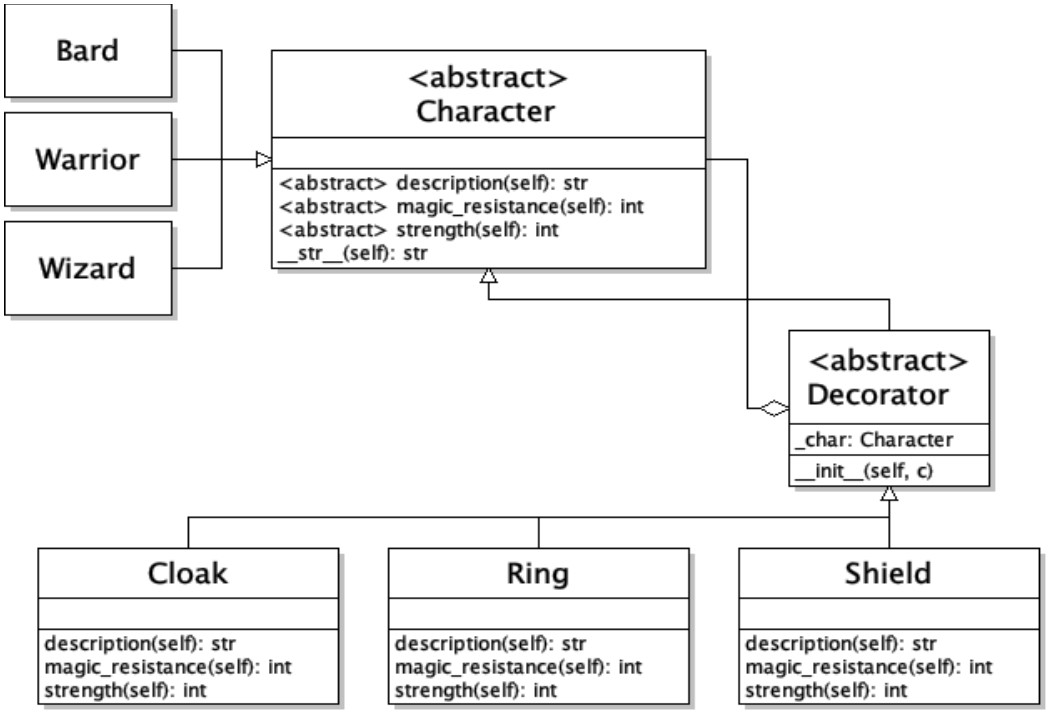
CECS 277 – Lab 11 – Decorator Pattern

Character Creator

Using the Decorator pattern, allow the user to build a character that will fight against three different monsters. The user can choose from three base types of characters (ie. Bard, Warrior, or Wizard). Then the user can choose two different items (ie. Shield, Ring, Cloak) to decorate their characters for a boost in abilities (ie. strength and magic resistance). Once the character is created, then it will face the three trials. If it passes two out of the three trials, then the user wins.

Use the following UML and class descriptions to create your program:



Classes:

1. Character - interface.
    a. description(self) – abstract – this will be the character's name and items.
    b. magic_resistance(self) – abstract – this will be the character's magic resist value.
    c. strength(self) – abstract – this will be the character's strength value.
    d. __str__(self) – return a string with the character's description, magic resistance, and strength.
2. Concrete Characters (Bard/Warrior/Wizard) - extends Character
    a. Override the three abstract methods and return the values according to the table below.
3. Decorator – abstract and extends from Character
    a. __init__(self, c) – set the character attribute to c.
    b. Override the three abstract methods and call each on the character attribute.
4. Item Classes (Cloak/Ring/Shield) – extends Decorator

a. Override the three abstract methods and call each method with super and add on the additional value according to the table bleow.

5. Main – present the user with a menu to choose the base character type. Then display a menu that allows the user to choose to decorate their character with two of the three items. Display the character. Then, the user will be presented with a series of three randomly selected monsters from the table below. Display the monster's stats. Based on the stats, the user may now choose to battle or dodge the attack. If they choose to battle, then compare the stats of the character and the monster, if both of the character's stats are greater than or equal to both of the monster's stats, then they pass the trial, otherwise they fail. If they choose to dodge, then they have a 25% chance of dodging the attack and passing the trial, otherwise they fail. If the user fails two trials, then the game ends. If the user passes two of the three trials, then they win the game, otherwise they lose.

