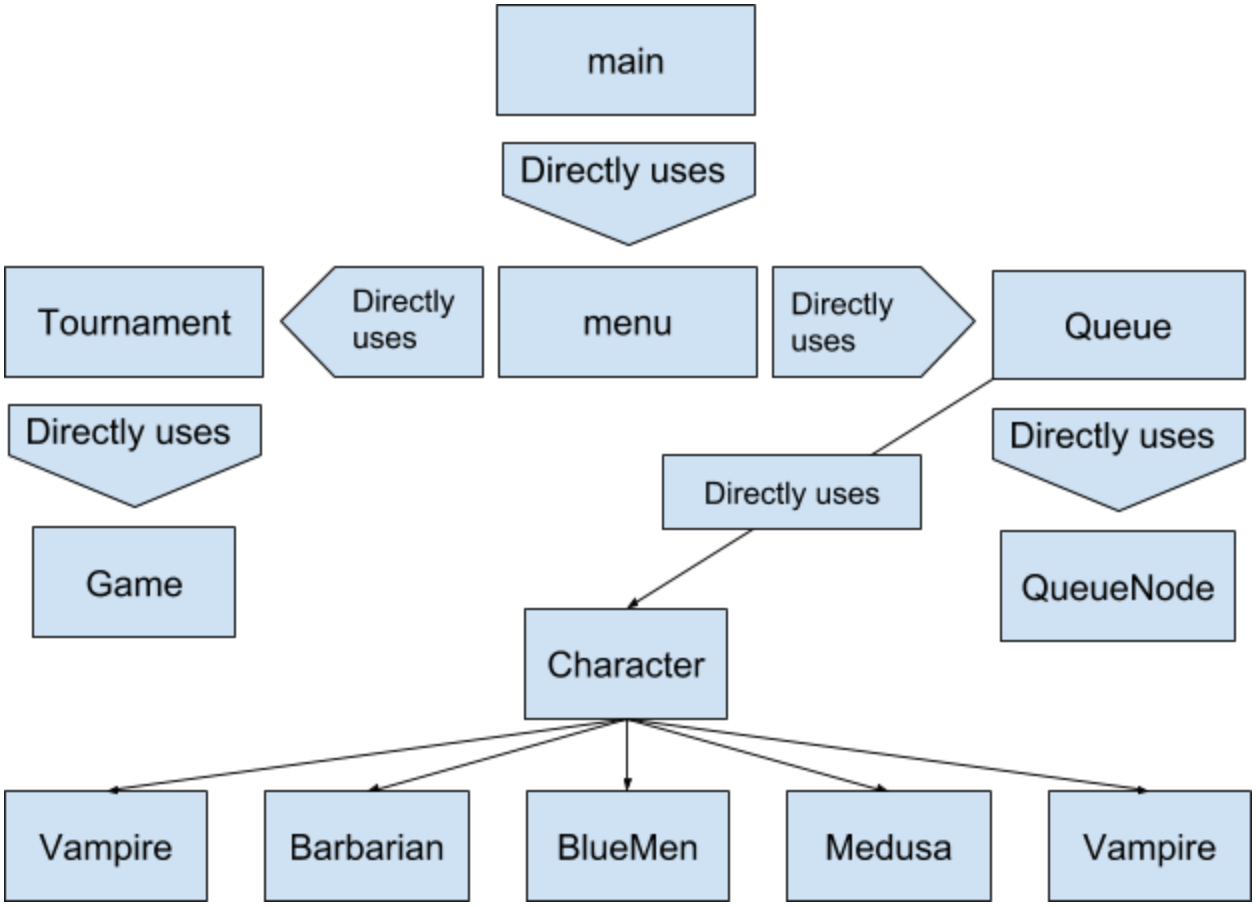
Design

Menu():
Display welcome prompt
Display choices for player:

1. Play
2. Exit
    a. If 1: Begin setup for tournament
        i. Create queue for each team
        ii. Create queue for losers
        iii. Get number of fighter for team from user
        iv. For each team:
            1. For each fighter:
                a. Get type of fighter from user
                b. Get custom name from user
                c. Create (specific) character object from this info
                d. Create character pointer to this object
                e. Add character pointer to team queue
            2. Run tournament with this above info
                a. While team1 != empty and team2 != empty
                    i. Run battle between first fighter in each queue
                    ii. Display results of battle
                    iii. Move loser fighter's pointer to losers queue
                    iv. Have winner fighter run recovery function
                    v. Move winner fighter's pointer to back of team queue
                    vi. Attribute point to winner fighter's team
                b. Display the final score for each team.
                c. Declare the winning team, or a tie
                d. Prompt user if they'd like to view the losers queue
                    i. If yes: for each pointer in loser queue
                        1. From last defeated to first defeated
                        2. Print fighter type
                        3. Print custom name
                        4. Print team
                    ii. If no: don't do anything
        v. Prompt user to play again:
            1. Play again.
            2. Exit
                a. If 1: Loop through and set up another tournament
                b. If 2: exit loop
    b. If 2: exit loop
        i. Delete all remaining objects on the heap

