Christian Roccanova

CS 162 Project 4

5/27/2017

Project 4 Reflections

The purpose of project 4 is to build a fantasy combat tournament based on the original combat game from project 3. To this end, the creatures from project 3 will need to be stored in two separate queues, each representing a team. This will necessitate the creation of a queue class with functions to add fighters to the back of the queue and remove them from the front of the queue. It will also need a recovery function to take the winner of a fight and add them back to the end of their team. The loser will be added to a third queue called loserPile. A display function will need to be added to print the loser pile at the end of the fight. Classes for each creature type (Harry Potter, BlueMen, Barbarian, Vampire and Medusa) will be taken from the files of project 3. These all inherit from the parent Creature class, which will also come from project 3. Set and get functions will be added to the creature class to allow the user to make custom names for each fighter. As in project 3, the game itself will be run by a game class which will call the fight menu to choose the number of fighters the teams will use and what kind of creatures they will be. The fight function will then pull the first fighter from each team and call their attack and defense functions. When a fighter is reduced to 0 strength it will be removed to the loserPile and the winner will be returned to the end of their team.

Test	Expected Outcome	Observed Outcome
Attempt to input a number outside of those defined in the main menu	Prints an error message and asks for a new input.	Prints an error message and asks for a new input.
Attempt to input a number outside of those defined in the fight menu	Prints an error message and asks for a new input.	Prints an error message and asks for a new input.
Attempt to input a number outside of those defined in the loser menu	Prints an error message and asks for a new input.	Prints an error message and asks for a new input.
Input a non-integer for team size	Prints an error message and asks for a new input.	Prints an error message and asks for a new input.
Play a game with one of each creature class	The game should run, showing attack and defense rolls for each creature. After each round it should display the winner and after one team is defeated it should print the scores and determine the victor.	Initially caused an infinite loop when it came to Harry Potter vs. Harry Potter. The game ran as expected after the fix described in the reflections below.
Play the above game and choose to display the losers.	The 9 losers should be printed to the screen, showing their custom	The custom names of the losers are displayed in list format.

	names in a list.	
Play a game with all barbarians.	The game should run, showing attack and defense rolls for each creature. After each round it should display the winner and after one team is defeated it should print the scores and determine the victor.	The game runs, showing attack and defense rolls for each creature. After each round it should display the winner and after one team is defeated it should print the scores and determine the victor.
Play the above game and choose to display the losers.	The losers should be printed to the screen, showing their custom names in a list.	The losers printed are from the correct team, however the names are all that of the final fighter of each team.
Choose a vampire	50% of attacks against the vampire should not do any damage to reflect the Charm skill	Charm is observed to activate roughly 50% of the time
Choose Blue Men	The defense roll of the Blue Men should decrease for every 4 strength points lost such that the maximum possible roll decreases by 6 each time.	The Blue Men defense rolls decrease as they lose strength.
Choose a Medusa and continue until a twelve is rolled.	When the Medusa rolls a 12, the opponent should automatically lose all strength points and lose the game.	The Medusa hits the opponents such that their defense is completely overpowered.
Choose Harry Potter and play until he loses his first life.	When Harry is reduced to zero Strength, then he should reset his strength to 20.	Harry dies once and is resurrected.
Choose a Medusa and continue until a twelve is rolled, against Harry Potter.	Harry's Strength should be reduced to zero when the Medusa uses Glare, but he should then immediately reset to 20 on his first life.	Harry dies once and is resurrected.
Choose Harry Potter and play until he loses his second life.	Harry should not resurrect, remaining at 0 strength, losing the game.	Harry dies once, resurrects and then permanently dies.
Enter an invalid menu option.	Prints an error message and asks for a new input.	Prints an error message and asks for a new input.
Fight a Medusa and a Vampire until glare and charm both trigger.	Charm should overpower glare, the vampire should take no damage.	The vampire takes no damage.
Choose to play again after the first game.	The game should run normally.	The game runs normally.
Choose to quit.	The program should shut exit.	The program exits.

During the first test, the program crashed because it tried to get a fighter from an empty queue. To remedy this, an isEmpty function was added to the Queue class to check if either team's queue is empty. If the queue is empty, the program exits the fight loop and compares team scores to determine a winner. Another issue encountered was that fighting Harry Potter vs. Harry Potter caused an infinite loop. This turned out to be because the "life" variable that determined if his resurrect ability should trigger was being reset every time he defended. This made him effectively, undefeatable. Moving the "life" variable outside of the defense function solved this issue. One other issue that was revealed during testing was that when multiples of a creature were included in a single team, only the name of the last added creature would be displayed in the loser pile (ie if barbarian Bob was added after barbarian Dave and both lost, two barbarian Bobs would be present in the loser pile). Also, as an integer input was required, an InputValidation class was added to validate that the user input was indeed an integer. Further, the instructions stated that the recovery function should restore some strength to the victor so a line to restore a random 1d6 strength was added to the recover function.