Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчёт по рубежному контролю №12
Вариант №19

Выполнил:
студент группы РТ5-31Б
Сафонов Федор
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2021 г

Задание

Условия рубежного контроля №2 по курсу БКИТ

Рубежный контроль представляет собой разработку тестов на языке Python.

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

Текст программы

- RK_1_upd.py

```
class Detail:
    '''Деталь'''
    def __init__(self, id, name, price, id_producer):
        self.id = id
        self.name = name
        self.price = price
        self.id_producer = id_producer

class DetProd:
    '''Детали производителя'''
    def __init__(self, det_id, prod_id):
        self.det_id = det_id
        self.prod_id = prod_id

class Producer:
    '''Произовдитель'''
    def __init__(self, id, name):
        self.id = id
        self.name = name

#Производители
prods = [
    Producer(1, "немецкая компания"),
    Producer(2, "строительный завод"),
    Producer(3, "китайский завод"),
    Producer(4, "авиационный завод"),
    Producer(5, "антиквариатная фабрика"),
]

#Детали
dets = [
```

```
    Detail(1, "айфон", 100, 3),
    Detail(2, "шасси", 50, 3),
    Detail(3, "антресоль", 10, 1),
    Detail(4, "двигатель", 80, 1),
    Detail(5, "кран", 10, 2),
]

#Связь деталь-сотрудник
dets_prods = [
    DetProd(2, 1),
    DetProd(4, 1),
    DetProd(1, 5),
    DetProd(3, 5),
    DetProd(3, 4),
    DetProd(3, 2),
    DetProd(5, 1),
    DetProd(5, 2),
    DetProd(5, 3),
    DetProd(5, 4),
    DetProd(5, 5),
]

#один ко многим
ONE_TO_MANY = [(p.name, d.name, d.price)
        for p in prods
        for d in dets
        if p.id == d.id_producer]

#многие ко многим
MANY_TO_MANY = [(d.name, p.name)
    for p in prods
    for d in dets
    for dp in dets_prods
    if dp.det_id == d.id and dp.prod_id == p.id
]

def solve1():
    '''Вывести производителей, в которых есть слово "завод" и их детали'''
    correct_prods = set(
        [elem[0] for elem in ONE_TO_MANY if 'завод' in elem[0]])  # отбираем
производителей в которых есть слово "завод"
    keys = [[] for i in range(len(correct_prods))]
    ans = dict(zip(correct_prods, keys))  # словарь с заводами и пустыми списками
    for p, d, _ in ONE_TO_MANY:
        if p in ans:
            ans[p].append(d)
    return ans

def solve2():
    '''Вывести производителей отсортированных по средней цене их деталей.
        Среднюю цену округлить до двух знаков после запятой'''
    prod_names = set(elem[0] for elem in ONE_TO_MANY)
    mas = []
    for elem in prod_names:
```

```
            cnt = 0  # количество деталей завода elem
            sum = 0  # сумма цен всех деталей
            for p, d, price in ONE_TO_MANY:
                if p == elem:
                    sum += price
                    cnt += 1
            mas.append((elem, sum / cnt))
        mas = sorted(mas, key=lambda x: x[1])
        return mas

    def solve3():
        '''Вывести все детали, начинающиеся на букву "а",
            а также их отделы'''
        d = dict()
        for d_name, p_name in MANY_TO_MANY:
            if d_name.startswith('а'):  # название детали начинается на "а"
                try:  # ключ уже есть
                    d[d_name].append(p_name)
                except:  # создаем ключ
                    d[d_name] = []
                    d[d_name].append(p_name)
        return d


    def main():
        '''Основная функция'''
        print(solve1())
        print()
        print(solve2())
        print()
        print(solve3())


    if __name__ == '__main__':
        main()
```

- unittests.py

```
    import unittest

    from RK_1_upd import *

    class MyTests(unittest.TestCase):

     def test_contains(self):
     # проверяем во всех ли заводах есть слово "завод"
     self.assertTrue(all([x.find('завод') != -1 for x in solve1().keys()]))

        def test_sort(self):
            self.assertEqual([x[1] for x in solve2()], sorted([x[1] for x in solve2()]))

        def test_prefix(self):
```

```
        self.assertTrue(all([x.startswith('a') for x in solve3().keys()]))

if __name__ == '__main__':
    unittest.main()
```

Вывод программы

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK

Process finished with exit code 0
```