CS 162 Project03 Reflection

Hailey Wilder

Design

I plan to create an overarching class called Creature which is referenced by the five required combatants (Vampire, Barbarian, Blue Men, Medusa, and Harry Potter). This Creature class will be abstract so as not to be a combatant itself, but rather a form for the other combatants. I will also have a class called CombatRunner which will execute combat sessions and display the play progression. The main function will loop through the play of repeated games and select which types of games/ characters will battle.

Classes

Creature:

Protected members:

- Int armor
- Int strength
- Std::string name

Public members:

- Creature() – simple constructor to seed the random number generator
- Virtual int attack () = 0
- Virtual int defend() = 0
- Virtual int take_damage(int attack, int defense)
    - Attack -= armor
    - Damage -= attack – defense
    - If(damage >0)
        - Strength -= damage
        - Return damage
    - Else
        - Return 0
- Bool is_dead()
    - Return true if strength <-0
- Int sum_dice(int #dice, int sides)
- Void revive() = 0
- Std::string get_name()

HarryPotter:

Protected members:

Public members:

- HarryPotter()
    - Armor, strength, name
- Int attack()
    - Dice

  - Return sum
- Int defend()
  - Dice
  - Return roll
- Int damage(int attack, int defense)
- Void revive()
- Bool get_special

Barbarian:

Protected members:

Public members:

- Barbarian()
  - Armor, strength, name
- Int attack()
  - Dice
  - Return sum
- Int defend()
  - Dice
  - Return roll
- Int damage(int attack, int defense)
- Void revive()
- Bool get_special

Medusa:

Protected members:

Public members:

- Medusa()
  - Armor, strength, name
- Int attack()
  - Dice
  - Return sum
- Int defend()
  - Dice
  - Return roll

- Int damage(int attack, int defense)
- Void revive()
- Bool get_special

BlueMen:

Protected members:

Public members:

- BlueMen()
    - Armor, strength, name
- Int attack()
    - Dice
    - Return sum
- Int defend()
    - Dice
    - Return roll
- Int damage(int attack, int defense)
- Void revive()
- Bool get_special

Vampire:

Protected members:

Public members:

- HarryPotter()
    - Armor, strength, name
- Int attack()
    - Dice
    - Return sum
- Int defend()
    - Dice
    - Return roll
- Int damage(int attack, int defense)
- Void revive()

- Bool get_special

CombatRunner:

Protected members:

- Creature*p1
- Creature*p2
- Const bool p1_won = true
- Const bool p2_won = false
- Int simulations
- Int currentSim

Public members:

- CombatRunner(Creature*p1, Creature*p2, int simulations)
- Bool run_simulation()
- Int get_result()

Main():

- Create two arrays of pointers to Creatures
- Use nested loops to pair each pointer in the first array with those in the second
- Free memory from pointers in arrays
- Exit

Test Plan

I will incrementally develop functions working on fundamental functions first and test them before using them in larger code. Once basic functions for creatures are done, to verify that damage is correctly applied during each turn of gameplay, I will print statements to display numerical values representing dice rolls, sums of dice rolls, base attack, armor, defense, and current strength/ defense functions. Finally, I will print a final verification of the winner when the game is over. I will use the play bool values to count the number of games each player wins in my main() function. This will help me find overall test results. I expect that when similar matched types are paired up with one another the results will be approximately even (50-50). However, unmatched players will have unmatched results.

Test Results

I ran into a few snags with syntax while coding, but was able to work through them by focusing on small portions of code at a time. I also tested one class with a test main function to ensure that my classes were working as expected before continuing to build the rest of them. Initially I seeded srand() incorrectly (with 0 when I meant to do time(0)) which resulted in player 1 always winning at the same ratio.

In the table below the results of creature matchups are displayed for 5000 simulations. In each set of simulations where a Creature object fought its same type (excluding Harry Potter), the results were close to 50%, but slightly in favor of player 1. This is due to the fact that player 1 always attacks first. In the case of Harry Potter, player 1 wins by a large margin because of the special involved.

	Player 2					
Player 1		Harry Potter	Barbarian	Medusa	Blue Men	Vampire
	Harry Potter	99.96	34.28	34.1	0	24.38
	Barbarian	100	60.36	57.46	0	58.48
	Medusa	84.48	55.84	55.58	0	58.16
	Blue Men	100	100	100	55.14	100
	Vampire	92.12	59.92	60.12	0	58.12