Бирюкова Екатерина 3А

Задание для РК1

Вариант 3.

Класс 1 – водитель, класс 2 – автопарк.

Вариант запроса А.

1. «Автопарк» и «Водитель» связаны соотношением один-ко-многим. Выведите список всех связанных водителей и автопарков, отсортированный по автопаркам, сортировка по водителям произвольная.
2. «Автопарк» и «Водитель» связаны соотношением один-ко-многим. Выведите список автопарков с суммарной зарплатой водителей в каждом автопарке, отсортированный по суммарной зарплате.
3. «Автопарк» и «Водитель» связаны соотношением многие-ко-многим. Выведите список всех автопарков, у которых в названии присутствует слово «Московский», и список работающих в них водителей.

Задание для РК2

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.

Текст файла rk_1_refactored.py

```
from operator import itemgetter

class Driver:
    """Водитель"""
    def __init__(self, id, fio, sal, park_id):
        self.id = id
        self.fio = fio
        self.sal = sal
        self.park_id = park_id

class Park:
    """Автопарк"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class Driver_Park:
    """Водители автопарка"""
    def __init__(self, driver_id, park_id):
        self.driver_id = driver_id
        self.park_id = park_id

# Автопарки
parks = [
    Park(1, 'Нижегородский автопарк'),
    Park(2, 'Московский автопарк'),
    Park(3, 'Костромской автопарк'),


    Park(11, 'Нижегородский (другой) автопарк'),
```

```
    Park(22, 'Московский (другой) автопарк'),
    Park(33, 'Костромской (другой) автопарк'),
]

# Водители
drivers = [
    Driver(1, 'Петров', 10000, 1),
    Driver(2, 'Сидоров', 15000, 2),
    Driver(3, 'Иванов', 20000, 3),
    Driver(4, 'Абакумов', 25000, 1),
    Driver(5, 'Больков', 30000, 2),
]

drivers_parks = [
    Driver_Park(1,1),
    Driver_Park(2,2),
    Driver_Park(3,3),
    Driver_Park(4,1),
    Driver_Park(5,2),


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

def data():
    one_to_many = [(d.fio, d.sal, p.name)
        for p in parks
        for d in drivers
        if d.park_id==p.id]

    many_to_many_temp = [(p.name, dp.park_id, dp.driver_id)
        for p in parks
        for dp in drivers_parks
        if p.id==dp.park_id]

    many_to_many = [(d.fio, d.sal, park_name)
        for park_name, park_id, driver_id in many_to_many_temp
        for d in drivers if d.id==driver_id]

    return one_to_many, many_to_many

def task_1 (one_to_many):
    return sorted(one_to_many, key=itemgetter(2))

def task_2 (one_to_many):
    res_12_unsorted = []
    for p in parks:
        drivers_of_park = list(filter(lambda i: i[2]==p.name, one_to_many))
        if len(drivers_of_park) > 0:
            p_sals = [sal for _,sal,_ in drivers_of_park]
            p_sals_sum = sum(p_sals)
            res_12_unsorted.append((p.name, p_sals_sum))
    return sorted(res_12_unsorted, key=itemgetter(1), reverse=True)

def task_3 (many_to_many, word):
    res_13 = {}
    for p in parks:
        if word in p.name:
            drivers_of_park = list(filter(lambda i: i[2]==p.name, many_to_many))
            drivers_of_park_names = [x for x,_,_ in drivers_of_park]
            res_13[p.name] = drivers_of_park_names
    return res_13
```

```
def main():
    one_to_many, many_to_many = data()

    # Задание 1
    # «Автопарк» и «Водитель» связаны соотношением один-ко-многим.
    # Выведите список всех связанных водителей и автопарков, отсортированный по автопаркам, сортировка по водителям произвольная.
    print('Задание А1')
    print('«Автопарк» и «Водитель» связаны соотношением один-ко-многим.')
    print('Список всех связанных водителей и автопарков, отсортированный по автопаркам, сортировка по водителям произвольная:')
    res_11 = task_1(one_to_many)
    for elem in res_11:
        print(elem)

    # Задание 2
    # «Автопарк» и «Водитель» связаны соотношением один-ко-многим.
    # Выведите список отделов с суммарной зарплатой водителей в каждом автопарке, отсортированный по суммарной зарплате.
    print('\nЗадание А2')
    print('«Автопарк» и «Водитель» связаны соотношением один-ко-многим.')
    print('Список отделов с суммарной зарплатой водителей в каждом автопарке, отсортированный по суммарной зарплате:')
    res_12 = task_2(one_to_many)
    for elem in res_12:
        print(elem)

    # Задание 3
    # «Автопарк» и «Водитель» связаны соотношением многие-ко-многим.
    # Выведите список всех автопарков, у которых в названии присутствует слово «Московский», и список работающих в них водителей.
    print('\nЗадание А3')
    print('«Автопарк» и «Водитель» связаны соотношением многие-ко-многим.')
    print('Список всех автопарков, у которых в названии присутствует слово «Московский», и список работающих в них водителей:')
    res_13 = task_3(many_to_many, 'Московский')
    for elem, amount in res_13.items():
        print("{} ({})".format(elem, amount))

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

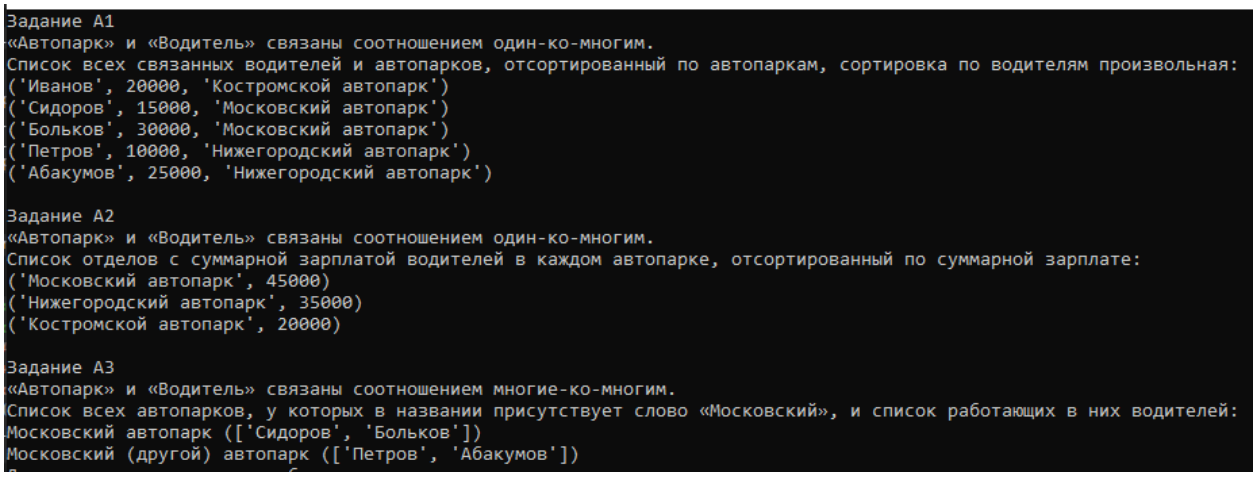
Текст файла rk_2.py

