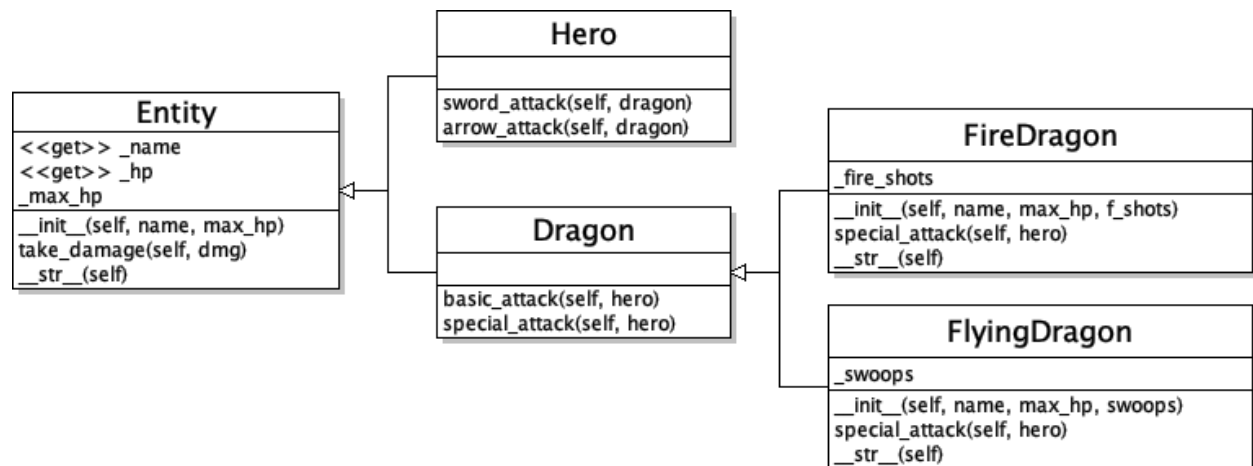


CECS 277 – Lab 8 – Polymorphism

Dragon Trainer

Create a game where the user must defeat three dragons to pass the trails. Use inheritance to implement the following class diagram in your program.



Entity Class (entity.py) –

1. `__init__(self, name, max_hp)` – set the `_name`, `_max_hp`, and `_hp`. Assign the `max_hp` value to both the `_max_hp` and `_hp` attributes.
2. `name` and `hp` properties – use decorators to get (not set) the values of `_name` and `_hp`.
3. `take_damage(self, dmg)` – the damage the entity takes. Subtract the `dmg` value from the entity's `_hp`. Do not let the `hp` go past 0 (if it's a negative value, reset it to 0).
4. `__str__(self)` – display the entity's name and `hp` in the format "Name: hp/max_hp".

Hero Class (hero.py) –

1. `sword_attack(self, dragon)` – the dragon takes a random amount of damage in the range 2D6 (1-6 + 1-6). Return a string with the description of the attack and the damage dealt to the dragon.
2. `arrow_attack(self, dragon)` – the dragon takes a random amount of damage in the range 1D12 (1-12). Return a string with the description of the attack and the damage dealt to the dragon.

Dragon Class (dragon.py) –

1. `basic_attack(self, hero)` – tail attack – the hero takes a random amount of damage in the range 3-7. Return a string with the description of the attack and the damage dealt to the hero.
2. `special_attack(self, hero)` – claw attack – the hero takes a random amount of damage in the range 4-8. Return a string with the description of the attack and the damage dealt to the hero.

FireDragon Class (fire_dragon.py) –

1. `__init__(self, name, max_hp, f_shots)` – call `super` to set the name and `hp`, then set the number of `fire_shots`.
2. `special_attack(self, hero)` – overridden fire attack – if the dragon has any `fire_shots` left, then the hero takes a random amount of damage in the range 5-9 and the number of

fire_shots is decremented and a string with the description of the attack and the damage dealt to the hero is returned. Otherwise, no damage is dealt and a string with the description of the failure is returned.

3. `__str__(self)` – use super to get the `__str__` from the entity class, then concatenate on the number of fire_shots.

FlyingDragon Class (flying_dragon.py) –

1. `__init__(self, name, max_hp, swoops)` – call super to set the name and hp, then set the number of swoops.
2. `special_attack(self, hero)` – overridden swoop attack – if the dragon has any swoops left, then the hero takes a random amount of damage in the range 5-8 and the number of swoops is decremented and a string with the description of the attack and the damage dealt to the hero is returned. Otherwise, no damage is dealt and a string with the description of the failure is returned.
3. `__str__(self)` – use super to get the `__str__` from the entity class, then concatenate on the number of swoops.

