Andrew P. Sturtevant
CS162-400 Project 3
26 July 2017

Class Requirements

Creature Class (base class)
Vampire(is a Creature)
Barbarian(is a Creature)
Blue Men (is a Creature)
Medusa (is a Creature)
HarryPotter (is a Creature)
grandFight

Creature Class Requirements

Data Members

Protected
Int attackNumberDie
Int attackDieSize
Int defenseNumberDie
Int defenseDieSize
Int armor
Int strengthPoint
String description
String getName
Int attackDamage
Int defenseDamage
Int damageDealt

Member Functions
Virtual Creature()
Virtual ~Creature()
Get & Set functions for all Data Members
Virtual Int attack
Virtual Int defend
Virtual reduceStrengthPoint(int)

Creature Class Design

Simple get & set functions for all data members

Int Creature::attack
    attackDamage = random integer between attackNumberDie and (attackNumberDie * attackDieSize)
    Return attackDamage

Int creature::defend
    defenseDamage = random integer between defenseNumberDie and (defenseNumberDie * attackDieSize)
    Return defenseDamage

reduceStrengthPoint(int damage)
    if(damage-armor > 0)
        damageDealt = damage - armor
        strengthPoint = strengthPoint - damageDealth

```
        return
    Else
        damageDealth = 0
        return
```

Vampire Class Requirements (is a Creature)

Member Functions
Vampire()
~Vampire()
Int defend

Vampire Class Design

```
Vampire()
    attackNumberDie = 1
    attackDieSize = 12
    defenseNumberDie = 1
    defenseDieSize = 6
    Armor = 1
    strengthPoints = 18
    Name = "Vampire"
    Description = "Suave, debonair, but vicious and surprisingly resilient!"

~Vampire(){}

Int defend()
    Random number between 0 and 1

    If 0
        Return random integer between defenseNumberDie and (defenseNumberDie *
attackDieSize)

    Else
        Output "The vampire charmed the enemy, the Enemy will not attack!"
        attackDamage = 100
        Return attackDamage
```

Barbarian Class Requirements

MemberFunctions
Barbarian()
~Barbarian()

Barbarian Class Design

Barbarian()
attackNumberDie = 2
attackDieSize = 6
defenseNumberDie = 2
defenseDieSize = 6
Armor = 0
strengthPoints = 18
Name = “Barbarian”
Description = “Think Conan or Hercules from the movies. Big sword, big muscles, bare torso.”

Blue_Men Class Requirements

Data member
Bool eightOrLess
Bool fourOrLess

Member Functions

Blue_Men()
~Blue_Men()
reduceStrengthPoint(int)

Blue_Men Class Design

Blue_Men()
attackNumberDie = 2
attackDieSize = 10
defenseNumberDie = 3
defenseDieSize = 6
Armor = 3
strengthPoints = 12
Name = “Blue Men”
Description = “They are small (6” tall), fast and tough. So they are hard to hit and can take some damage. As for the attack value, you can do a LOT of damage when you can crawl inside the armor or clothing of your opponent.”

eightOrLess = false
fourOrLess = false

~Blue_Men(){}

reduceStrengthPoint(int damage)
strengthPoint = strengthPoint - (damage - armor)

if(strengthPoint <= 8 && !eightOrLess)
defenseNumberDie --

eightOrLess = true

if(strenghPoint <=4 && !fourOrLess)
defenseNumberDie--
fourOrLess = true

Medusa Class Requirements

Medusa()
~Medusa()
Int attack()

Medusa Class Design

Medusa()
attackNumberDie = 2
attackDieSize = 6
defenseNumberDie = 1
defenseDieSize = 6
Armor = 3
strengthPoints = 8
Name = “Medusa”
Description = “Scrawny lady with snakes for hair. They help with fighting. Just don’t look at her!”

~Medusa(){}}

Int attack()
attackDamage = random integer between attackNumberDie and (attackNumberDie * attackDieSize)

if(attackDamage == 12)
Cout “The Medusa caught the enemies eyes with a Gaze and turned them to stone!”
attackDamage = 100
Return attackDamage

Else
Return attackDamage

Harry_Potter Class Requirements

Data member
Bool death

Member Functions

Harry_Potter()
~Harry_Potter()
reduceStrengthPoint(int)

Harry_Potter Class Design

Harry_Potter()
    attackNumberDie = 2
    attackDieSize = 6
    defenseNumberDie = 2
    defenseDieSize = 6
    Armor = 0
    strengthPoints = 10
    Name = “Harry Potter”
    Description = “Why are you reading this? How can you not know who Harry Potter is?”
    Death = false

~Harry_Potter(){}

reduceStrengthPoint(int damage)
    strengthPoint = strengthPoint - (damage - armor)

    if(strengthPoint <= 0 && !death)
        Cout “Harry Potter died, and has re-risen at Hogwarts with 20 strength points!”
        strengthPoint = 20

