Robert Elsom

2/25/2019

Project 4

Reflection:

My biggest issue came from memory leaks. It became hard to trace the memory leak as it went from my game class, to queue, then to character. It was showing as a leak in queue but really was an error in my game class that was not deallocating the memory in the right spot. Also, I had an issue where my printing of the loser container was not working correctly, which I later changed my approach and fixed a few minor errors that I was not seeing in the beginning and everything began working correctly.

Other than the two issues above, everything else were design changes that I had to make, like adding a get and set for team name for characters, thus I could print what team they were on after adding them to the loser container. Also, I reused the Queue function with a few minor changes that at first were not planned in order for it to fit the project and add a little more ability for the user using the queue. I had to change my original plan to include a validStr function to validate Y/N input when prompting to print the loser container. I also changed my original plan for the loser container. At first, I was trying to add to the head of the queue, but instead I decided that printing in reverse would accomplish the same goal. Instead of having head as the top for printing top to bottom, I viewed the last node as the top, and it corrected some errors I was experiencing.

Start Menu/ Repeat Menu test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Input Values | Driver Function | Expected Outcome | Observed Outcome |
| Function a not an unsiged integer | A, 1.5, -4 | ValidInt() | Display error, repeat options to user | Display error, repeat options to user |
| Integer not a valid option | 5 | ValidInt() | Display error, repeat options to user | Display error, repeat options to user |
| Integer a valid option | 1 | validInt(), if statements | Print starting prompts and get inputs for functions | Print prompts and get inputs for functions |
| Integer second option | 2 | Return | Quit program | Quit program |

Character Creation test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Input Values | Driver Function | Expected Outcome | Observed Outcome |
| Function a not an unsiged integer | A, 1.5, -4 | ValidStr() | Display error, repeat options to user | Display error, repeat options to user |
| Integer not a valid option | 10 | ValidStr() | Display error, repeat options to user | Display error, repeat options to user |
| Integer first option | 1 | If input == 1 | Creates new blue men object | Creates new blue men object |
| Integer second option | 2 | If input == 2 | Creates new barbarian object | Creates new barbarian object |
| Integer third options | 3 | If input == 3 | Creates new Harry Potter object | Creates new Harry Potter object |
| Integer fourth option | 4 | If input == 4 | Creates new medusa object | Creates new medusa object |
| Integer fifth option | 5 | If input == 5 | Creates new vampire object | Creates new vampire object |

Print Loser container test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Input Values | Driver Function | Expected Outcome | Observed Outcome |
| Function a not a valid option | A, 1.5, -4 | ValidStr() | Display error, repeat options to user | Display error, repeat options to user |
| Enters a char ‘Y’ or ‘y’ | Y, y | ValidStr() | Prints loser list | Prints loser list |
| Enters a char ‘N’ or ‘n’ | N, n | validStr() | Does nothing | Does nothing |

**Class hierarchy:**

Game class

Queue Class

Vampire class

Barbarian class

Medusa class

Blue Men class

Harry Potter class

Character Class

**Design:**

* Queue class
  + Makes containers with characters in them, holds both team lineups along with the loser container
  + Functions
    - Print
    - Print reverse
    - Add back
    - Add head
    - getFront – returns character in first node
    - removeFront – deletes first node
    - moveToBack – moves winner to back of queue
* Game class
  + Creates lineups, makes appropriate calls to fight for each team, then calls functions to move characters based on outcomes of fight. Also calculates score and displays final results to user
  + Functions
    - newCharacter-Creates characters and stores them in queue elements
    - addToLoser – adds loser to loserQue and deletes from team lineup
    - winnerMove – calls recovery function for character and moves to back of lineup
    - battle – stores results of fights, calculates when team is empty, and displays final score
    - fight – individual fight between characters
    - updateScore
    - printLosers – prints loser list
* Base class
  + Abstract class, virtual functions that = 0
  + Functions
    - Attack
    - Defense
      * Takes roll, attack points, and calculates actual damage inflicted then updateStrength
    - getArmor
    - getStrength
    - updateStrength
    - getName
* Menu
  + Display names and get user to pick two
    - Can pick two of the same characters
  + Print round stats including
    - Attacker type
    - Defender type, armout, strength
    - Attackers die roll
    - Defenders die roll
    - Damage inflicted calculations
    - Defenders updated strength points after attack
      * If defenders die, ends game
  + Continue
    - 1 continue
    - 2 quit
* Vampire class
  + Attack
    - 1 12 sided die
  + Defense
    - 1 6 sided die
    - Charm power- 50% chance to charm attacker into not attacking for any attack
      * Trumps medusas glare if happens at same time
    - Takes roll, attack points, and calculates actual damage inflicted then updateStrength
  + getArmor
    - 1
  + getStrength
    - 18
  + updateStrength
* Barbarian class
  + Attack
    - 2 6 sided die
  + Defense
    - 2 6 sided die
    - Takes roll, attack points, and calculates actual damage inflicted then updateStrength
  + getArmor
    - 0
  + getStrength
    - 12
  + updateStrength
* Blue men class
  + Attack
    - 2 10 sided die
  + Defense
    - 3 6 sided die
    - For every four points strength lost, lose a defense die, represents loosing a couple of little blue men
    - Takes roll, attack points, and calculates actual damage inflicted then updateStrength
  + getArmor
    - 3
  + getStrength
    - 12
    - Mob power- for every four points strength lost, loses one defense die
  + updateStrength
* Medusa class
  + Attack
    - 2 6 sided die
    - Glare power – if rolls a 12, target instantly gets turned to stone and medusa wins, unless against harry potter who still has Hogwarts power left
  + Defense
    - 1 6 sided die
    - Takes roll, attack points, and calculates actual damage inflicted then updateStrength
  + getArmor
    - 3
  + getStrength
    - 8
  + updateStrength
* Harry Potter class
  + Attack
    - 2 6 sided die
  + Defense
    - 2 6 sided die
    - Takes roll, attack points, and calculates actual damage inflicted then updateStrength
  + getArmor
    - 0
  + getStrength
    - 10
    - Hogwarts power- if strength <= 0, resets to 20 one time per fight
  + updateStrength