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»
Рубежный контроль № 2
Вариант 3

Выполнил:
Студент группы ИУ5-34Б
Верин Дмитрий
Дата и подпись:

Преподаватель:
Гапанюк Ю.Е.
Дата и подпись:

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 driver.py

```
class Driver:
    def __init__(self, id, fio, sal, car_park_id):
        self.id = id
        self.fio = fio
        self.sal = sal
        self.car_park_id = car_park_id
```

Файл carpark.py

```
class CarPark:
    def __init__(self, id, name):
        self.id = id
        self.name = name
```

Файл drivercarpark.py

```
class DrCarPark:
    def __init__(self, car_park_id, driver_id):
        self.car_park_id = car_park_id
        self.driver_id = driver_id
```

Файл main.py

```
from operator import itemgetter
from driver import Driver
from carpark import CarPark
from drivercarpark import DrCarPark

drivers = [
    Driver(1, 'Петров', 25000, 1),
    Driver(2, 'Иванов', 27000, 2),
    Driver(3, 'Сидоров', 30000, 3),
    Driver(4, 'Михайлов', 24000, 3),
    Driver(5, 'Базаров', 31000, 1)]

car_parks = [
    CarPark(1, 'Автопарк №1'),
    CarPark(2, 'Автопарк №2'),
    CarPark(3, 'Автопарк №3'),
    CarPark(11, 'Парк №1'),
    CarPark(22, 'Парк №2'),
    CarPark(33, 'Парк №3')]

dr_car_park = [
    DrCarPark(1,1),
    DrCarPark(2,2),
    DrCarPark(3,3),
    DrCarPark(2,4),
    DrCarPark(3,5),

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

```python
def first_task(car_parks, one_to_many):
    res_11 = [(c_p.name,
            list(fio for fio, _, name in one_to_many if name == c_p.name))
            for c_p in car_parks if c_p.name[0] == 'A']
    return res_11

def second_task(car_parks, one_to_many):
    res_12_unsorted = []
    for c_p in car_parks:
        c_p_drivers = list(filter(lambda x: x[2] == c_p.name, one_to_many))
        if len(c_p_drivers) > 0:
            res_12_unsorted.append(
                (c_p.name, max(c_p_drivers, key = lambda x: x[1])[1]))
    res_12 = sorted(res_12_unsorted, key = itemgetter(1), reverse = True)
    return res_12

def third_task(many_to_many):
    res_13 = []
    for fio, _, c_p_name in many_to_many:
        res_13.append((fio, c_p_name))
    res_13 = sorted(res_13, key = itemgetter(1))
    return res_13

def main():
    one_to_many = [(dr.fio, dr.sal, c_p.name)
        for c_p in car_parks
        for dr in drivers
        if dr.car_park_id == c_p.id]

    many_to_many_temp = [(c_p.name, d_c_p.car_park_id, d_c_p.driver_id)
        for c_p in car_parks
        for d_c_p in dr_car_park
        if c_p.id == d_c_p.car_park_id]

    many_to_many = [(dr.fio, dr.sal, c_p_name)
        for c_p_name, _, driver_id in many_to_many_temp
        for dr in drivers if dr.id == driver_id]

    print('Задание Г1')
    print(first_task(car_parks, one_to_many))
    print('Задание Г2')
    print(second_task(car_parks, one_to_many))
    print('Задание Г3')
    print(third_task(many_to_many))

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

Файл tests.py

```
import unittest
from main import drivers, car_parks, dr_car_park
from main import first_task, second_task, third_task

one_to_many = [(dr.fio, dr.sal, c_p.name)
    for c_p in car_parks
    for dr in drivers
    if dr.car_park_id == c_p.id]

many_to_many_temp = [(c_p.name, d_c_p.car_park_id, d_c_p.driver_id)
    for c_p in car_parks
    for d_c_p in dr_car_park
    if c_p.id == d_c_p.car_park_id]

many_to_many = [(dr.fio, dr.sal, c_p_name)
    for c_p_name, _, driver_id in many_to_many_temp
    for dr in drivers if dr.id == driver_id]

class Tests(unittest.TestCase):

    def test_first_task(self):
        self.assertEqual(first_task(car_parks, one_to_many), [('Автопарк №1', ['Петров',
'Базаров']),
         ('Автопарк №2', ['Иванов']), ('Автопарк №3', ['Сидоров', 'Михайлов'])])

    def test_second_task(self):
        self.assertEqual(second_task(car_parks, one_to_many), [('Автопарк №1', 31000),
         ('Автопарк №3', 30000), ('Автопарк №2', 27000)])

    def test_third_task(self):
        self.assertEqual(third_task(many_to_many), [('Петров', 'Автопарк №1'),
        ('Иванов', 'Автопарк №2'), ('Михайлов', 'Автопарк №2'),
         ('Сидоров', 'Автопарк №3'), ('Базаров', 'Автопарк №3'), ('Петров', 'Парк №1'),
         ('Иванов', 'Парк №2'), ('Михайлов', 'Парк №2'), ('Сидоров', 'Парк №3'),
          ('Базаров', 'Парк №3')])


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

Результат выполнения

```
C:\Users\Дмитрий\Desktop\prog>python tests.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```