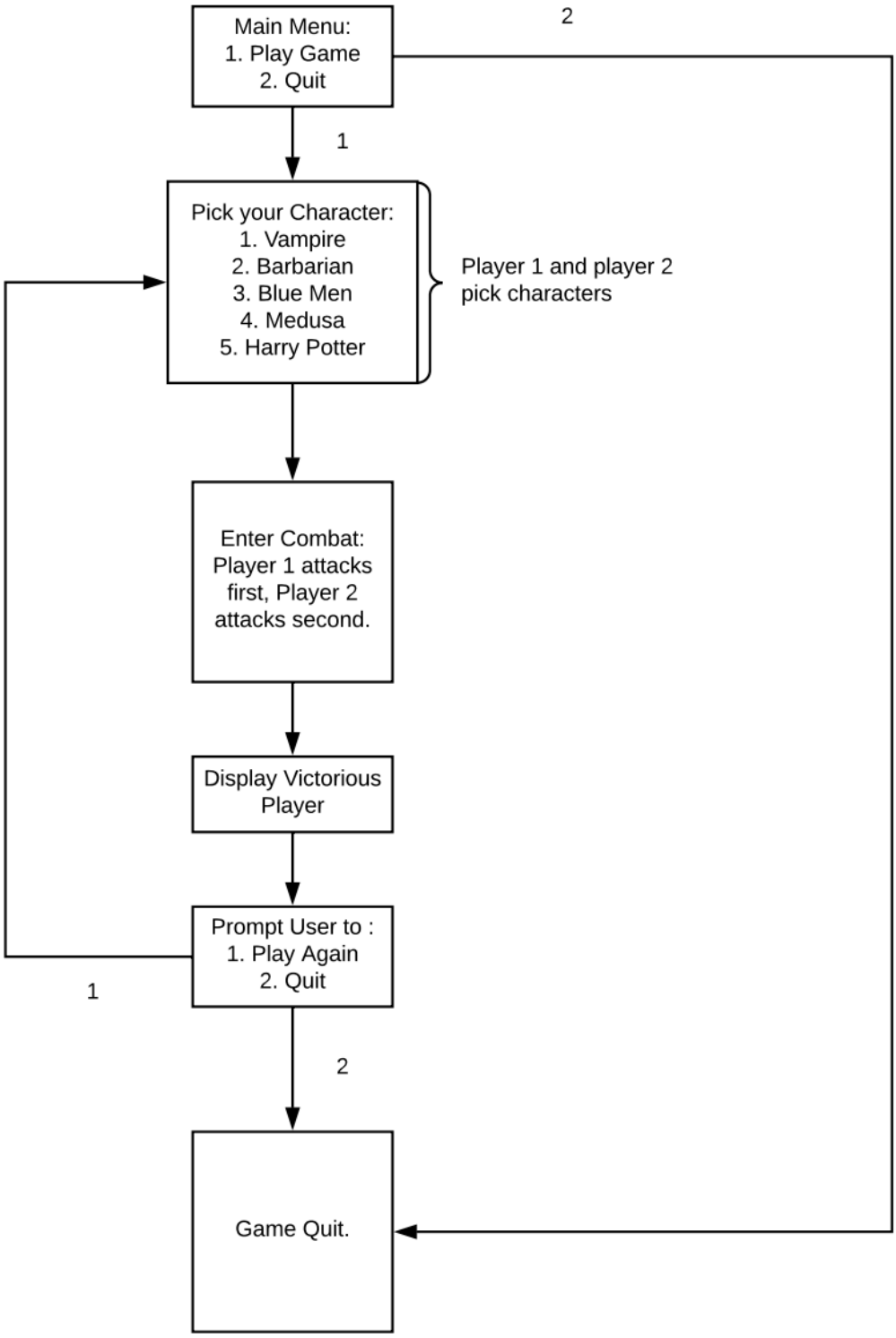
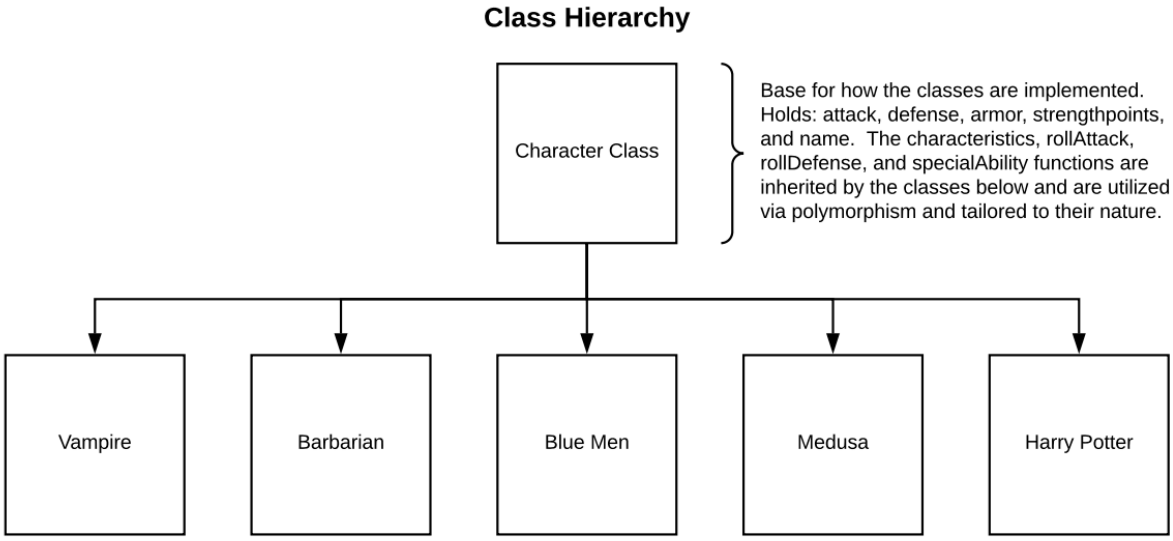
Devin Gendron
Project 3 – Fantasy Combat Game

