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 ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»
Отчет по рубежному контролю №2

Выполнил:
студент группы ИУ5-33Б
Требуков Д.А.

Проверил:
Преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич

Москва, 2021 г.

Задание:

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:

main.py

```
from operator import itemgetter


class Detail:
    """Детали"""
    def __init__ (self, id, name, value, manufacturer_id):
        self.id = id
        self.name = name
        self.value = value
        self.manufacturer_id = manufacturer_id

class Manufacturer:
    """Производители"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class ManufactDetail:
    """Детали от производителей"""
    def __init__(self, manufacturer_id, detail_id):
        self.manufacturer_id = manufacturer_id
        self.detail_id = detail_id

#Производители
manufacturers = [
    Manufacturer(1, 'Schlieckmann'),
    Manufacturer(2, 'Klokkerholm'),
    Manufacturer(3, 'Signeda'),

    Manufacturer(11, 'Sigma'),
    Manufacturer(22, 'Tyg'),
    Manufacturer(33, 'Kito')
]
#Детали
details = [
    Detail(1, 'Кузов', 15000, 1),
    Detail(2, 'Поршни', 20000, 1),
    Detail(3, 'Ремень ГРМ', 5000, 2),
    Detail(4, 'Подушки двигателя', 25000, 3),
    Detail(5, 'Система охлаждения', 7000, 3)
]

detail_manufacs = [
```

```
    ManufactDetail(1, 1),
    ManufactDetail(1, 2),
    ManufactDetail(2, 3),
    ManufactDetail(3, 4),
    ManufactDetail(3, 5),

    ManufactDetail(11, 1),
    ManufactDetail(22, 2),
    ManufactDetail(22, 3),
    ManufactDetail(33, 4),
    ManufactDetail(33, 5)
]

def res_1(arr):
    answer_list = []
    for detail, value, manufacturer in arr:
        if detail[0] == 'П':
            answer_list.append([detail, manufacturer])
    return answer_list

def res_2(arr):
    answer_list = [[arr[0][2], arr[0][1]]]
    for detail, value, manufacturer in arr:
        if manufacturer == answer_list[len(answer_list)-1][0]:
            if value < answer_list[len(answer_list)-1][1]:
                answer_list[len(answer_list)-1][1]= value
        else:
            answer_list.append([manufacturer, value])
    return sorted(answer_list, key=itemgetter(1))

def res_3(arr):
    answer_list = []
    for detial, value, manufacturer in arr:
        answer_list.append([detial, manufacturer])
    return sorted(answer_list, key=itemgetter(0))
def main():
    # Соединение данных один-ко-многим
    one_to_many = [(d.name, d.value, m.name)
                   for m in manufacturers
                   for d in details
                   if d.manufacturer_id == m.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(m.name, md.manufacturer_id, md.detail_id)
                         for m in manufacturers
                         for md in detail_manufacs
                         if m.id == md.manufacturer_id]

    many_to_many = [(d.name, d.value, m_name)
                    for m_name, m_id, d_id in many_to_many_temp
                    for d in details if d.id == d_id]

    print('Задание В1')
    """
        «Производитель» и «Деталь» связаны соотношением один-ко-многим.
        Выведите список всех деталей, у которых названия
        начинаются с буквы «П», и названия их производителей.
    """
    print(res_1(one_to_many))

    print('Задание В2')
    """
        «Производитель» и «Деталь» связаны соотношением один-ко-многим.
        Выведите список производителей с минимальной стоимостью деталей
```

```
        у каждого производителя, отсортированный по минимальной стоимости.
    """
    print(res_2(one_to_many))

    print('Задание В3')
    """
    «Производитель» и «Деталь» связаны соотношением многие-ко-многим.
    Выведите список всех связанных производителей и деталей,
    отсортированный по деталям, сортировка по производителям произвольная.
    """
    print(res_3(many_to_many))

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

tests.py

```
from main import Detail, Manufacturer, ManufactDetail, res_1, res_2, res_3
from unittest import TestCase

class Test(TestCase):
    def setUp(self) -> None:
        # Производители
        self.manufacturers = [
            Manufacturer(1, 'Schlieckmann'),
            Manufacturer(2, 'Klokkerholm'),
            Manufacturer(3, 'Signeda'),

            Manufacturer(11, 'Sigma'),
            Manufacturer(22, 'Tyg'),
            Manufacturer(33, 'Kito')
        ]
        # Детали
        self.details = [
            Detail(1, 'Кузов', 15000, 1),
            Detail(2, 'Поршни', 20000, 1),
            Detail(3, 'Ремень ГРМ', 5000, 2),
            Detail(4, 'Подушки двигателя', 25000, 3),
            Detail(5, 'Система охлаждения', 7000, 3)
        ]
        self.detail_manufacs = [
            ManufactDetail(1, 1),
            ManufactDetail(1, 2),
            ManufactDetail(2, 3),
            ManufactDetail(3, 4),
            ManufactDetail(3, 5),

            ManufactDetail(11, 1),
            ManufactDetail(22, 2),
            ManufactDetail(22, 3),
            ManufactDetail(33, 4),
            ManufactDetail(33, 5)
        ]
        self.one_to_many = [(d.name, d.value, m.name)
                    for m in self.manufacturers
                    for d in self.details
                    if d.manufacturer_id == m.id]
        self.many_to_many_temp = [(m.name, md.manufacturer_id, md.detail_id)
                          for m in self.manufacturers
                          for md in self.detail_manufacs
                          if m.id == md.manufacturer_id]
        self.many_to_many = [(d.name, d.value, m_name)
                     for m_name, m_id, d_id in self.many_to_many_temp
```

```
                    for d in self.details if d.id == d_id]

    def test1(self):
        result = res_1(self.one_to_many)
        desired = [['Поршни', 'Schlieckmann'], ['Подушки двигателя',
'Signeda']]
        self.assertEqual(result, desired)

    def test2(self):
        result = res_2(self.one_to_many)
        desired = [['Klokkerholm', 5000], ['Signeda', 7000], ['Schlieckmann',
15000]]
        self.assertEqual(result, desired)

    def test3(self):
        result = res_3(self.many_to_many)
        desired = [['Кузов', 'Schlieckmann'], ['Кузов', 'Sigma'],
                   ['Подушки двигателя', 'Signeda'], ['Подушки двигателя',
'Kito'],
                   ['Поршни', 'Schlieckmann'], ['Поршни', 'Tyg'], ['Ремень
ГРМ', 'Klokkerholm'],
                   ['Ремень ГРМ', 'Tyg'], ['Система охлаждения', 'Signeda'],
['Система охлаждения', 'Kito']]
        self.assertEqual(result, desired)
```

Результаты программы: