Stephen Townsend

# CS 162 – Assignment 3

## Design

The design of Assignment 3 involved creating a Fantasy fighting game. The user will be shown a menu that will display the type of creatures that they can make. The game will be for two players and ask for player1 creature. The program will then load the character and ask for player2 to choose their creature. This will be the only input the user will need besides pressing the spacebar to proceed to each battle.

The settings for each class will need to be hard coded into the objects themselves. I will use the game file to create a creature object. The object will use the abstract creature class and use pure void functions. Given the data is hard coded, I will only be using the constructor in the creature class to set the data for each derived class. The game file should just call the creature class and then depending on the object that was created, the respectable dependent class will be called to use their functions.

The assignment specifically asks for the creature class to be an abstract. I feel that this will cause the use of a lot of duplicate code. From the start, I feel that if I have a getArmor function, then why should I have to have 5 of these functions. One in each separate derived class. I am not sure if I can keep the creature class abstract if there are some functions that are not virtual and set to 0.

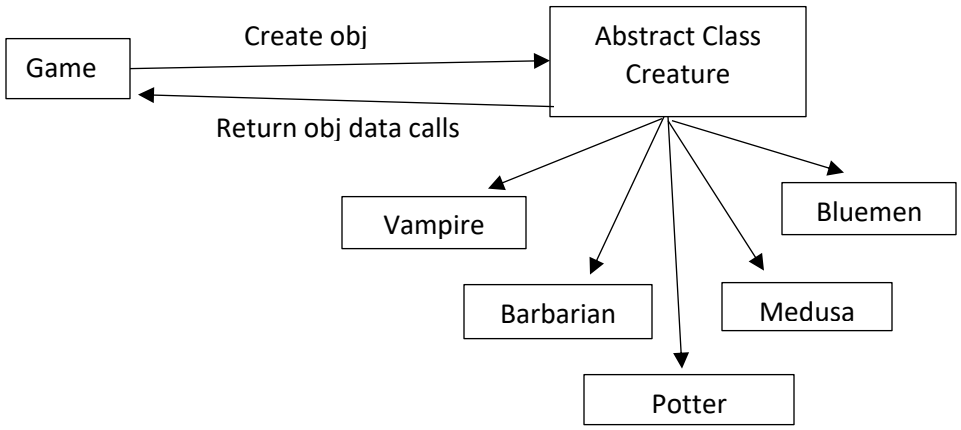
Once the data is hard coded then the play function in game will be used to run the game. This should have options to call the attack and defense functions. The attack and

defense should have the ability to call a roll function. To do this, the roll function will have to be created to be universal.

While the user progresses through the program, the program will need a way to keep track of the player's health and output it to the user so that they understand the changes. There also needs to be an output for each roll. All creatures will be using their own set of dice. Whether it is multiple dice or a single die. Many having different number of sides.

Once the player's health reaches zero, there needs to be an output that displays the final results and who won the game. From here the user will have another option to either continue playing the game with Y or exit with N.

## Hierarchy




## Drivers

Main

menu

1. Vampire
2. Barbarian
3. Blue Men
4. Medusa
5. Harry Potter

Player1.loadCreature();
Player2.loadCreature();
play(player1, plaer2)

Game

Play()
Display()
LoadCreature()

Class

Virtual void attack() = 0;
Virtual void defend() = 0;
Virtual void diceRoll() = 0;

## Test Plan

| Player1 | Player 2 | Outcome |
|---|---|---|
| 1 | 2 | The program should go to loadCreature function in game and return a Creature object that was dynamically allocated. |
| Anything besides int | - | The program should have to tell the user that the input is not valid. |
| 1 | 1 | Same creature. The program should make two vampire objects. |
| 3 | Int > 5 | The program will create the object for player 1, but will ask player 2 to enter a valid input. |

## Reflection

For the program, I kept the overall general layout of my design as described above. When I started, I decided to implement one class. I started with the Vampire. This class seemed like it would be simple to work out any issues for.

I tried to make my objects as simple as possible. I did not include much data besides the strength and armor to start off. I found that this would not be enough because while vampire only uses one die for each attack and each defense roll, the other classes use a variation of dice.

As a result, have the attack and defense functions call the roll dice function. I used return rollDice(6); in side both of the functions. The six is the number of sides the dice has. This number would be used to make a range for my dice 1 – n and return a random int for the roll. This worked well because I can pass a constant int into the roll. Even though my return is random, the input is not and I was able to find any problems by taking a sample of rolls and testing if the program added correctly.

When I implemented other classes, I found that to do this constant in the attack and defense would be problematic and if I needed to change the dice sides for each class later on then it would be much harder to update attack and defense function in all 5 classes individually.

