William Kim
kimw@oregonstate.edu
CS162
Assign 4 Reflection
11/22/15

Design:

For assignment 4 I reused my Assignment 3 class hierarchy that uses the parent Creature class and five subclasses (Barbarian, BlueMen, Goblin, ReptilePeople, and TheShadow). It correctly uses polymorphism to inherit traits. I also added a new regenerate function to Creature so that all of the subclasses inherit the same. I allowed the characters that win to regenerate 4 strength points every time. The Goblin class I overloaded the regenerate method to give this character a fighting chance by regenerating its health to full. Then I overloaded the Reptile People to give it a disadvantage by only regenerating 2 strength points. Then I used my lab6 queue to hold the team line up and the loser pile. I tweaked it to hold a fighter struct that holds the name string, creatureType string, Creature object, and a pointer to the next fighter struct. I also created two adding functions, an add that creates a new fighter at the back of the queue and an addBack that takes a fighter object and adds it to the queue. The remove function just moves the front pointer to the next object. The Game class initializes 3 queues, 2 for the lineup and 1 for the loser lineup. Also the game initializes 4 variables, numOfCreatures is to hold the users input of how many fighters for the tournament. The p1lose, p2lose, and roundCount are just counters to keep track of the game. Also I created a struct to hold first, second, and third place winner. The intro function starts the game and gets the number of fighters. The lineup function gets the lineup wanted from the user for each player. The battle function determines how the game is played and outputs all the needed information. The printWinner function determines the winner who loses less rounds and outputs the winner. The printFinal function will print out the first second and third place winners, I only allowed fighters who won to be on the list. The printLoser function loops the loser queue and outputs the list.

Testing: For testing I ran many scenarios to test each creature and then when I scaled up the teams I just tested once to make sure everything was building correctly.

1v1 testing –

Barbarian vs Blue Men = Blue Men won and placed in first. Barbarian placed in loser. Player 2 wins.

Goblin vs Reptile People = Reptile People won and placed in first. Goblin placed in loser. Player 2 wins.

The Shadow vs Barbarian = The Shadow won and placed in first. Barbarian placed in loser. Player 1 wins.

The Shadow vs Blue Men = Blue Men won and placed in first. The Shadow placed in loser. Player 2 wins.

Goblin vs Barbarian = Barbarian won and placed in first. Goblin placed in loser. Player 2 wins.

After testing several 1 versus 1, the program looks to be running according to plan and the subclass polymorphism is working correctly as some characters are winning over others.

2 v 2 testing –

Barb & Goblin V Blue Men & The Shadow = this game went two rounds because players 2’s characters are stronger or have better defense. The Blue men beat the Barbarian and placed in first. Then the Shadow defeated the goblin and placed in second. Losers contained Barbarian first and Goblin second. Player 2 wins the tournament

Blue Men & Reptile People V The Shadow & Barbarian = this game went three rounds. Blue fought the shadow and won. Then Reptile fought barb and lost. Then Blue fought the barb and won. First place was Blue and second place was barb. The game losers are Shadow, Reptile, and Barb. Player 1 wins the tournament.

After testing the 2v2 the outcomes still look good, nothing out of the ordinary and the placement of winners and losers as well has first and second place is good.

3 v 3 testing –

Round - 1
Player 1 Fighter - Name: 1; Creature: Barbarian
Player 2 Fighter - Name: 3; Creature: Goblin
Player 1 wins

Round - 2
Player 1 Fighter - Name: 2; Creature: Blue Men
Player 2 Fighter - Name: 4; Creature: Reptile People
Player 1 wins

Round - 3
Player 1 Fighter - Name: 3; Creature: Goblin
Player 2 Fighter - Name: 5; Creature: Shadow
Player 2 wins

Round - 4
Player 1 Fighter - Name: 1; Creature: Barbarian
Player 2 Fighter - Name: 5; Creature: Shadow
Player 2 wins

Round - 5
Player 1 Fighter - Name: 2; Creature: Blue Men
Player 2 Fighter - Name: 5; Creature: Shadow

```
Player 1 wins

FIRST PLACE - name: 1 team: 1 creature: Barbarian
SECOND PLACE - name: 2 team: 1 creature: Blue Men
THIRD PLACE - name: 5 team: 2 creature: Shadow


GAME LOSERS
Name: 3 Creature: Goblin
Name: 4 Creature: Reptile People
Name: 3 Creature: Goblin
Name: 1 Creature: Barbarian
Name: 5 Creature: Shadow


Player 1 Wins the Tournament!
```

The 3v3 results look good and according to plan.

