CS_162_400_S2019
Project4
Susan Hibbert
ID# 933913975

Fantasy Combat Tournament

Design Description

I plan to implement the team lineup as a queue-like, singly linked list and the loser container as a stack-like, singly linked list, using a struct to represent the data structure of the linked list. For simplicity I decided to choose a singly linked list as I feel that a doubly linked or circular linked list are not necessary for this project and feel it would overcomplicate my design.

I plan to implement the lineup queue linked list before creating the loser stack, checking its functionality before implementing the stack linked list. I will ensure that I am able to successfully create two team lineups with the user-specified number of players and test whether the fighters at the front of each lineup fight each other and get put to the back of the lineup if they win. I will also check that the players are listed in the order in which they were entered by the user. My initial plan is to verify that the first two Characters from the two teams are able to fight successfully then if this works implement a while loop to get the whole team to fight.

Once this is functional I will write the recovery function of the Character class which will be called when the winning Character is placed at the back of its lineup queue.

Following this I will create the loser stack and check its functionality before moving onto the next part of the project. I will ensure that the losing Character is added to the top of the stack and that the stack prints out the Characters in the order of last defeated to first defeated.

In terms of the game play, I plan to have both teams be the same starting size, so the user will only be asked once for the team size. I feel this design decision is more in keeping with traditional tournament rules where both teams have the same number of players. With regards to the scoring system, I plan on keeping it simple. A win scores the respective team 1 point and a loss scores 0 points.

After I have written and compiled the various modules of my program, I plan to implement the test cases listed in the Test Cases table to detect any runtime issues with my code.

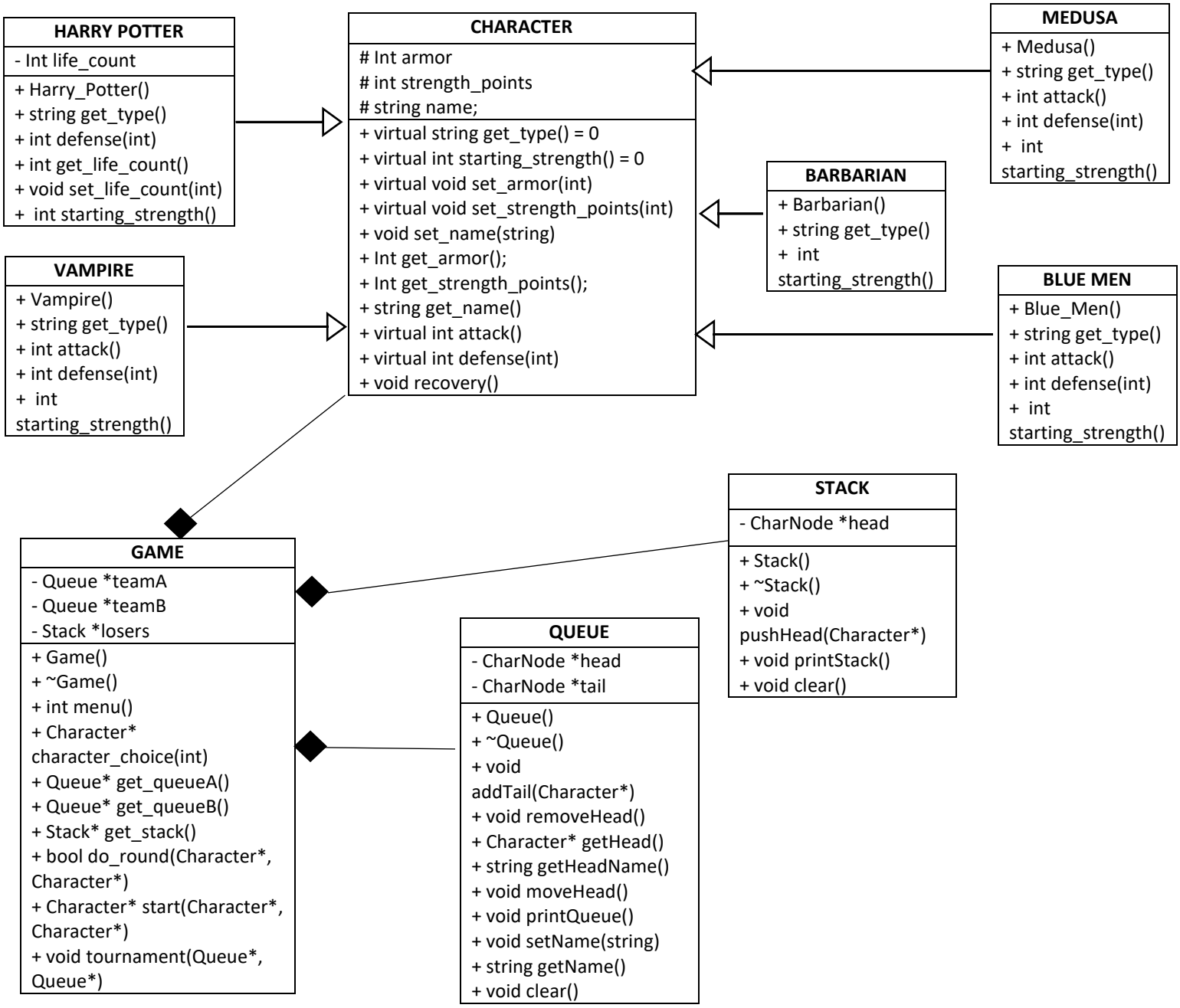
CS_162_400_S2019
Project4
Susan Hibbert
ID# 933913975

Class Hierarchy Diagram

= protected + = public - = private

= indicates inheritance; direction from child class to parent class

= indicates composition



CS_162_400_S2019
Project4
Susan Hibbert
ID# 933913975

Test Cases

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
START OF GAME				
User does not enter an integer when prompted to play game or exit program	Input is a not an integer. May be a character, float, or a mix of characters and integers including spaces. User may just have pressed the enter key	Int_input_val() function Do-while loop prompts user to enter an integer	User is prompted to enter an integer until an integer is entered by the user	User is prompted to re-enter an integer until integer is entered, as expected. Chars, floats and input containing spaces are rejected. If user presses enter, input is rejected
User enters an integer other than 1 or 2 when asked whether they would like to play game or exit	Input is an integer other than 1 or 2	Display_menu() function called by menu function of Game class While loop prompts user to enter 1 or 2	Loop back to the question prompting user to enter 1 or 2	Program loops back to the question prompting user to enter 1 or 2, as expected.
User does not enter an integer when prompted to select a character from the 5 integer options displayed on screen	Input is a not an integer. May be a character, float, or a mix of characters and integers including spaces. User may just have pressed the enter key	Int_input_val() function Do-while loop prompts user to enter an integer	User is prompted to enter an integer until an integer is entered by the user	User is prompted to re-enter an integer until integer is entered, as expected. Chars, floats and input containing spaces are rejected. If user presses enter, input is rejected
User enters an integer other than 1 – 5 when prompted to select a character from the 5 integer options displayed on screen	Input is an integer other than 1-5	Display_menu() function called by menu function of Game class While loop prompts user to enter an integer between 1 and 5	Loop back to the question prompting user to enter a number between 1 and 5	Program loops back to the question prompting user to enter an integer between 1 and 5, as expected.

CS_162_400_S2019
Project4
Susan Hibbert
ID# 933913975

User does not enter an integer for team size	Input is a not an integer. May be a character, float, or a mix of characters and integers including spaces. User may just have pressed the enter key	Int_input_val() function Do-while loop prompts user to enter an integer	User is prompted to enter an integer until an integer is entered by the user	User is prompted to re-enter an integer until integer is entered, as expected. Chars, floats and input containing spaces are rejected. If user presses enter, input is rejected
Check lineup order of Characters is the same as the order entered by the user for both teams	n/a - Characters appear in each team lineup in the order in which they were entered by the user, from first entered to last entered	addTail() function of Queue class printQueue() function of Queue class	After the user makes each team selection, the team lineup is displayed in order from first entered to last entered Character	For both team selections, Characters appear in each team lineup in the order in which they were entered by the user, from first entered to last entered
GAME PLAY				
Check Characters at the head of each lineup fight each other	n/a – check the Characters at the head of the lineup at each round fight each other	getHead() function of Queue class getHeadName() function of Queue class	The Characters at the head of the line up fight each round, as per the order of the Character lineup displayed before the tournament starts and after each round	The first Characters entered by the user for both teams are the first to fight in the tournament. After each round, the new Characters at the head of each line up fight, as expected.
Check recovery function is called for winning Character and restores an appropriate amount of strength points following the fight	n/a – check an appropriate number of strength points is added back to the winning Character's total strength points. Their strength points cannot be restored back to their starting strength points	Recovery() function of Character class used by each subclass	Recovery is calculated as follows: Damage = Initial strength – current strength New strength = (Damage*x) + current strength, where x ranges from 0.1 to 0.9 depending on the random number generated.	For each type of Character, recovery is calculated correctly for each winning Character as displayed in the stats after each fight. When the initial strength and current strength are the same, the strength points do not change following recovery. Strength points are