Flowchart:



Hierarchy:



Vampire:Constructor initializes attack (0), defense (0), armor (1), strengthPoints(18), and name (Vampire).
Characteristics: Suave, debonair, but vicious and surprisingly resilient.
rollAttack:Rolls one 12 sided die.
rollDefense: Rolls one 6 sided die.
specialAbility: Charm - 50% of the defense rolls activate charm special ability where the Vampire will not be attacked.

Barbarian: Constructor initializes attack (0), defense (0), armor (0), strengthPoints(12), and name (Barbarian).
Characteristics: Think Conan or Hercules from the movies. Big sword, big muscles, bare torso.
rollAttack: Rolls two 6 sided die.
rollDefense: Rolls two six sided die.
specialAbility: No special ability.

Blue Men: Constructor initializes attack (0), defense (0), armor (3), strengthPoints(12), and name (Blue Men).
Characteristics: They are small, 6 inch tall, but fast and tough. They are hard to hit so they can take some damage. They can also do a LOT of damage when they crawl inside enemies' armor or clothing.
rollAttack: Rolls two 10 sided die.
rollDefense: Rolls die depended on special ability. Rolls three six sided defense die, then 2, then 1. Rolls dependent on strengthPoints.
specialAbility: SP > 8 = three defense rolls, SP > 4 = two defense rolls, SP > 0 && SP < 4 = 1 defense roll.

Medusa: Constructor initializes attack (0), defense (0), armor (3), strengthPoints(8), and name (Medusa).
Characteristics: Scrawny lady with snakes for hair which helps her during combat. Just don't look at her!
rollAttack: Rolls two 6 sided die.
rollDefense: Rolls one 6 sided die.
specialAbility: If attack roll == 12, then Medusa uses glare and turns her opponent to stone.

Harry Potter: Constructor initializes attack (0), defense (0), armor (0), strengthPoints(10), and name (Harry Potter).
Characteristics: Harry Potter is a wizard.
rollAttack: Rolls two 6 sided die.
rollDefense: Rolls two 6 sided die.
specialAbility: Hogwarts ability - Harry returns to life the first time he dies.

PseudoCode:

Main()

```
//Menu function
//switch statements to play game or quit
        //If game is played, choose which characters to play as.
```

Menu()

```
//print statements
//use inputvalidation func to receive proper return
//return value
```

playAgainMenu() //user chooses if they would like to play the game again

```
//print statements
//use inputvalidation func to receive proper return
//return value
```

charMenu()

```
//using inputval(); to choose character types.
```

contCombat()

```
//using inputval(); to continue flow of combat
```

Combat()

```
//sets characters to their players (Player 1 and Player 2)
//using loop, have player 1 attack and player 2 defend, then if player 2 is alive, they
attack player 1.
        //While combat ensues, print information on the characters as they battle.
//If there is a victorious player, display who won and who lost.
//else, ask they user to continue the combat.

-This loops until there is a victor and the loop is ended.
```

