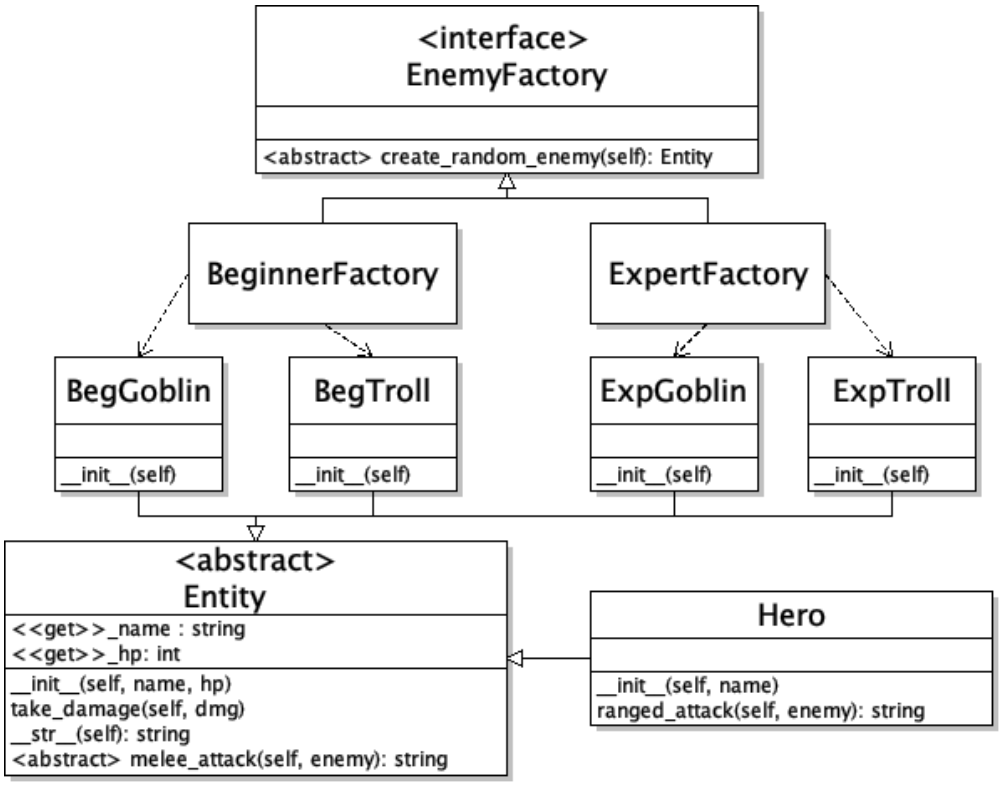
# CECS 277 – Lab 10 – Factory Method

## Monster Trials

Create a game where the user must defeat three monsters to pass the trials. Use the following UML diagram and the method descriptions below to create your program.




Classes:

1. Entity (entity.py) – abstract class that the monsters and the hero extend from.
    a. __init__(self, name, hp) – sets the name and hp attributes.
    b. name and hp properties – use decorators to get (not set) the values of _name and _hp.
    c. take_damage(self, dmg) – deals 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 negative, reset it back to 0).
    d. __str__(self) – return a string with the entity's name and hp.
    e. melee_attack(self, enemy) – abstract – the attack the entity does to another entity.
2. Hero (hero.py) – the user's character, extends from Entity.
    a. __init__(self, name) – passes the name and default hp to the superclass's init.
    b. melee_attack(self, enemy) – deals 2D6 (the sum of two 6-sided dice) damage to the enemy and returns a string description of the attack.
    c. ranged_attack(self, enemy) – deals 1D12 (one 12-sided die) damage to the enemy and returns a string description of the attack.
3. EnemyFactory (enemy_factory.py) – interface – template for all enemy factories.
    a. create_random_enemy(self) – abstract method (no code) that each concrete factory overrides to create and return enemy objects.

4. BeginnerFactory (beg_factory.py) – creates easy enemies, extends from EnemyFactory.
    a. create_random_enemy(self) – randomly construct and return one of the beginner enemies (BegGoblin or BegTroll).
5. ExpertFacory (exp_factory.py) – creates difficult enemies, extends from EnemyFactory.
    a. create_random_enemy(self) – randomly construct and return one of the expert enemies (ExpGoblin or ExpTroll).
6. BegGoblin (beg_goblin.py), BegTroll (beg_troll.py), ExpGoblin (exp_goblin.py), ExpTroll (exp_troll.py) – extend from Entity – the types of monsters that the factories construct.
    a. __init__(self) – using super, give each monster a default name and randomize its hp based on the table below. (Note: give the difficult enemies a scarier name so I can tell that the correct factory was used (ex. "Angry Troll" or "Horrible Goblin")).
    b. melee_attack(self, enemy) – randomize the damage based on the table below, deal the damage to the enemy (the hero), and return a string describing the attack.

| **Enemy** | **Goblin** | **Troll** |
|---|---|---|
| **Easy** | HP: 7-9, Dmg: 4-6 | HP: 8-10, Dmg: 5-9 |
| **Difficult** | HP: 12-15, Dmg: 5-8 | HP: 15-18, Dmg: 8-12 |

Main – prompt the user to enter their name, then construct a hero, and the factories. Use the factories to generate a list of three monsters, two beginners and an expert, that the user will fight. Create a loop that repeats until the hero dies, or until the monsters are defeated. Have the user choose a monster to fight and the type of attack. The hero will attack the selected monster with the user's choice of attack and the resulting string will be displayed. If the monster is still alive, it will attack the hero back. Display the result of the monster's attack. If the monster is slain, then remove it from the list of monsters.

