OOP - Practice 7

This practice does not need to be submitted.

Refer to code below (which is the solution to Practice 6):

```
class Cake:
    def __init__(self, flavor, price, slices):
        self.flavor = flavor
        self.price = price
        self.slices = slices
        self.remaining = slices

    def print_description(self):
        print(f"The {self.flavor} cake costs ${self.price}")
        print(f" and is divided into {self.slices} slices.")

    def sell(self, count):
        if count <= 0:
            return("Cannot sell zero or negative slices!")
        elif count > self.remaining:
            return(f"Cannot sell more slices than we have ({self.remaining})!")
        else:
            self.remaining -= count
            return f"This cake has {self.remaining} slices remaining."

    def getValue(self):
        return (self.price / self.slices) * self.remaining

    def isEqualTo(self, otherCake):
        return (self.getValue() == otherCake.getValue())

    def isLessThan(self, otherCake):
        return (self.getValue() < otherCake.getValue())

    def isGreaterThan(self, otherCake):
        return (self.getValue() > otherCake.getValue())

spice_cake = Cake("spice", 18, 8)
chocolate_cake = Cake("chocolate", 24, 6)

print(spice_cake.sell(1))
print(chocolate_cake.sell(3))
print(spice_cake.isEqualTo(chocolate_cake))
print(spice_cake.isGreaterThan(chocolate_cake))
print(spice_cake.isLessThan(chocolate_cake))
```

1. Name the instance variables.
2. Name the methods.
3. Is `self.price` part of the ***state*** or part of the ***behavior*** of an object of the Cake class?

4. Is `sell()` part of the ***state*** or part of the ***behavior*** of an object of the Cake class?

5. How many instances of the class Cake are being created?

6. When the cake objects are created, what is the value for self.remaining? Is it the same for all cakes?

7. What value is being passed to the parameter `self` when the line `print(spice_cake.isEqualTo(chocolate_cake))` is executed?

8. What will happen if we add the line `coconut_cake = Cake()` to the program?

9. What will each of the following lines do if added to the main program?

```
a) print("Spice: ", spice_cake.getValue())
b) print("Spice: ", spice_cake.remaining)
c) print("Chocolate: ", getValue(chocolate_cake))
```

10. Both `self.slices` and `self.remaining` are initialized to the value passed in the parameter `slices`. Why does the class need 2 variables if they are both initialized to the same value?

11. What will happen if we change the name of the `__init__` method to `_init_`?

12. Create a class named Bakery, that will have a list of objects of the class Cake as an instance variable. That class must have a method to add a new cake to the list, and a method to display the information for all the cakes in the list.

13. Create a method in the Bakery class that searches the list of cakes for the cake with a given flavor and **returns that cake object, if found, or *None* if not found**. The method definition will look like this: `get_cake(self, flavor)`.

14. Create a method in the Bakery class that calls `get_cake(flavor)` to find a specific cake with a given flavor and **returns the number of slices remaining for that cake**. **If the flavor is not found, return *None*.** The method should call `get_cake(flavor)` to find the specific cake. The method definition will look like this: `get_remaining_slices(self, flavor)`.

15. Create a method in the Bakery class to sell a slice of a specific cake. The method definition will look like this: `sell_slice(self, flavor)`. If the given flavor exists, this method must check if the cake has any slices remaining and then call `sell(1)` to update that cake information. It must **return the price for the slice sold, or *None* if there are no slices left or the flavor does not exist**.

16. Add a menu to interact with the user, providing options to add a new cake, display information for all cakes, purchase a slice of a specific cake, and lookup the number of slices remaining for a specific cake.