DESIGN AND REFLECTION

Cassidy Bullock. CS 162 Online Section

DESIGN

Classes:

Character: Base Class

Functions: public

- Attack(): pure virtual function
- Defense(): pure virtual function
- updateStrenPts(): updates how many strength points a player has left
- checkDead(): virtual function, checks if strength points have depleted
- printStatus(): pure virtual function

Data: Protected

- int Number of attack dice
- int attack dice size
- int number of defense dice
- int defense dice size
- int armor
- int strength points

Reptile: Derived Class

Functions:

- Reptile(): constructor to assign dice values, armor, and initial strength points
- Attack(): rolls attack dice with modded attack dice size, returns total amount of damage as an int
- Defense(int): takes damage passed from opponent and rolls attack dice and subtracts defense and armor from damage
- printStatus(): prints the current level of strength points

Data: N/A

Medusa: Derived Class

Functions: Public

- Medusa(): constructor to assign dice values, armor, and initial strength points
- Attack(): rolls attack dice with modded attack dice size, returns total amount of damage as an int, if a 12 is rolled returns a specific number which Main will check for when it runs, which will then activate Glare Power
- Defense(int): takes damage passed from opponent and rolls attack dice and subtracts defense and armor from damage
- printStatus(): prints the current level of strength points

Data: N/A

Gollum: Derived Class

Functions: Public

- Gollum(): constructor to assign dice values, armor, and initial strength points
- Attack(): rolls attack dice with modded attack dice size, returns total amount of damage as an int, has a 5% chance of rolling 3 dice instead of one, this is the Ring Power, have a ring int, modded by 20 and then if 0 comes up this is when the ring power is activated.
- Defense(int): takes damage passed from opponent and rolls attack dice and subtracts defense and armor from damage
- printStatus(): prints the current level of strength points

Data: N/A

Potter: Derived Class

Functions: Public

- Potter(): constructor to assign dice values, armor, and initial strength points, as well as the flag for whether Harry has died yet or not
- Attack(): rolls attack dice with modded attack dice size, returns total amount of damage as an int
- Defense(int): takes damage passed from opponent and rolls attack dice and subtracts defense and armor from damage
- printStatus(): prints the current level of strength points
- checkDead(): checks if strength points has depleted and if it has and the flag is false then it will return strength points to 10 and change the flag to 1, if strength points are depleted for the second time will return true, and if not will return false

Data: Protected

- bool life_flag: keeps track if character has died yet

Blue: Derived Class

Functions: public

- Blue(): constructor to assign dice values, armor, and initial strength points
- Attack(): rolls attack dice with modded attack dice size, returns total amount of damage as an int, has a 5% chance of rolling 3 dice instead of one, this is the Ring Power, have a ring int, modded by 20 and then if 0 comes up this is when the ring power is activated.
- Defense(int): takes damage passed from opponent and rolls attack dice and subtracts defense and armor from damage
- printStatus(): prints the current level of strength points and the number of defense dice. Checks if strength points have gone down 4 or not, and if they have takes off one defense die.

Data: N/A

Main:

Functions:

- printOptions(): prints character options for the players to choose from
- getChoice(): grabs the choice of character for each player and returns it
- run(Character* player1, Character* player2, int hits, int damage): conducts rounds of the players, calls attack function for first player and passes what is returned to the defense function to the other player, then updates the amount of strength points left, then it checks if the defensive player has died. Each round does this for both players, if one player dies then it breaks, at the end of the round it prints the status for the user to see where the characters are at in terms of strength.
- createGame(int choice1, int choice2, Character *player1… (for each instance of character created), int hits, int damage): looks at the choices picked by player one and two and passes the correct character objects into the run function.

Int main():

- seed random to time
- create all the objects of characters
- declare the choices, hits and damage int,
- call the getChoice function for each player
- call create game

REFLECTIONS

Testing:

Test Case	Input Values	Expected Outcome	Actual Outcome	Comments
Medusa vs Medusa	1, 1	One of the Medusas will win, possibly the Glare Power will be activated	Medusa won, Glare activated and worked as expected	works as expected
Medusa vs Gollum	1,2	Medusa will probably win, Glare power or Ring power may be activated	Medusa won, Ring power got activated and worked as expected	works as expected
Medusa vs Reptile People	1, 3	Reptile people will probably win, Glare power may be activated	Reptile people won,	works as expected
Medusa vs Blue Men	1, 4	Medusa will probably win, Glare power may be activated, Mob power will probably come into play	Blue Men won	works as expected
Medusa vs Harry Potter	1, 5	Harry potter will probably win, Glare power or Hogwarts power may be activated	Harry potter won, Hogwarts power activated	works as expected
Gollum vs Gollum	2, 2	One of the gollums will win, Ring power may be activated	Gollum won, ring power activated	works as expected
Gollum vs Reptile People	2, 3	Reptile people will probably win, ring power may be activated	reptile people won	works as expected
Gollum vs Blue Men	2, 4	blue men will probably win, ring power or mob	blue people won	works as expected

		power may be activated		
Gollum vs Harry Potter	2, 5	harry potter will probably win, ring power or Hogwarts power may be activated	Harry Potter won	works as expected
Reptile People vs Reptile People	3, 3	one of the reptile people will win	Reptile people won	works as expected
Reptile People vs Blue Men	3, 4	reptile people will probably win, mob power may be activated	Blue men won	works as expected
Reptile People vs Harry Potter	3, 5	harry potter will probably win, Hogwarts power may be activated	Reptile people won, Hogwarts power activated	works as expected
Blue Men vs Blue Men	4, 4	one of the blue men will win, mob power will definitely be activated	Blue men won, Mob power activated	works as expected
Blue Men vs Harry Potter	4, 5	harry potter will probably win, mob power or Hogwarts power may be activated	Blue men won, Hogwarts power activated	works as expected
Harry Potter vs Harry Potter	5, 5	one of the harry potters will win, Hogwarts power will definitely be activated	Harry Potter won, Hogwarts power activated twice	works as expected

General Other Thoughts:

Everything works as expected. Each variation of the functions was tested for the different characters and they all appear to work. I originally tried to only do declarations based off of the user choice but I found that separating out declarations like that created syntax errors. I decided to just do all of the declarations in main and then create a function to pass the correct objects based on player choice into the function that runs the program.

I didn't run into any problems with the special abilities, I only had to make sure my syntax with polymorphism and the functions calls were correct.

To Use the Makefile:

Type **make -f makefile.txt** into the command line