**Example Output:**

```
Monster Trials
What is your name? Link

You will face a series of 3
monsters, Link.
Defeat them all to win.

Choose an enemy to attack:
1. Troll HP: 9
2. Goblin HP: 8
3. Angry Goblin HP: 13
Enter choice: 1

Link HP: 25
1. Sword Attack
2. Arrow Attack
Enter choice: 2

Link pierces a Troll with an arrow
for 11 damage.
You have slain the Troll

Choose an enemy to attack:
1. Goblin HP: 8
2. Angry Goblin HP: 13
Enter choice: 1

Link HP: 25
1. Sword Attack
2. Arrow Attack
Enter choice: 1

Link slashes a Goblin with a sword
for 3 damage.
Goblin bites Link for 6 damage.

Choose an enemy to attack:
1. Goblin HP: 5
2. Angry Goblin HP: 7
Enter choice: 1

Link HP: 19
1. Sword Attack
2. Arrow Attack
Enter choice: 1
```

```
Link slashes a Goblin with a sword
for 9 damage.
You have slain the Goblin

Choose an enemy to attack:
1. Angry Goblin HP: 13
Enter choice: 1

Link HP: 19
1. Sword Attack
2. Arrow Attack
Enter choice: 2

Link pierces a Angry Goblin with an
arrow for 5 damage.
Angry Goblin slams Link for 6
damage.

Choose an enemy to attack:
1. Angry Goblin HP: 8
Enter choice: 1

Link HP: 13
1. Sword Attack
2. Arrow Attack
Enter choice: 1

Link slashes a Angry Goblin with a
sword for 10 damage.
You have slain the Angry Goblin

Congratulations! You defeated all
three monsters!
Game Over
```

**Notes:**

1. You should have 10 different files: enemy_factory.py, beg_factory.py, exp_factory.py, entity.py, hero.py, beg_troll.py, beg_goblin.py, exp_troll.py, exp_goblin.py, main.py.
2. Place your names, date, and a brief description of the program in a comment block at the top of your main file. Place brief comments throughout your code.
3. Use docstrings to document each of the classes, their attributes, and their methods.
4. Please do not create any global variables or use attributes globally (ie. do not access any of the attributes using the underscores).
5. Do not create any extra methods, attributes, parameters, or change the class hierarchy.
6. Check all user input using the get_int_range function in the check_input module.
7. You may modify the starting hp of the monsters and the hero. You may also modify the random damage ranges of the monsters.
8. Thoroughly test your program before submitting:
    a. Make sure that your class hierarchy is correct: abstract classes are abstract and have abstract methods using the @abc.abstractmethod decorator, and the subclasses extend from the correct superclasses (based on the UML above).
    b. Make sure that you have the correct number of each type of enemy.
    c. Make sure that the opposing enemy takes the correct amount of damage when hit.
    d. Make sure that the monsters are removed from the list when they are defeated.
    e. Make sure the game ends when the user runs out of hp or when all three monsters are defeated.

**Monster Trials Rubric – Time estimate: 4 hours**

| **Monster Trials**<br>**10 points** | Correct.<br><br>2 points | A minor mistake.<br>1.5 points | A few mistakes.<br>1 point | Several mistakes.<br>0.5 points | No attempt.<br>0 points |
|---|---|---|---|---|---|
| **Factory classes** (in separate files)**:**<br>1. EnemyFactory is abstract w/ abstract create_random_enemy method.<br>2. Beginner and Expert Factories extend from EnemyFactory.<br>3. BeginnerFactory randomly constructs beginner trolls and goblins.<br>4. ExpertFactory randomly constructs expert trolls and goblins. | | | | | |
| **Entity and Hero classes** (sep files)**:**<br>1. Entity is abstract with abstract attack method, has properties for name & hp.<br>2. Hero class extends Entity.<br>3. Hero's init sets name and default hp.<br>4. Hero overrides melee attack (2D6), and has ranged attack (1D12). | | | | | |
| **Enemy classes** (separate file)**:**<br>1. Enemy classes extend from Entity.<br>2. init sets name and random hp.<br>3. Overrides melee attack with random damage and returns attack string. | | | | | |
| **Main file** (separate file)**:**<br>1. Constructs factories and hero.<br>2. Uses factories to create list of enemies (2 beginner, 1 expert).<br>3. Repeats until user dies or destroys all three enemies.<br>4. Prompts user to attack enemy and type of attack & displays attack string.<br>5. Enemy attacks back if still alive.<br>6. Removes dead enemies from list.<br>7. Error checks user input. | | | | | |
| **Code Formatting:**<br>1. Correct spacing.<br>2. Meaningful variable names.<br>3. No exceptions thrown.<br>4. No global variables or using attributes directly.<br>5. Correct documentation. | | | | | |