Name	**Magic**	**Strength**
Barth the Bard	2	2
Harcor the Warrior	0	4
Altar the Wizard	3	1
Protective Cloak	+1	+1
Magic Ring	+2	+0
Sturdy Shield	+0	+2
Spiked Dragon	0	6
Orc Warlord	1	5
Shadow Knight	2	4
Lava Monster	3	3
Fiendish Ghoul	4	2
Goblin Mage	5	1
Dark Magician	6	0

Example Output:

```
Character Maker!
Choose a starting character:
1. Bard
2. Warrior
3. Wizard
> 1

Name: Barth the Bard
MR: 2
STR: 2
Choose 2 items:
1. Sturdy Shield
2. Magic Ring
3. Protective Cloak
> 1

Name: Shielded Barth the Bard
MR: 2
STR: 4
Choose 1 items:
1. Magic Ring
2. Protective Cloak
> 2

Name: Cloaked Shielded Barth the
Bard
MR: 3
STR: 5

You must pass two of the following
three trials!
Trial 1 of 3:
You encounter a Orc Warlord
MR: 1
STR: 5
Fight:
1. Battle
2. Dodge
> 1
You battle with the Orc Warlord and
easily defeat it.
You have passed this trial.
```

```
Trial 2 of 3:
You encounter a Goblin Mage
MR: 5
STR: 1
Fight:
1. Battle
2. Dodge
> 2
You attempt to dodge the Goblin
Mage, but it manages to hit you.
You have failed this trial.

Trial 3 of 3:
You encounter a Lava Monster
MR: 3
STR: 3
Fight:
1. Battle
2. Dodge
> 1
You battle with the Lava Monster
and easily defeat it.
You have passed this trial.

You have passed the three
trials...barely.
```

Notes:

1. You should have 10 different files: bard.py, character.py, check_input.py, cloak.py, decorator.py, main.py, ring.py, shield.py, warrior.py, and wizard.py.
2. Check all user input using the get_int_range function in the check_input module.
3. Do not create any extra methods, attributes, functions, parameters, etc.
4. Please do not create any global variables, or use any of the attributes globally (ie. do not access any of the attributes using the underscores, use the methods instead).
5. Use docstrings to document each of the classes, their attributes, and their methods.
6. 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.
7. Feel free to change the names of the monsters.
8. Each of your item classes should add on to the character's description, magic_resistance, and strength (even if that value is 0).
9. Thoroughly test your program before submitting:
    a. Make sure that each base character starts with the correct values.
    b. Make sure that every time the user decorates the character, its description, magic resistance, and strength change by the correct amount.
    c. Make sure that a random monster is chosen every round (if you would like to remove the monster from the list so you don't get repeats, you can, but it is not required).
    d. Make sure that the character's and the monster's stats are compared correctly (compares both the strength and magic values and both must be greater than or equal (ex. a character with MR: 4, STR: 4, will beat a monster with M: 4 and STR: 4, but not M: 4 and STR: 5 or M: 5 and STR: 4)).
    e. Make sure that the program ends when all three trials have been attempted or if the first two trials were failures.

Character Creator– Time estimate: 3-4 hours

Character Creator **10 points**	Correct. 2 points	A minor mistake. 1.5 points	A few mistakes. 1 point	Several mistakes. 0.5 points	No attempt. 0 points
Character classes (separate files)**:** 1. Character class is abstract with the three abstract methods and str method. 2. Bard/War/Wiz extend Character. 3. Bard/War/Wiz override 3 methods.					
Decorator (separate file)**:** 1. Inherits from ABC and Character. 2. Has a character attribute set in init. 3. Overrides the three methods and calls the method on the attribute.					
Item Decorations (separate files)**:** 1. Item classes extend the decorator. 2. Overrides the three methods that each call super and then add the appropriate values for the item.					
Main file (in separate file)**:** 1. Allows user to choose a character. 2. Allows user to choose two items. 3. Fights against three monsters. 4. User may battle or dodge. 5. Compares stats correctly. 6. Reports failure or success correctly. 7. Repeats until 2 failures or 3 battles. 8. Correctly reports win or loss.					
Code Formatting: 1. Correct documentation. 2. Meaningful variable names. 3. No exceptions thrown. 4. No global variables or accessing attributes directly. 5. Correct spacing.					