Main (main.py) – Construct a Hero object and then create a list that contains one of each of the dragons (Dragon, FireDragon, FlyingDragon). Present a menu that allows the user to choose which dragon to attack, and then another menu that gives them the option of attacking with a sword or an arrow. Call the hero's attack method on the dragon they chose and display the attack message returned. If the user defeats the dragon (hp is 0), then remove that dragon from the list. Then choose a random (surviving) dragon that will attack the user, and randomly choose either a basic or special attack and display the attack message returned. Repeat the attacks until the user defeats all three dragons, or until the hero is knocked out. Check all input for validity.

Example Output (user input is in italics):

What is your name, challenger?
Astrid
Welcome to dragon training, Astrid
You must defeat 3 dragons.

Astrid: 50/50
1. Attack Deadly Nadder: 10/10
2. Attack Gronckle: 15/15
Fire Shots remaining: 3
3. Attack Timberjack: 20/20
Swoop attacks remaining: 5
Choose a dragon to attack: 3

Attack with:
1. Arrow (1 D12)
2. Sword (2 D6)
Enter weapon: 2

You slash the Timberjack with your sword for 9 damage.
Gronckle smashes you with its tail for 7 damage!

Astrid: 43/50
1. Attack Deadly Nadder: 10/10

2. Attack Gronckle: 15/15
Fire Shots remaining: 3
3. Attack Timberjack: 11/20
Swoop attacks remaining: 5
Choose a dragon to attack: 3

Attack with:
1. Arrow (1 D12)
2. Sword (2 D6)
Enter weapon: 1

You hit the Timberjack with an arrow for 9 damage.
Gronckle engulfs you in flames for 7 damage!

Astrid: 36/50
1. Attack Deadly Nadder: 10/10
2. Attack Gronckle: 15/15
Fire Shots remaining: 2
3. Attack Timberjack: 2/20
Swoop attacks remaining: 5
Choose a dragon to attack: 3

Attack with:

1. Arrow (1 D12)
2. Sword (2 D6)
Enter weapon: 2

You slash the Timberjack with your sword for 5 damage.
Gronckle engulfs you in flames for 5 damage!

Astrid: 31/50
1. Attack Deadly Nadder: 10/10
2. Attack Gronckle: 15/15
Fire Shots remaining: 1
Choose a dragon to attack: 3
Invalid input - should be within range (1-2).
Choose a dragon to attack: 2

Attack with:
1. Arrow (1 D12)
2. Sword (2 D6)
Enter weapon: 1

You hit the Gronckle with an arrow for 9 damage.
Deadly Nadder slashes you with its claws for 8 damage!

Astrid: 23/50
1. Attack Deadly Nadder: 10/10

2. Attack Gronckle: 6/15
Fire Shots remaining: 1
Choose a dragon to attack: 1

Attack with:
1. Arrow (1 D12)
2. Sword (2 D6)
Enter weapon: 2

You slash the Deadly Nadder with your sword for 11 damage.
Gronckle smashes you with its tail for 5 damage!

Astrid: 18/50
1. Attack Gronckle: 6/15
Fire Shots remaining: 1
Choose a dragon to attack: 1

Attack with:
1. Arrow (1 D12)
2. Sword (2 D6)
Enter weapon: 2

You slash the Gronckle with your sword for 9 damage.

Congratulations! You have defeated all 3 dragons, you have passed the trials.

Notes:

1. You should have 7 different files: entity.py, hero.py, dragon.py, fire_dragon.py, flying_dragon.py, main.py, and check_input.py.
2. Check all user input using the get_int_range function in the check_input module.
3. Do not create any extra methods or functions or add any extra parameters.
4. Please do not create any global variables or use the attributes globally (ie. do not access any of them using the underscore). You can access the name and hp using the read-only properties, but you shouldn't access the max_hp, fire_shots, or swoops outside of their own classes.
5. Use docstrings to document each of the classes, their attributes, and each of their methods. See the lecture notes for examples.
6. Place your names, date, and a brief description of your program in a comment block at the top of your program. Place brief comments throughout your code.
7. Thoroughly test your program before submitting:
 - a. Make sure that your classes are inherited properly. Hero and Dragon should inherit from Entity, and Fire and Flying Dragons should inherit from Dragon.
 - b. Make sure user input is validated. Including when a dragon is defeated and removed from the list, that dragon should no longer be selectable.
 - c. Make sure that the damage dealt is correctly subtracted from the opponent.
 - d. Make sure that the randomly selected dragon is alive.

- e. Make sure that the flying/fire dragons cannot do their special attack once they run out of their respective charges (and no damage is dealt).
- f. Make sure the game ends when the user defeats all 3 dragons, or when the hero runs out of hp.