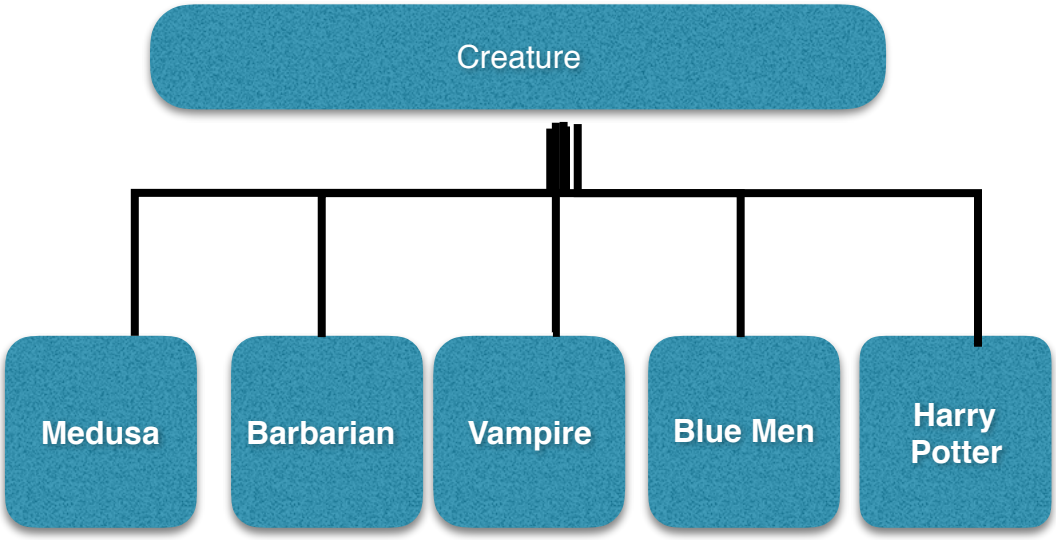

Creature

- attack_Die_Sides
- attack_Die_Rolls
- defense_Die_Sides
- defense_Die _Rolls
- armor_Value
- strength_Points

- Medusa
  - Constructor
    - ♣ attack_Die_Sides = 6
    - ♣ attack_Die_Rolls = 2
    - ♣ defense_Die_Sides = 6
    - ♣ defense_Die _Rolls = 1
    - ♣ armor_Value = 3

- ♣ strength_Points = 8

- int getAttack()
  - ♣ Glare
    - Roll 2 6 sided Die (Roll 6 sided die twice and sum total)
    - If Die Roll = 12
      - AUTOMATIC WIN
    - Else
      - return sum of 2 rolls
- int getDefense()
  - ♣ Roll 1 6 sided die
  - ♣ Return Roll
- int getArmor()
  - ♣ Return armor
- void applyDamage (int Attack(from other character), int Armor, int Defense)
  - ♣ change = Attack – (Armor + Defense)
  - ♣ If change > 0
    - strength -= change
  - ♣ If strength < 0
    - Return 0
  - ♣ Else
    - Return strength
- int getStrengthPoints()
  - ♣ Return strength

- Barbarian
    - Constructor
        - ♣ attack_Die_Sides = 6
        - ♣ attack_Die_Rolls = 2
        - ♣ defense_Die_Sides = 6
        - ♣ defense_Die _Rolls = 2
        - ♣ armor_Value = 0
        - ♣ strength_Points = 12
    - int getAttack()
        - ♣ Roll 2 6 sided Die (Roll 6 sided die twice and sum total)
        - ♣ Return sum of 2 rolls
    - int getDefense()
        - ♣ Roll 2 6 sided die (Roll 6 sided die twice and sum total)
        - ♣ Return sum of 2 rolls
    - int getArmor()
        - ♣ Return armor
    - void applyDamage (int Attack, int Armor, int Defense)
        - ♣ change = Attack – (Armor + Defense)
        - ♣ If change > 0
            - strength -= change
        - ♣ If strength < 0
            - Return 0
        - ♣ Else

- Return strength

- int getStrengthPoints()
  - ♣ Return strength

- Vampire
  - Constructor
    - ♣ attack_Die_Sides = 12
    - ♣ attack_Die_Rolls = 1
    - ♣ defense_Die_Sides = 6
    - ♣ defense_Die _Rolls = 1
    - ♣ armor_Value = 1
    - ♣ strength_Points = 18
  - int getAttack()
    - ♣ Roll 1 12 sided Die
    - ♣ Return roll
  - int getDefense()
    - ♣ Roll 1 6 sided die
    - ♣ Return Roll
  - int getArmor()
    - ♣ Return armor
  - void applyDamage (int Attack, int Armor, int Defense)
    - ♣ Charm
      - Rand num % 2
      - If rand num == 0

- Do nothing
- else
  - change = Attack – (Armor + Defense)
  - If change > 0
    - ♣ strength -= change

- int getStrengthPoints()
  - ♣ Return strength

- Blue Men
  - Constructor
    - ♣ attack_Die_Sides = 10
    - ♣ attack_Die_Rolls = 2
    - ♣ defense_Die_Sides = 6
    - ♣ defense_Die _Rolls = 3
    - ♣ armor_Value = 3
    - ♣ strength_Points = 12
  - int getAttack()
    - ♣ Roll 2 10 sided Die
    - ♣ Return sum of rolls
  - int getDefense()
    - ♣ Mob
      - If getStrengthPoints() > 8
        - Roll 3 6 sided die
        - Return sum of rolls