4v4 testing –

```
Round - 1
Player 1 Fighter - Name: 1; Creature: Barbarian
Player 2 Fighter - Name: 2; Creature: Goblin
Player 1 wins

Round - 2
Player 1 Fighter - Name: 2; Creature: Goblin
Player 2 Fighter - Name: 3; Creature: Reptile People
Player 2 wins

Round - 3
Player 1 Fighter - Name: 3; Creature: Shadow
Player 2 Fighter - Name: 4; Creature: Shadow
Player 2 wins

Round - 4
Player 1 Fighter - Name: 1; Creature: Blue Men
Player 2 Fighter - Name: 1; Creature: Barbarian
Player 1 wins

Round - 5
Player 1 Fighter - Name: 1; Creature: Barbarian
Player 2 Fighter - Name: 3; Creature: Reptile People
Player 1 wins

Round - 6
Player 1 Fighter - Name: 1; Creature: Blue Men
Player 2 Fighter - Name: 4; Creature: Shadow
```

Player 1 wins

FIRST PLACE - name: 1 team: 1 creature: Barbarian
SECOND PLACE - name: 3 team: 2 creature: Reptile People
THIRD PLACE - name: 4 team: 2 creature: Shadow

GAME LOSERS
Name: 2 Creature: Goblin
Name: 2 Creature: Goblin
Name: 3 Creature: Shadow
Name: 1 Creature: Barbarian
Name: 3 Creature: Reptile People
Name: 4 Creature: Shadow

Player 1 Wins the Tournament!

Again the 4v4 results look good another nothing looks out of the norm.

5v5 testing – Round - 1
Player 1 Fighter - Name: 1; Creature: Barbarian
Player 2 Fighter - Name: 1; Creature: Shadow
Player 2 wins

Round - 2
Player 1 Fighter - Name: 2; Creature: Blue Men
Player 2 Fighter - Name: 2; Creature: Reptile People
Player 1 wins

Round - 3
Player 1 Fighter - Name: 3; Creature: Goblin
Player 2 Fighter - Name: 3; Creature: Goblin
Player 1 wins

Round - 4
Player 1 Fighter - Name: 4; Creature: Reptile People
Player 2 Fighter - Name: 4; Creature: Blue Men
Player 2 wins

Round - 5
Player 1 Fighter - Name: 5; Creature: Shadow
Player 2 Fighter - Name: 5; Creature: Barbarian
Player 1 wins

Round - 6
Player 1 Fighter - Name: 2; Creature: Blue Men
Player 2 Fighter - Name: 1; Creature: Shadow
Player 1 wins

Round - 7

```
Player 1 Fighter - Name: 3; Creature: Goblin
Player 2 Fighter - Name: 4; Creature: Blue Men
Player 2 wins

Round - 8
Player 1 Fighter - Name: 5; Creature: Shadow
Player 2 Fighter - Name: 4; Creature: Blue Men
Player 2 wins

Round - 9
Player 1 Fighter - Name: 2; Creature: Blue Men
Player 2 Fighter - Name: 4; Creature: Blue Men
Player 1 wins

FIRST PLACE - name: 1 team: 2 creature: Shadow
SECOND PLACE - name: 2 team: 1 creature: Blue Men
THIRD PLACE - name: 3 team: 1 creature: Goblin

GAME LOSERS
Name: 1 Creature: Barbarian
Name: 2 Creature: Reptile People
Name: 3 Creature: Goblin
Name: 4 Creature: Reptile People
Name: 5 Creature: Barbarian
Name: 1 Creature: Shadow
Name: 3 Creature: Goblin
Name: 5 Creature: Shadow
Name: 4 Creature: Blue Me

Player 1 Wins the Tournament!
```

5v5 ran as expected so overall I feel that my program works according to plan.

Reflection:

The overall original design of mine worked as planned. The major issues I ran into while programming this assignment was the memory leaks in the queue. When I first started testing I ran into issues of memory leaks so I had to narrow in on my Queue class. I knew that my queue should add new queues and when I placed the front pointer queue in my addBack method, I should be able to addBack my front queue how ever many times I would like. When I created a queue object q and called q.add("1") … q.add("3), I then called (10) q.addBack(q->front). It returning exit error 9 because of a memory leak. I pinpointed the issue down to a line of code where I was creating new struct objects to hold the front pointer and not deleting it correctly. Also my back pointers were not point correctly because I was deleting the object after pointing it. I realized that I needed to not delete the queue object but move the pointers around.

Then I ran into a few issues in the battle function with my game loop. I resolved this by making sure the game ended when it needed by ended the loop when ever either player 1 or player 2 lineups is equal to the numOfCreatures.