Test Plan:

Menu();

Test Case	Expected Outcomes	Observed Outcomes
Input letters	Inputvalidation(); returns "Incorrect entry…". Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input symbols or space	Inputvalidation(); returns "Incorrect entry…". Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input too low	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Input too high	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Empty input	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Correct Input	Counter for exit increments, print correct entry statement. Program continues.	Returns value to main to continue with program

playAgainMenu();

Test Case	Expected Outcomes	Observed Outcomes
Input letters	Inputvalidation(); returns "Incorrect entry…". Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input symbols or space	Inputvalidation(); returns "Incorrect entry…". Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input too low	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Input too high	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Empty input	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Correct Input	Counter for exit increments, print correct entry statement. Program continues.	Returns value to main to continue with program

charMenu();

Test Case	Expected Outcomes	Observed Outcomes
Input letters	Inputvalidation(); returns “Incorrect entry…”. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input symbols or space	Inputvalidation(); returns “Incorrect entry…”. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input too low	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Input too high	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Empty input	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Correct Input	Counter for exit increments, print correct entry statement. Program continues.	Returns value to main to continue with program

contCombat();

Test Case	Expected Outcomes	Observed Outcomes
Input letters	Inputvalidation(); returns “Incorrect entry…”. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input symbols or space	Inputvalidation(); returns “Incorrect entry…”. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input too low	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Input too high	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Empty input	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Correct Input	Counter for exit increments, print correct entry statement. Program continues.	Returns value to main to continue with program

Medusa vs Harry

Test Case	Expected Outcomes	Observed Outcomes
Medusa uses glare	Harry survives using hogwarts	Harry survived.
Medusa uses glare again	Harry dies after using up his Hogwarts ability	Harry dies.

Vampire vs Medusa

Test Case	Expected Outcomes	Observed Outcomes
Medusa uses glare	Vampire uses charm	Vampire survives
Medusa uses glare	Vampire uses no ability	Vampire dies

Reg damage vs Harry

Test Case	Expected Outcomes	Observed Outcomes
Char attacks harry	Harry survives using hogwarts	Harry survived.
Char attacks harry	Harry dies after using up his Hogwarts ability	Harry dies.

Reg damage vs Vampire

Test Case	Expected Outcomes	Observed Outcomes
Char attacks vampire	Vampire uses charm	Vampire survives
Char attacks vampire	Vampire uses no ability	Vampire dies

Medusa vs Char

Test Case	Expected Outcomes	Observed Outcomes
Medusa uses glare	Character dies	Char died

Blue Men taking damage

Test Case	Expected Outcomes	Observed Outcomes
Blue Men takes 4 points of dmg	They lose a defense die	Defense die is lost
Blue Men take 8 points of total damage	They lose two defense die	Defense die is lost

Reflection:

I felt super comfortable starting this project. I've become much more confident in my use of pointers, class objects, inheritance, and polymorphism - so this project wasn't as difficult as the first two. Designing and structuring this program was fairly simple considering that menu functions and input validation was conquered earlier in the course. The prior assignments involving inheritance and polymorphism really helped me wrap my head around the subject matter, so I did not hit any road blocks in regard to their technical application and implementation.

I wrote the initial program with the barbarian in about an hour and had two barbarians locked in combat without issue, so I began implementing the other characters. Slowly but surely, I had a full line up with working special abilities using polymorphism and inheritance, but then my first issue arose. The way I had structured my combat system did not include a way for polymorphism to implement Harry Potter's special ability to revive himself (since this was a between turns skill). I brainstormed on it for a bit and left the program to the side while I worked on the group project. After I had come up with a solution to my problem, I ran it by a TA during office hours to make sure that my implementation was in accordance with the assignment specifications. With their word, I made my changes to my program.

First, I created a special ability function for the characters to use. Previously, I had hard coded the special abilities in the dice roll functions. This was a simple change and allowed a function call to utilize the special abilities. I then changed each special ability call for each character to only call if the character was alive. Then with some restructuring of my combat sequence, I had a fully working fantasy combat game!

To fully summarize project 3, I thought that it was an easier project. I believe that this is only the case, because of the really steady increments of difficulty of each lab and project. I also followed the suggestion in the assignment to implement the barbarian first, and I believe that was a great starting point. So as we get deeper and deeper into the course and subject matter, more concepts are clicking with me and designs that would have previously been much too difficult are becoming an easy task and it's very exciting to grow more confident in my knowledge of c++ and computer science.