Joel Huffman
2/25/2018
Project 4 Reflection

Display exiting prompt
Exit program



Test Plans

Test Case	Input	Expected Result	Observed Result
Test low team size	1 1 Vamp 2 Barb 2 2	---------- Team 1 score: 1 Team 2 score: 0 Team 1 wins the tournament! ----------	---------- Team 1 score: 1 Team 2 score: 0 Team 1 wins the tournament! ----------
Loser stack printout	1 3 1 A1	Losers stack is: Type: Vampire, Name: a1, Team: team1 Type: Harry Potter, Name:	Losers stack is: Type: Vampire, Name: a1, Team: team1 Type: Harry Potter, Name:

	2 A2 3 A3 4 B1 5 B2 3 B3 1 2	b2, Team: team2 Type: Blue Men, Name: a3, Team: team1 Type: Barbarian, Name: a2, Team: team1 Type: Medusa, Name: b1, Team: team2	b2, Team: team2 Type: Blue Men, Name: a3, Team: team1 Type: Barbarian, Name: a2, Team: team1 Type: Medusa, Name: b1, Team: team2
Looping function for multiple tournaments	1 1 A 1 B 2 1 1 A2 1 B2 2 2	---------- Team 1 score: 1 Team 2 score: 0 Team 1 wins the tournament! ---------- Then ---------- Team 1 score: 1 Team 2 score: 0 Team 1 wins the tournament! ----------	---------- Team 1 score: 1 Team 2 score: 0 Team 1 wins the tournament! ---------- Then ---------- Team 1 score: 1 Team 2 score: 0 Team 1 wins the tournament! ----------
Recovery function working	1 1 A2 1 B2 2 2	-------- |FIGHT!| -------- Team1: Type: Vampire, Name: a2 Team2: Type: Vampire, Name: b2 Victor: Type: Vampire, Name: a2 Loser: Type: Vampire, Name: b2 6 strength recovered! Current strength now 12 b2 sent to loser's queue a2 sent to back of team queue	-------- |FIGHT!| -------- Team1: Type: Vampire, Name: a2 Team2: Type: Vampire, Name: b2 Victor: Type: Vampire, Name: a2 Loser: Type: Vampire, Name: b2 6 strength recovered! Current strength now 12 b2 sent to loser's queue a2 sent to back of team queue

Joel Huffman
2/25/2018
Project 4 Reflection

		---------- Team 1 score: 1 Team 2 score: 0 Team 1 wins the tournament! ----------	---------- Team 1 score: 1 Team 2 score: 0 Team 1 wins the tournament! ----------
Check for memory leaks	1 1 1 1 1 1 2 2	==28921== HEAP SUMMARY: ==28921== in use at exit: 0 bytes in 0 blocks ==28921== total heap usage: 35 allocs, 35 frees, 1,330 bytes allocated ==28921== ==28921== All heap blocks were freed -- no leaks are possible	==28921== HEAP SUMMARY: ==28921== in use at exit: 0 bytes in 0 blocks ==28921== total heap usage: 35 allocs, 35 frees, 1,330 bytes allocated ==28921== ==28921== All heap blocks were freed -- no leaks are possible

Reflection

My project design didn't change much throughout the coding process. I did originally plan to write new classes from scratch (for the Queue and Game classes), but I later decided to use the classes I wrote from previous projects and labs and tweak them to work in this setting. This provided me more time to work on other homework and reinforce the idea of creating modular, reusable code.

I was originally going to write a separate Stack class for use as the loser container, but I found it much easier to simply use the Queue class we already made in lab 7 and write a new print function for it that traversed the Queue from last to first to emulate a stack's FILO functionality.

I did run into some memory leak issues that took a bit of time to resolve. By walking through my program and noting any time I allocated or freed memory on the heap I was able to pinpoint what memory I wasn't freeing. Using valgrind also helped a lot with this problem.

This project and the last couple labs have really taught me the value of modular code. I certainly could rewrite container systems or menu functions or input validation tools, but by using existing code that I have already tested, debugged and I know

works I am able to save time coding. I should keep in mind that this is only the right choice if the piece of code requires only minor alterations. If the constraints vary too much from the original code's function, it might be less of a headache to just write fresh code rather than try to hack together something from a vaguely similar product.