Cody Hannan
CS162

PROJECT 4 - REFLECTIONS

DESIGN DESCRIPTION

Purpose:

In this project we are tasked with writing program that will run a tournament using the creatures from project 3. Linked structures will be utilized to manipulate and hold the data as the tournament progresses.

Requirements:

- A user will enter the number of fighters both players will use. The user should also enter the type of creatures and fighter names.
- The first player supplies a lineup of creatures in order. The second player then supplies his lineup of creatures in order.
- The head of each lineup fight in the same way they fight in project 3. The winner gets put at the back of her/his team's lineup; the loser goes to the container for those who lost.
- The lineup order cannot be changed once the players are entered. Similarly, the loser pile should have the last loser at the top and the first loser at the bottom.
- Even if a creature wins, she/he may have taken damage. Some percentage of the damage should be restored when they get back in line.
- After each round, the type of creatures that fought and which won should be displayed on screen as well as the updated point totals. At the end of the tournament (when one team or both run out of fighters in the lineup), the program should display final total points for each team. The winner must also be determined and displayed.
- An option to display the contents of the loser pile at the end of the tournament must be provided, i.e. print them out in order with the loser of the first round displayed last.

Design:

<u>Fight Class:</u>

Contains the functions for the where the fight occurs. The constructor will initialize the two fighters and then carry out the battle by calling the attack, defence, and get armor functions in order to calculate damage. The damage will be subtracted from a health variable, which will have been initialized using the getter function for strength.

<u>Rosters Class:</u>

This class will create a linked structure to hold creature objects. Should contain methods for adding and removing creature objects. The player will add fighters to each team and then remove the fighter from the list or move it to the back as the tournament progresses. When fighter wins, their health is fully restored and they are sent to the back.

<u>Creature Class:</u>

The abstract base class. It will contain pure virtual functions for attack and defense, which will be defined in each subclass. This class will also contain protected variables for armor and strength, which will be constructed in the subclasses. There will be getter functions to call the armor and strength values.

Creature Subclasses:

The menu function will display the menu options available to the user.

Menu function::

The menu function will display the menu options available to the user.

Input Validation function:

The inputValidation function will validate user inputs by checking that entered values are in the correct range and of the correct type.

Main function:

The main function will primarily be where the user selects menu options and selects the fighters that will battle.

Pseudocode:

Fight Class:

```
Fight class constructor that takes user input for which fighter will battle
    If fighter1 is Vampire
        Allocates memory for a vampire object and assigns it to ptr1
    .
    .
    .
    If fighter2 is Vampire
        Allocates memory for vampire object and assigns it to ptr2
    .
    .
    .
    There are if statements for each class.

Fight class winner function that executes the battle and determines the winner
    Get fighter1 health
    Get fighter2 health

    While fighter 1 and 2 health is more than 0
        Randomly choose the target

    Output who is attacking to user

    Damage=fighter attack - opposing fighter defense and armor

    If damage  is less than 0
        damage=0

    Damage is deducted from fighters health
    Protected variable of corresponding object is changed
    Output value to new value to user

    If health is less than 0
        Return other fighter
```

Creature Class:

```
class Creature
{
Protected variables for armor and strength
Public:
    Constructor takes values to init armor and strength
    Pure virtual functions for attack and defense
    Getter functions for armor and strength
    Setter function for strength
```

Menu function:

Outputs menu options available to user:

1. Battle.
2. Exit program.

Input Validation function template:

```
Function call template is inputValid(int min, int max, string "variable")
    If variable == "some type of input that is needed"
        Cin >> User inputted value
        while(cin fails or input is outside of max and min range)
            If cin failed
                Notifies user to enter the proper value type.
                Cin >> user inputted value
            Else if value is outside of range
                Notifies user to enter a value within max and min range
                Cin >> user inputted value

    If variable == "some other type of input that is needed"
    .
    .
    .
```

Main function:

```
While option doesn't equal 2
    Menu function is called
    User is prompted for selection

    If selects 1
        Ask user to choose their fighters
        List the available fighters

        User is prompted for fighter one and fighter two

        Fight object is created

        Outputs winner to the user
```

TEST PLAN / RESULTS

Case	Input	Function	Expected Outcome	Actual Outcome
Vampire vs other fighters	User selects option 1 and chooses fighters	fight.winner()	Outputs Fight progression and winner. Charm should occasionally activate	Outputs Fight progression and winner. Charm occasionally activated
Barbarian vs other fighters	User selects option 1 and chooses fighters	fight.winner()	Outputs Fight progression and winner	Outputs fight progression and winner
BlueMen vs other fighters	User selects option 1 and chooses fighters	fight.winner()	Outputs Fight progression and winner. Defense should lower as strength decreases.	Outputs Fight progression and winner. Defense lowered as strength decreased.
Medusa vs other fighters	User selects option 1 and chooses fighters	fight.winner()	Outputs Fight progression and winner. Glare should occasionally activate	Outputs Fight progression and winner. Glare occasionally activated.
HarryPotter vs other fighters	User selects option 1 and chooses fighters	fight.winner()	Outputs Fight progression and winner. Should revive when health reaches 0	Outputs Fight progression and winner. Revived when health reached 0
Fighters vs themselves	User selects option 1 and chooses fighters	fight.winner()	Should work the same. Fighters are differentiated by fighter one: and fighter two:	Worked the same. Fighters are differentiated by fighter one: and fighter two:
Team1 vs team2	User selects option2 then 3		The tournament should play out as described in the requirements	The tournament plays out as described in the requirements.

REFLECTIONS

This assignment mostly drew from our last couple labs and project 3. Initially I fixed up the child class interactions and cleaned up the some of the code from my project 3. Then I attempted to implement the object class into a linked list. I decided to keep it simple and use a non circular and non double linked linked list opting instead to just add the creature type to the back of the list and removing the one at the front if it lost. The winning creature's health is then fully restored. The most difficult part of this project was taking what we learned from lab6 and lab7 and adapting the linked lists to utilizing objects instead of values, where the main issue for me was properly understanding the syntax. Another was decided where to use the new operator when creating creature classes. What ended up working best for me was including those in the Rosters constructor rather than in the listNode struct, since it made it easier to destruct the allocated space in the Rosters destructor. Overall I felt better about this project and felt more comfortable with the memory allocation and deallocation than I have in the past.