- Else if getStrengthPoints() > 4
    - Roll 2 6 sided die
    - Return sum of rolls
- Else
    - Roll 1 6 sided die return roll

- int getArmor()
    - ♣ Return armor
- void applyDamage (int Attack, int Armor, int Defense)
    - ♣ change = Attack – (Armor + Defense)
    - ♣ If change > 0
        - strength -= change
- int getStrengthPoints()
    - ♣ Return strength

- Harry Potter
    - Constructor
        - ♣ attack_Die_Sides = 6
        - ♣ attack_Die_Rolls = 2
        - ♣ defense_Die_Sides = 6
        - ♣ defense_Die _Rolls = 2
        - ♣ armor_Value = 0
        - ♣ strength_Points = 10
    - int getAttack()
        - ♣ Roll 2 6 sided Die

- ♣ Return sum of rolls

- int getDefense()
  - ♣ Roll 2 6 sided Die
  - ♣ Return sum of rolls
- int getArmor()
  - ♣ Return armor
- void applyDamage (int Attack, int Armor, int Defense)
  - ♣ change = Attack – (Armor + Defense)
  - ♣ If change > 0
    - strength -= change
  - ♣ If strength < 0
    - Strength = 20
- int getStrengthPoints()
  - ♣ Return strength

Begin Program

Loop Menu as long as they do not choose to exit fight!

Please Choose from the following options:

1. Would you like to initiate a Fight
2. Would you like to exit the program

Gets user input for menu choice

If option 2 { End program }

If options 1{

Please choose the first character you would like to see in battle:

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

Gets user input for first character

Please choose the character you would like to see in battle against (character 1):

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

Gets user input for second character

Initialize Characters!

Initialize a round counter (int Round = 0)

While(character 1 strength points > 0 && character 2 strength points > 0){

//character 1 attack

Character1->getAttack()

Character2->get defense

Character2->getArmor

Character2->applyDamage

Check to see if character 2 is dead before initiating there attack

{

```
            //character 2 attack
            Character2->getAttack()
            Character1->get defense
            Character1->getArmor
            Character1->applyDamage
        }
        Display round statistics(Attack, Defense, Resulting Strength Points of each)
    }
    Display Winner and final strength
}
End Program
```

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Ensure random values come from each attack roll function	None	Each Attack function	Random numbers that fall within the parameters of the die and number of rolls	Random numbers fell within the parameters
Ensure random values come from each defense roll function	None	Each Defense function	Random numbers that fall within the parameters of the die and number of rolls	Random numbers fell within the parameters
Ensure apply damage works correctly	Values derived from Attack, Defense, and Armor Functions	Each applyDamage function	math is correctly applied	math was correctly applied

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Medusa vs. Medusa	none	Character attack, armor, defense functions	should be a close finish most of the time. about 50/50 win ratio	close finishes. around 50/50 win ratio for each character
Medusa vs. Barbarian	none	Character attack, armor, defense functions	should be a close finish most of the time. about 50/50 win ratio	close finishes. around 50/50 win ratio for each character
Medusa vs. Vampire	none	Character attack, armor, defense functions	vampire should win most of the time	vampire won most of the time
Medusa vs. Blue Men	none	Character attack, armor, defense functions	Blue Men should win most of the time	Blue Men won most of the time
Medusa vs. Harry Potter	none	Character attack, armor, defense functions	Harry Potter should win most of the time	Harry Potter won most of the time
Barbarian vs. Barbarian	none	Character attack, armor, defense functions	should be a close finish most of the time. about 50/50 win ratio	close finishes. around 50/50 win ratio for each character
Barbarian vs. Vampire	none	Character attack, armor, defense functions	vampire should win most of the time	vampire won most of the time
Barbarian vs. Blue Men	none	Character attack, armor, defense functions	Blue Men should win most of the time	Blue Men won most of the time
Barbarian vs. Harry Potter	none	Character attack, armor, defense functions	Harry Potter should win most of the time	Harry Potter won most of the time
Vampire vs. Vampire	none	Character attack, armor, defense functions	should be a close finish most of the time. about 50/50 win ratio	close finishes. around 50/50 win ratio for each character
Vampire vs. Blue Men	none	Character attack, armor, defense functions	Blue Men should win most of the time	Blue Men won most of the time
Vampire vs. Harry Potter	none	Character attack, armor, defense functions	Harry Potter should win slightly more	Harry Potter won slightly more

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Blue Men vs. Blue Men	none	Character attack, armor, defense functions	should be a close finish most of the time. about 50/50 win ratio	close finishes. around 50/50 win ratio for each character
Blue Men vs. Harry Potter	none	Character attack, armor, defense functions	Blue Men should win most of the time	Blue Men won most of the time
Harry Potter vs. Harry Potter	none	Character attack, armor, defense functions	should be a close finish most of the time. about 50/50 win ratio	close finishes. around 50/50 win ratio for each character

Changes:

-I added a resurrected variable to tell me if harry potter had been resurrected or not.

-I also added a name variable and getName function so that I could output the name of the creature during testing and also thought it would be useful for the future assignment.

-I also added an if statement so that if char1 died in the attack by char2 then it cannot attack in that round since it died. Char2 attacks first so in a sense it does have an advantage.

Problems:

The only problem I ran into was figuring out how to resurrect Harry Potter only once which is why I created the resurrect variable in the classes. I think my pre planning really helped to limit my errors.

I did have a couple of mistype errors where I did not type the right name. Other than that, my tests seem to be consistent with what I expected.