This was the first change. I decided to add the dice sides and the number of dice into the object. As a result, the data would be in one location (loadCreature function in game). This means now all I have to do is pass the name for the sides and amount of dice into the roll dice function and it is easier to update at a later time.

After implementing the new change with the objects, I needed to figure out a way to deal with multiple dice rolls and the Blue Men special. This posed a challenge but I decided to make the dice roll a recursive function to combat the multiple dice needed. This made the dice roll much easier because as the number of rolls passed in changes, the depth level needed also changes.

The next major challenge that I had was implementing Medusa's special ability. Each class has a special function and if they had a special something would happen when the criteria was met. Medusa on the other had had a special that did not affect her, but the opponent. As a result, her special function is not able to touch the opponents class. As a result, when doing the damage calculation, I decided to add an if statement that said,

```
if (player1->getCreature() == "Medusa" && attack == 12) //If player is Medusa (check for special kill)
{
	player1->special(); //Call Medusa's special dialog.
	player2->damage(100); //Insta kill opponent.
	player2->isAlive(); //check and set player dead if strength == 0.
}
```

This means that if the player is Medusa and they roll a 12 then the opponent would take 100 damage. The 100 is used to make sure any creature will die because this is an insta win for Medusa. If the player is not Medusa then the damage will be ran as normal.

```
else
{
	damage = player1->damage(attack - defense); //Send damage calculation to player 1.
	player1->isAlive();
}
```

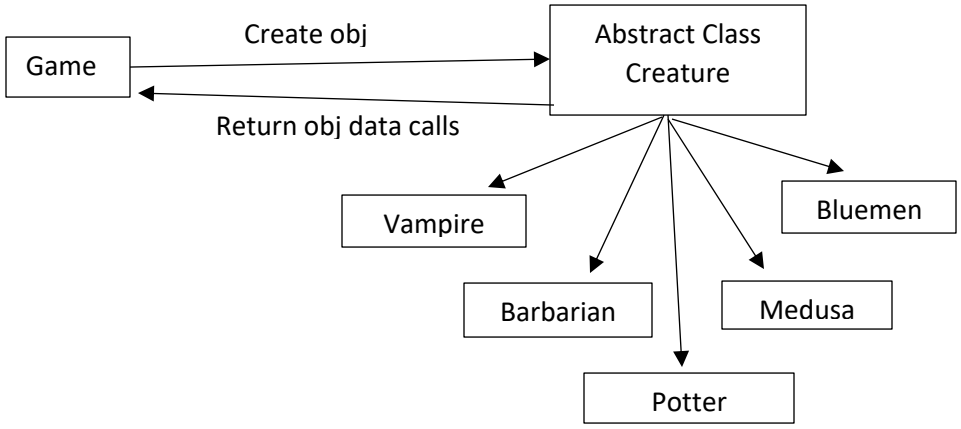
This is how I took care of Medusa's special challenge that I had.

Other than what was listed above, the last challenge I had was fixing the output and make sure the information I want is presented easily. I did go back and forth of a design choice. Most of the time I had the attack and defense totaled then outputted to the user. At the last minute, I decided to change this and display each roll instead. I wanted to show that the special of the Blue Men worked. As a result, it can now be seen that when their health is lowered by 4 then they lose a dice roll.

Stephen Townsend

# Final

## Hierarchy




## Drivers

**Changes from original (added) are in blue

Main

menu
1. Vampire
2. Barbarian
3. Blue Men
4. Medusa
5. Harry Potter

choice = inputVal(choice)
Player1.loadCreature();
Player2.loadCreature();
play(player1, plaer2)
play again loop

Game

Creature *loadCreature(string creature);
void storyLine(string);
void play(Creature *player1, Creature *player2);
void getTombStone(Creature *player);
void display(Creature*, Creature*);
string inputVal(string);

Class

virtual string getCreature() = 0;
virtual int attack() = 0;
virtual int defense() = 0;
virtual int getArmor() = 0;
virtual void isAlive() = 0;
virtual STATUS getState() = 0;
virtual int getStrength() = 0;
virtual int damage(int) = 0;
virtual int rollDice(int, int, int) = 0;
virtual SPECIAL special() = 0;

Creature(); //Default constructor
Creature(string, int, int, int, int, int, int);

Stephen Townsend

# Test Plan

My test plan came out the same as what I predicted.

| Player1 | Player 2 | Outcome |
|---|---|---|
| 1 | 2 | The program should go to loadCreature function in game and return a Creature object that was dynamically allocated. |
| Anything besides int | - | The program should have to tell the user that the input is not valid. |
| 1 | 1 | Same creature. The program should make two vampire objects. |
| 3 | Int > 5 | The program will create the object for player 1, but will ask player 2 to enter a valid input. |

| Play again | Outcome |
|---|---|
| Y | Program will start back over. |
| N | Program will exit. |
| Any other input | Pleas enter a valid choice. |