```
# homework 20 solutions / classwork

import random

class item:

    def __init__(self, condition, rarity):
        self.cond = condition
        self.rar = rarity
        self.value = 0
        # two options:
        # options 1:
        #self.calc_value_slow

        # option 2:
        #we can just use self.rar_coef for calculations
        self.rar_coef = 0
        self.calc_rar_coef()
        self.calc_value_fast()

        # Question for discussion: which option is better?

    def calc_value_slow(self):
        if self.rar == "common":
            self.value = self.cond * 1
        elif self.rar == "rare":
            self.value = self.cond * 2
        else:
            self.value = self.cond * 3

    def calc_value_fast(self):
        self.value = self.cond * self.rar_coef

    def calc_rar_coef(self):
        if self.rar == "common":
            self.rar_coef = 1
        elif self.rar == "rare":
            self.rar_coef = 2
        else:
            self.rar_coef = 3

    def upgrade(self):
        if self.rar == "common":
            self.rar = "rare"
        if self.rar == "rare":
            self.rar = "epic"
        self.calc_rar_coef()
```

```python
        self.calc_value_fast()

    def display(self):
        print(f"The {self.__class__.__name__} ({self.cond}) is {self.rar}.")

class sword(item):
    base_attack = 10

    def __init__(self, c, r):
        super().__init__(c, r)
        self.attack = 0
        self.calc_attack()

    def calc_attack(self):
        self.attack = self.base_attack * self.rar_coef * (self.cond/100)

    def display(self):
        print(f"The {self.__class__.__name__} ({self.cond}) is {self.rar}.")
        print(f"Its attack value is {self.attack}.")

class shield(item):
    base_defense = 8

    def __init__(self, c, r):
        super().__init__(c, r)
        self.defense = 0
        self.calc_defense()

    def calc_defense(self):
        self.defense = self.base_defense * (self.rar_coef + 1) * (self.cond/50)

    def display(self):
        print(f"The {self.__class__.__name__} ({self.cond}) is {self.rar}.")
        print(f"Its defense value is {self.defense}.")

# initial testing
#x = item(85, "rare")
#y = sword(100, "epic")
#z = shield(50, "common")
#inv_test = [x, y, z]
#for i in inv_test:
    #i.display()

def gen_random_rarity():
    random_rarity = random.randint(1, 3)
    if random_rarity == 1:
        rr_str = "common"
    elif random_rarity == 2:
        rr_str = "rare"
    else:
        rr_str = "epic"
    return(rr_str)

def gen_random_condition():
    return(random.randint(0, 100))

# create four items, two shields and two swords
sword1 = sword(gen_random_condition(), gen_random_rarity())
sword2 = sword(gen_random_condition(), gen_random_rarity())
shield1 = shield(gen_random_condition(), gen_random_rarity())
shield2 = shield(gen_random_condition(), gen_random_rarity())
inv1 = [sword1, sword2, shield1, shield2]
print("_" * 80)
for i in inv1:
    i.display()

#inv2 = []
## we also could use a for loop with an if statement
#for i in range(4):
#    if i < 2: # gen two swords, for i=0 and i=1
#        inv2.append(sword(gen_random_condition(), gen_random_rarity()))
#    else:
#        inv2.append(shield(gen_random_condition(), gen_random_rarity()))
#print("_" * 80)
#for i in inv2:
#    i.display()

sword1.cond = sword1.cond * 0.5
shield1.cond = shield1.cond * 0.5

print()
for i in inv1:
    i.calc_value_fast()
    if isinstance(i, sword):
        i.calc_attack()
    if isinstance(i, shield):
        i.calc_defense()
    i.display()

print()
print("Finally let's also check MRO")
help(sword)
#help(shield)
```