Grand_Fight Class Requirements

Data Members

Creature player1
Creature player2
Bool player1Win
Bool player2Win

Member Functions

createPlayers()
createCreature(int)
player1Attack()
player2Attack()
fullAttack()
winner()

```
Creature createCreature(int option)
    Creature tempCreature
    If (option == 1)
        tempCreature = new Vampire()
    if(option == 2)
        tempCreature = new Barbarian()
    if(option ==3)
        tempCreature = new Blue_Men()
    if(option == 4)
        tempCreature = new Medusa()
    if(option == 5)
        tempCreature = new Harry_Potter()
```

```
createPlayers()

    Cout “Select an Option for Player 1
    Player1 = createCreature(multiMenu())
    Cout “Select an Option for Player2”
```

```
        Player2 = createCreature(multiMenu())



player1Attack()

        Int attack = player1.attack()
        Int defense = player2.defense()
        Int total

        if(defense == 100)
                Return

        Else if(attack == 100)
                player1Win = true
                Return

        Else if(attack > defense)
                Total = attack - defense
                player2.reduceStrenthPoint(total)

                if(player2.getStrength() <= 0)
                        player1Win = true
                return
        Else
                return

player2Attack()

        Int attack = player2.attack()
        Int defense = player1.defense()
        Int total

        if(defense == 100)
                Return

        Else if(attack == 100)
                player2Win = true
                Return

        Else if(attack > defense)
```

```
            Total = attack - defense
            player1.reduceStrengthPoint(total)

            if(player1.getStrength() <=0)
                player2Win = true
            return
        Else
            return



fullAttack()
        Int playerFirst = random number between 1 and 2
        Int counter = 1

        While (!player1Win || !player2Win)

            Cout “Round #” << counter << “ results”
            if(playerFirst = 1)
                player1Attack()
                Cout << Player 1 Attack Damage: player1.getAttackDamage()
                Cout << Player 2 Defense: player2.getDefenseDamage()
                Cout << Player 1 Dealt << player2.damageDealt() << damage to Player 2
                if(!player1Win)
                    player2Attack()
                    Cout << Player 2 Attack Damage: player2.getAttackDamage()
                    Cout << Player 1 Defense: player1.getDefenseDamage()
                    Cout << Player 2 Dealt << player1.damageDealt() << damage to
Player 1

            Else if (playeFirst = 2)
                player2Attack()
                Cout << Player 2 Attack Damage: player2.getAttackDamage()
                Cout << Player 1 Defense: player1.getDefenseDamage()
                Cout << Player 2 Dealt << player1.damageDealt() << damage to

                if(!player2Win)
                    Cout << Player 1 Attack Damage: player1.getAttackDamage()
                    Cout << Player 2 Defense: player2.getDefenseDamage()
                    Cout << Player 1 Dealt << player2.damageDealt() << damage to
Player 2
                    player1Attack()
```

```
            counter++



winner()
    if(player1Win)
        Cout Player 1 is the winner!

    if(player2Win)
        Cout Player 2 is the winner!
```

Main

```
Grand_Fight theFight()

while(simpleMenu())
    Seed the random time
    theFight.createPlayers()
    theFight.fullAttack()
    theFight.winner()
```

Test Plan

Note: For any random features, the program was run multiple times until the the desired effect took place (i.e the Medusa's Gaze attack or Vampire's Charm). Input validation for this Project has been tested in previous assignments. Also, if a specific Creature is being tested for a particular function, the rest of that Creature is being tested as well. Each Creature was also tested against each of the other creatures, and fighting a Creature of the same type. Below is a table of some of the corner cases that were specifically tested for multiple times due to the randomness of the game.

Test Case	**Input Values**	**Driver Functions**	**Expected Outcomes**	**Observed Outcomes**
Vampire Charm in general	Vampire vs. Any other creature	Vampire.defend()	The vampire will prevent the other creature from attacking	As expected
Medusa Gaze	Medusa vs. Any other creature except Harry Potter	Medusa.attack()	The medusa will automatically win	As expected
Vampire Charm vs. Medusa Gaze	Vampire vs. Medusa	Vampire.defend() and Medusa.attack()	The Vampire Charm will "go off" before the Medusa's gaze, negating the Gaze attack	As expected
Medusa Gaze vs. Harry Potter's 1st life	Medusa vs. Harry Potter	Medusa.attack() and Harry Potter.reduceStrengthPoint()	Harry Potter will "die" and then come back with 20 SP	As expected
Blue Men loose Defense as mob gets smaller	Blue Men vs. Any other creature	Blue_Men.reduceStrengthPoint()	The die roll for Blue Men gets smaller as their SP goes down	As expected
Barbarian class	Barbarian vs. any other creature	Barbarian	All basic functions work	As expected
Harry Potter class	Harry Potter vs. any other creature	Harry Potter	All basic functions work, as does coming back to life with 20 SP	As expected

Reflection

For this assignment, I feel like I did a good job designing my classes. The biggest difference that happened in this is how my Fight class was designed. Instead of creating Creatures in one method and assigning them to players, this happens within the fullAttack() method. I was also able to consolidate the Fight.attack function into 1 instead of a different one for each player. With that change, my main() was also adjusted accordingly.