			If the initial strength and current strength are the same, the strength points will not change. The Character's strength points before and after recovery are displayed in the stats after each fight	not restored to higher than the starting strength points for each type of Character, as expected
Check winning Characters move to back of their respective lineups	n/a – check the winning Character of each fight moves to the back of their lineup	moveHead() function of Queue class printQueue() function of Queue class	After the fight, the Character that wins is sent to the back of their line up and the second Character in the lineup is now the head of the lineup, as per the order of the team lineup displayed after each round	The Character that wins each fight is sent to the back of their line up and the second Character in the lineup becomes the head of the lineup, as per the order of the team lineup displayed after each round, as expected
Check losing Characters go to loser pile and are removed from their team lineup	n/a – check losing Character is removed from their team lineup and are added to the head of the loser pile	removeHead() function of Queue class pushHead() function of Stack class	The losing Character is removed from their team lineup and placed on the top of the loser stack, as per the team lineups and loser pile displayed after each round	The losing Character is removed from their team lineup and added to the head of the loser stack, as per the team lineups and loser pile displayed after each round. The loser pile is displayed in order of last defeated to first defeated
The scoring system is implemented correctly for both teams and the winner of the tournament is	n/a – each win accrues 1 point for the respective team and a loss accrues 0 points for the losing team. The team with the	Tournament() function of Game class	After each round, the winning team gets one point added to their current score and the losing team gets 0 points	After each round, the winning team gets one point added to their current score and the losing team gets 0 points added to

determined correctly	most points at the end wins		added to their current score, as displayed in the stats after each round	their current score, as displayed in the stats after each round. At the end of the tournament, the scores are displayed correctly for both teams and the team with the most points wins the tournament, as expected
The tournament continues until one of the teams has no more characters left to fight	n/a – check the Characters at the head of each lineup continue to fight each other while there are still Characters on both teams. When a team has no Characters left, the tournament will end	Tournament() function of Game class	The Characters at the head of each line up fight each round while there are still Characters in each lineup, as per the Character lineups displayed after each round. When one team has no more Characters left, the tournament ends	The Characters at the head of each line up fight each round while there are still Characters listed in each lineup, as shown in the Character lineups displayed after each round. When one team has no more Characters left, the tournament ends, as expected
END OF GAME				
Check loser pile is displayed correctly and in order (last defeated to first defeated)	n/a - When the loser pile is printed all the losing Characters are listed in order of last defeated to first defeated	printStack() function of Stack class	Loser pile prints out all the losing Characters from the tournament in order of last defeated to first defeated	Loser pile prints out all the losing Characters in order of last defeated to first defeated, as expected. No left-over losing Characters from previous games.
User does not enter an integer when prompted to play again or exit program at end of game	Input is a not an integer. May be a character, float, or a mix of characters and integers including spaces	Int_input_val() function Do-while loop prompts user to enter an integer	User is prompted to enter an integer until an integer is entered by the user	User is prompted to re-enter an integer until integer is entered. Chars, floats and input containing spaces are rejected. If user presses enter, input is rejected

CS_162_400_S2019
Project4
Susan Hibbert
ID# 933913975

User enters an integer other than 1 or 2 when asked whether they would like to play again or exit at the end of the game	Input is an integer other than 1 or 2	Display_menu() function called by menu function of Game class While loop prompts user to enter 1 or 2	Loop back to the question prompting user to enter 1 or 2	Program loops back to the question prompting user to enter 1 or 2, as expected.
Check no players are left in either team line up or in loser pile if user opts to play again	n/a – if the user decides to play the game again, no Characters from the previous game remain in any of the team lineups or in the loser pile	Clear() function of Queue class printQueue() function of Queue class printStack() function of Stack class	If the user decides to play the game again, no Characters from the previous game remain in any of the team lineups or in the loser pile as per the Character lineup displayed before the tournament starts and in the loser pile displayed after each round	No Characters from the previous game remain in any of the team lineups as per the Character lineup displayed before the tournament starts. No losing Characters from the previous game remain in the loser pile as displayed after each round, as expected

CS_162_400_S2019
Project4
Susan Hibbert
ID# 933913975

Reflection

There were some important details I missed from my original design description.

One detail I missed from my initial design was the fact that I had to remove the losing Character from their team lineup, in addition to moving them to the Stack. Secondly, it had not occurred to me that if the user decided to play the game again that some Characters from the previous game would be left in the team lineups and in the loser pile. I noticed this issue when testing my program and had to write a function which cleared the team lineups and loser pile between games.

I had an issue with segmentation faults when testing my program. After some debugging I discovered the issue was coming from the removeHead function of the Queue class. I had not accounted for the case where if the head was deleted only one CharNode would remain. I added an extra if statement to my function to account for this scenario and it fixed the issue.

After running valgrind on my program, which appeared to be running normally, I discovered there were some memory leaks. I spent a lot of time debugging but I still could not figure out the cause. After perusing the posts on Piazza, I discovered another student had a very similar issue to me and after reading the replies to his post I realized what I was doing wrong - I was not freeing the dynamically allocated memory for each Character object. After adding an extra delete statement to my Queue and Stack destructors I was able to fix this issue.