Benjamin Tate
2/14/16
CS 162 – Section 400
Assignment 3 Design and Reflection

Design:

Creature Base Class:
Protected data members: ints for the following: Number of sides on attack dice (attSides), number of attack dice (attCount), number of sides on defense dice (defSides), number of defense dice (defCount), attack and defense for the turn (attack and defense), armor, and strength.

Public member functions:

- Creature(int attSides, int attCount, int defSides, int defCount, int armor, int strength)
    - Initialize corresponding data members from parameter values.
- Virtual void attack() = 0
- Virtual void defend(int enemyAtt) = 0
- Int getAtt()
    - Return att
- Int getDef()
    - Return def + armor
- Int getStrength()
    - Return strength

Barbarian Class:
No new data members

Member functions:

- Barbarian() : Creature(6, 2, 6, 2, 0, 12)
    - All members are shared with Creature, so nothing is needed in this constructor.
- Virtual void attack()
    - Attack = 0
    - For(iterate up to attCount)
        - attack += random value from 1 to attSides
- Virtual void defend(int enemyAtt)
    - Int damage = enemyAtt - armor
    - Defense = 0
    - For(iterate up to defCount)
        - defense += random value from 1 to defSides
    - Damage -= defense
    - If(damage >= 0)
        - Strength -= damage

Medusa Class:

No new data members

Member functions:

- Medusa() : Creature(6, 2, 6, 1, 3, 8)
- Virtual void attack()
    - Attack = 0
    - For(iterate up to attCount)
        - attack += random value from 1 to attSides
    - if(attack == 12) ←Glare
        - attack = 50
- Virtual void defend(int enemyAtt) ←SAME AS BARBARIAN
    - Int damage = enemyAtt - armor
    - Defense = 0
    - For(iterate up to defCount)
        - defense += random value from 1 to defSides
    - Damage -= defense
    - If(damage >= 0)
        - Strength -= damage

<u>Vampire Class:</u>

Private data members: int charm (charm will be set randomly between 0 and 1 in defend(), if charm == 1, enemy will not attack)

Member functions:

- Vampire() : Creature(12, 1, 6, 1, 1, 18)
    - Charm will be set in defend(), so nothing is needed in this constructor.
- Virtual void attack() ←SAME AS BARBARIAN
    - Attack = 0
    - For(iterate up to attCount)
        - attack += random value from 1 to attSides
- Virtual void defend(int enemyAtt)
    - Defense = 0
    - Charm = random value from 0 to 1
    - If(charm == 0)
        - Int damage = enemyAtt - armor
        - For(iterate up to defCount)
            - defense += random value from 1 to defSides
        - Damage -= defense
        - If(damage >= 0)
            - Strength -= damage

<u>Potter Class:</u>

Private data members: int hogwarts

Member functions:

- Potter() : Creature(6, 2, 6, 2, 0, 10)
    - Initialize hogwarts to 1
- Virtual void attack() ←SAME AS BARBARIAN
    - Attack = 0
    - For(iterate up to attCount)
        - attack += random value from 1 to attSides
- Virtual void defend(int enemyAtt)
    - Defense = 0
    - Int damage = enemyAtt - armor
    - For(iterate up to defCount)
        - defense += random value from 1 to defSides
    - Damage -= defense
    - If(damage >= 0)
        - Strength -= damage
    - If(strength <= 0 and Hogwarts > 0)
        - Strength = 20
        - Hogwarts -= 1

<u>BlueMen Class:</u>
No new data members

Member functions:

- BlueMen() : Creature(10, 2, 6, 3, 3, 12)
    - All members are shared with Creature, so nothing is needed in this constructor.
- Virtual void attack()
    - Attack = 0
    - For(iterate up to attCount)
        - attack += random value from 1 to attSides
- Virtual void defend(int enemyAtt)
    - Int damage = enemyAtt - armor
    - Defense = 0
    - For(iterate up to defCount)
        - Defense += random value from 1 to defSides
    - Damage -= defense
    - If(damage >= 0)
        - Strength -= damage
    - If(strength <= 4)
        - defCount = 1
    - else if(strength <= 8)
        - defCount = 2

Testing Plan:

I will be creating the classes one at a time, testing each new one with every class that I have made (including itself) before creating the next. To illustrate this, I will be following this sequence:

- Create Barbarian Class
  - Test Barbarian v. Barbarian
- Create Medusa Class
  - Test Barbarian v. Medusa
  - Test Medusa v. Medusa
- Create Vampire Class
  - Test Barbarian v. Vampire
  - Test Medusa v. Vampire
  - Test Vampire v. Vampire
- Create Potter Class
  - Test Barbarian v. Potter
  - Test Medusa v. Potter
  - Test Vampire v. Potter
  - Test Potter v. Potter
- Create BlueMen Class
  - Test Barbarian v. BlueMen
  - Test Medusa v. BlueMen
  - Test Vampire v. BlueMen
  - Test Potter v. BlueMen
  - Test BlueMen v. BlueMen

During test battles, my main function will print each combatant's attack and defense rolls, as well as their remaining strength at the end of the attack. I will place print statements within the relevant function of each class to show when the class' special ability is being used as the battle unfolds. At the end of each test battle, I will review each turn to make sure that every roll is within the possible range, that strength is being updated properly, and that special abilities are being used at the correct times and in the correct way. If any results seem odd (rolls don't seem random, etc.) I will redo that test battle.

Reflection:

As it happened, my design plan pretty much worked perfectly (for once). The only problems that I had were with syntax (which I simply addressed error by error). The only thing that I really had to change from my design plan was the name of the int attack (because there was a function named the same thing in the Creature class), so I changed it to att, and I went ahead and changed defense to def for consistency.

Testing went very well too. There was one test of Barbarian v. Barbarian that didn't really seem random, but I tested a couple more times and went back to my code to be sure that it was. The only class that gave me problems was BlueMen. For a while, I couldn't figure out what the heck was going wrong with them, but then I realized that I had just forgotten to update the initial values in the default constructor after I had copied the line over from the Barbarian class. Once I changed those numbers, it worked fine.