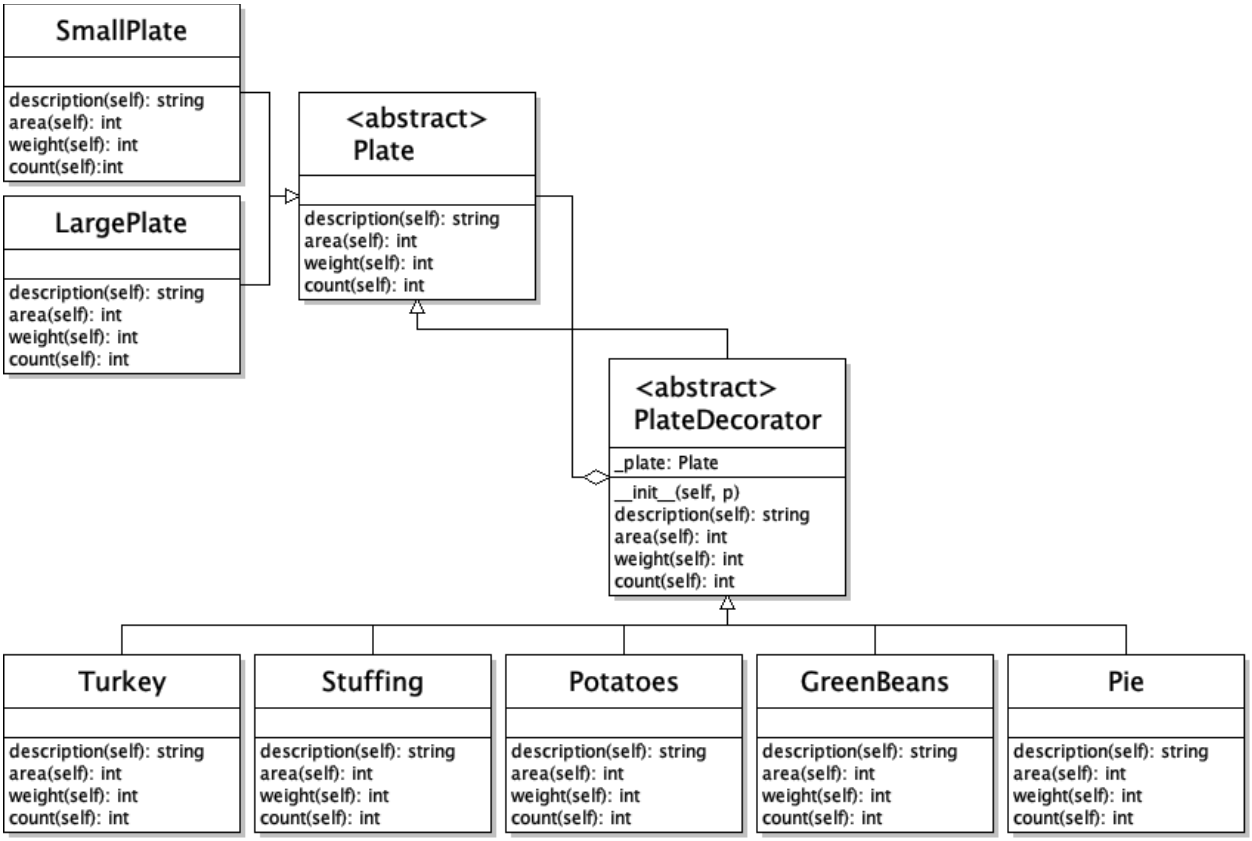
CECS 277 – Lab 12 – Decorator Pattern

Thanksgiving Dinner

Using the Decorator pattern, create a game that has the user add food to their plate without going over the weight or area limit of a paper plate. There are two different base types of plates: a small, sturdy 10-inch paper plate, or a large, flimsy, 12-inch plate. The plates can then be decorated with five different types of food: Turkey, Stuffing, Potatoes, Green Beans, and Pumpkin Pie. Each serving of the different foods has a weight, in ounces, and an area, in square inches. The user may load up their plate at the buffet as much as they like, but if the area or weight limit of the plate is surpassed, then their food falls to the floor.

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



Classes:

1. Plate - interface.
    a. description(self) – returns a string description of the plate and what is on it.
    b. area(self) – returns the remaining square inches the plate can hold.
    c. weight(self) – returns the remaining number of ounces the plate can hold.
    d. count(self) – returns the number of food items the plate is currently holding.
2. Concrete Plates (SmallPlate and LargePlate) - extends Plate
    a. description(self) – returns the description of the plate.
    b. area(self) – returns the area of the plate (small = 78, large = 113).
    c. weight(self) – returns the weight capacity of the plate (small = 32, large = 24).

   d. count(self) – returns 0 (ie. no items on the plate yet).
3. PlateDecorator – extends ABC and extends from Plate
   a. __init__(self, p) – pass in the plate p and assign it to the _plate attribute.
   b. description, area, weight, count methods: call each on your _plate attribute.
4. Food Decorations (Turkey, Stuffing, Potatoes, GreenBeans, Pie) – extend Decorator
   a. description(self) – call the superclass's description and additional description.
   b. area(self) – call the superclass's method and subtract the food item's area. Suggested areas: turkey=15, stuffing=18, potatoes=18, beans=20, pie=19.
   c. weight(self) – call the superclass's method and subtract the food item's weight. Suggested weights: turkey=4, stuffing=7, potatoes=6, beans=3, pie=8.
   d. count(self) – call the superclass's method and add 1 to increment the counter.
5. Main – Create a function examine_plate(p), that displays the plate's description, then based on the plate's area and weight capacity remaining, display a hint of how much more food the plate could hold and return False (Suggested hints for weight: 1-6: bending, 7-12: weak, 13+: strong. Suggested hints for area: 1-20: tiny bit, 21-40: some, 41+: plenty). If the plate failed (area or weight capacity is less than or equal to 0), display a message for the type of failure, then return True. In the main function, present the user with a menu to choose the base plate type. Then repeatedly prompt the user to add a new food item to the plate, decorate the plate with that food item and then call examine_plate to display the hint. Allow the user to add food until they decide to quit, or they spill their food on the floor. If they quit, display the contents of the final plate, number of items, and the space and weight remaining.

