4.4 Наследование. Атрибуты private и protected

7 out of 11 steps passed 12 out of 22 points received

Видео-разбор подвига (решение смотреть только после своей попытки):
https://youtu.be/d5aNVdHGj44 (https://youtu.be/d5aNVdHGj44)

Подвиг 7 (введение в паттерн слушатель). Своей работой вы немного впечатлили начальство и оно поручило вам доделать паттерн слушатель (listener). Идея этого паттерна очень проста и основа реализуется следующим образом:

```
class Observer:
    def update(self, data):
        pass

    def __hash__(self):
        return hash(id(self))


class Subject:
    def __init__(self):
        self.__observers = {}
        self.__data = None

    def add_observer(self, observer):
        self.__observers[observer] = observer

    def remove_observer(self, observer):
        if observer in self.__observers:
            self.__observers.pop(observer)

    def __notify_observer(self):
        for ob in self.__observers:
            ob.update(self.__data)

    def change_data(self, data):
        self.__data = data
        self.__notify_observer()
```

Здесь в объектах класса Subject можно зарегистрировать (добавить) множество объектов класса Observer (наблюдатель, слушатель). Это делается с помощью метода add_observer(). Затем, когда данные (self.__data) меняются путем вызова метода change_data() класса Subject, то у всех слушателей автоматически вызывается метод update(). В этом методе можно прописать самую разную логику работы при изменении данных в каждом конкретном слушателе.

В проекте данный паттерн предполагается использовать для отображения информации о погоде в различных форматах:

- текущая температура;
- текущее атмосферное давление;
- текущая влажность воздуха.

Для этого сами данные определяются классом:

```
class Data:
    def __init__(self, temp, press, wet):
        self.temp = temp    # температура
        self.press = press  # давление
        self.wet = wet      # влажность
```

А вам поручается разработать дочерние классы, унаследованные от класса Observer, с именами:

TemperatureView - слушатель для отображения информации о температуре;
PressureView - слушатель для отображения информации о давлении;
WetView - слушатель для отображения информации о влажности.

Каждый из этих классов должен переопределять метод update() базового класса так, чтобы выводилась в консоль информация в формате:

TemperatureView: "Текущая температура <число>"
PressureView: "Текущее давление <число>"
WetView: "Текущая влажность <число>"

Важно: для вывода информации в консоль используйте функцию print() с одним аргументом в виде F-строки.

Пример использования классов (эти строчки в программе писать не нужно):

```
subject = Subject()
tv = TemperatureView()
pr = PressureView()
wet = WetView()

subject.add_observer(tv)
subject.add_observer(pr)
subject.add_observer(wet)

subject.change_data(Data(23, 150, 83))
# выведет строчки:
# Текущая температура 23
# Текущее давление 150
# Текущая влажность 83
subject.remove_observer(wet)
subject.change_data(Data(24, 148, 80))
# выведет строчки:
# Текущая температура 24
# Текущее давление 148
```

P.S. В программе нужно объявить только классы. На экран выводить ничего не нужно.