Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)
Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Рубежный контроль 2

по дисциплине «Базовые компоненты интернет-технологий»

Вариант 19Г

Выполнил:

Студент группы ИУ5-34Б

Хатин М.С.

2021 г.

Постановка задачи

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
Файл main.py

```
class Det:
    """Деталь"""

    def __init__(self, id, name, price, prod_id):
        self.id = id
        self.name = name  # название детали
        self.price = price  # стоимость [$]
        self.prod_id = prod_id


class Prod:
    """Производитель"""

    def __init__(self, id, name):
        self.id = id
        self.name = name


class DetProd:
    """
    'Детали производителя' для реализации
    связи многие-ко-многим
    """

    def __init__(self, prod_id, det_id):
        self.prod_id = prod_id
        self.det_id = det_id


# Производители
prods = [
    Prod(1, 'Samsung'),
    Prod(2, 'LG'),
    Prod(3, 'Apple'),

    Prod(11, 'Acronis'),
    Prod(22, 'Microsoft'),
    Prod(33, 'Panasonic'),
]

# Детали
dets = [
    Det(1, 'Кабель type-c', 10, 1),
    Det(2, 'Шнур usb 2.0', 15, 2),
    Det(3, 'Шнур usb 3.0', 11, 3),
    Det(4, 'Блок питания 12 v', 12, 3),
    Det(5, 'Блок питания 14 v', 16, 3),
]

dets_prods = [
    DetProd(1, 1),
    DetProd(2, 2),
    DetProd(3, 3),
    DetProd(3, 4),
    DetProd(3, 5),

    DetProd(11, 1),
    DetProd(22, 2),
    DetProd(33, 3),
```

```
    DetProd(33, 4),
    DetProd(33, 5),
]


def g1(one_to_many):
    return list(filter(lambda entry: entry[2][0] == 'A', one_to_many))


def g2(one_to_many):
    return [(prod.name, max([price for _, price, prod_name in one_to_many if prod_name ==
prod.name])) for prod in prods
            if len(list(filter(lambda entry: entry[2] == prod.name, one_to_many))) > 0]


def g3(many_to_many):
    return sorted(many_to_many, key=lambda entry: entry[2])


def main():

    one_to_many = [(det.name, det.price, prod.name)
                   for prod in prods
                   for det in dets
                   if det.prod_id == prod.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(p.name, dp.prod_id, dp.det_id)
                         for p in prods
                         for dp in dets_prods
                         if p.id == dp.prod_id]

    many_to_many = [(d.name, d.price, prod_name)
                    for prod_name, _, det_id in many_to_many_temp
                    for d in dets if d.id == det_id]

    print('Задание Г1')
    print(g1(one_to_many))

    print('\nЗадание Г2')
    print(g2(one_to_many))

    print('\nЗадание Г3')
    print(g3(many_to_many))


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

Файл tests.py

```
import unittest
from main import g1, g2, g3
from main import dets, dets_prods, prods


one_to_many = [(det.name, det.price, prod.name)
               for prod in prods
               for det in dets
               if det.prod_id == prod.id]


many_to_many_temp = [(p.name, dp.prod_id, dp.det_id)
                     for p in prods
                     for dp in dets_prods
                     if p.id == dp.prod_id]

many_to_many = [(d.name, d.price, prod_name)
                for prod_name, _, det_id in many_to_many_temp
                for d in dets if d.id == det_id]


class Test(unittest.TestCase):
    def test_g1(self):
        self.assertEqual(g1(one_to_many), [('Шнур usb 3.0', 11, 'Apple'), ('Блок питания 12 v', 12,
'Apple'),\
                                           ('Блок питания 14 v', 16, 'Apple')])

    def test_g2(self):
        self.assertEqual(g2(one_to_many), [('Samsung', 10), ('LG', 15), ('Apple', 16)])

    def test_g3(self):
        self.assertEqual(g3(many_to_many), [('Кабель type-c', 10, 'Acronis'), ('Шнур usb 3.0', 11,
'Apple'),\
                                            ('Блок питания 12 v', 12, 'Apple'), ('Блок питания 14 v', 16,
'Apple'),\
                                            ('Шнур usb 2.0', 15, 'LG'), ('Шнур usb 2.0', 15, 'Microsoft'),\
                                            ('Шнур usb 3.0', 11, 'Panasonic'), ('Блок питания 12 v', 12,
'Panasonic'),\
                                            ('Блок питания 14 v', 16, 'Panasonic'), ('Кабель type-c', 10,
'Samsung')])


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

Результат выполнения программы

```
maxim@maxim-HLYL-WXX9:~/PycharmProjects/rk2[BKIT]$ python tests.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```