Example Outputs:

```
- Thanksgiving Dinner -
Serve yourself as much food as you
like from the buffet, but make sure
that your plate will hold without
spilling everywhere!
Choose a plate:
1. Small Sturdy Plate
2. Large Flimsy Plate
1
1. Turkey
2. Stuffing
3. Potatoes
4. Green Beans
5. Pie
6. Quit
1
Sturdy 10 inch paper plate with
Turkey
Sturdiness: Strong
Space available: Plenty
1. Turkey
2. Stuffing
3. Potatoes
4. Green Beans
5. Pie
6. Quit
2
Sturdy 10 inch paper plate with
Turkey and Stuffing
Sturdiness: Strong
Space available: Plenty
1. Turkey
2. Stuffing
3. Potatoes
4. Green Beans
5. Pie
6. Quit
1
Sturdy 10 inch paper plate with
Turkey and Stuffing and Turkey
Sturdiness: Strong
Space available: Some
1. Turkey
2. Stuffing
3. Potatoes
4. Green Beans
5. Pie
6. Quit
3
Sturdy 10 inch paper plate with
Turkey and Stuffing and Turkey and
Potatoes
Sturdiness: Weak
Space available: A tiny bit
1. Turkey
```

```
2. Stuffing
3. Potatoes
4. Green Beans
5. Pie
6. Quit
5
Sturdy 10 inch paper plate with
Turkey and Stuffing and Turkey and
Potatoes and Pie
Your plate isn't big enough for
this much food!  Your food spills
over the edge.

- Thanksgiving Dinner -
Serve yourself as much food as you
like from the buffet, but make sure
that your plate will hold without
spilling everywhere!
Choose a plate:
1. Small Sturdy Plate
2. Large Flimsy Plate
2
1. Turkey
2. Stuffing
3. Potatoes
4. Green Beans
5. Pie
6. Quit
1
Flimsy 12 inch paper plate with
Turkey
Sturdiness: Strong
Space available: Plenty
1. Turkey
2. Stuffing
3. Potatoes
4. Green Beans
5. Pie
6. Quit
2
Flimsy 12 inch paper plate with
Turkey and Stuffing
Sturdiness: Strong
Space available: Plenty
1. Turkey
2. Stuffing
3. Potatoes
4. Green Beans
5. Pie
6. Quit
1
Flimsy 12 inch paper plate with
Turkey and Stuffing and Turkey
Sturdiness: Weak
Space available: Plenty
1. Turkey
2. Stuffing
3. Potatoes
4. Green Beans
5. Pie
6. Quit
4
Flimsy 12 inch paper plate with
Turkey and Stuffing and Turkey and
Green Beans
Sturdiness: Bending
Space available: Plenty
1. Turkey
2. Stuffing
3. Potatoes
4. Green Beans
5. Pie
6. Quit
6
Flimsy 12 inch paper plate with
Turkey and Stuffing and Turkey and
Green Beans
Good job!  You made it to the table
with 4 items.
There was still 45 square inches
left on your plate.
Your plate could have held 6 more
ounces of food.
Don't worry, you can always go back
for more.  Happy Thanksgiving!
```

Notes:

1. You should have 10 different files: main.py, plate.py, small_plate.py, large_plate.py, plate_decorator.py, turkey.py, stuffing.py, potatoes.py, green_beans.py, and pie.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).
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. Give each of your food items different areas and weights. You can come up with your own or use the suggested values.
8. The examine_plate function should use different ranges to supply the hints for area and weight capacity remaining. You can come up with your own or use the suggested values.
9. Thoroughly test your program before submitting:
    a. Make sure that when the user adds food to the plate, its description, area, weight, and count are updated with the correct values.
    b. Make sure that the program ends when the user chooses to quit or the plate fails.

Thanksgiving Dinner – Time estimate: 3 hours

Thanksgiving Dinner **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
Plate classes (separate files)**:** 1. Plate class is abstract with the four abstract methods. 2. Small/LargePlate extend from Plate. 3. Small/LargePlate override methods.					
Plate Decorator (separate file)**:** 1. Inherits from ABC and Plate. 2. Has a plate attribute set in init. 3. Overrides the four methods and calls the method on the attribute.					
Food Decorations (separate files)**:** 1. Food classes extend the decorator. 2. Overrides the four methods that each call super and then add/subtract the appropriate values for the food item.					
Main file (in separate file)**:** 1. Has examine_plate function that displays the plate, checks if it failed, displays the hint, and returns a boolean. 2. Allows user to choose a plate. 3. Allows user to add food to the plate. 4. Displays hints. 5. Ends when plate fails or user quits. 6. If user quits, it displays the plate, number of food items, and how much space and weight was remaining.					
Code Formatting: 1. Correct documentation. 2. Meaningful variable names. 3. No exceptions thrown. 4. No global variables or accessing attributes directly. 5. Correct spacing.					