```
#!/usr/bin/python
# -*- coding: cp1251 -*-

import unittest
from rk_1_refactored import *

# Тестирование класса «Автопарк»
class TestPark(unittest.TestCase):

    def test_park(self):
        park_ex = Park(1, 'Нижегородский автопарк')
        self.assertEqual(park_ex.id, 1)
        self.assertEqual(park_ex.name, 'Нижегородский автопарк')

# Тестирование класса «Водитель»
class TestDriver(unittest.TestCase):

    def test_driver(self):
        driver_ex =  Driver(1, 'Петров', 10000, 1)
        self.assertEqual(driver_ex.id, 1)
        self.assertEqual(driver_ex.fio, 'Петров')
        self.assertEqual(driver_ex.sal, 10000)
```

```
        self.assertEqual(driver_ex.park_id, 1)

# Тестирование класса для реализация связи многие ко многим
class TestDriver_Park(unittest.TestCase):

    def test_driver_park(self):
        driver_park_ex =  Driver_Park(1,1)
        self.assertEqual(driver_park_ex.driver_id, 1)
        self.assertEqual(driver_park_ex.park_id, 1)


class TestFunctions(unittest.TestCase):
    # Генерация данных
    def setUp(self):
        self.one_to_many, self.many_to_many = data()

    # Тестирование задания №1
    def test_task_1(self):
        result = task_1(self.one_to_many)
        self.assertEqual(result, [('Иванов', 20000, 'Костромской автопарк'), ('Сидоров', 15000, 'Московский автопарк'),
('Больков', 30000, 'Московский автопарк'),
                                  ('Петров', 10000, 'Нижегородский автопарк'), ('Абакумов', 25000, 'Нижегородский автопарк')])

    # Тестирование задания №2
    def test_task_2(self):
        result = task_2(self.one_to_many)
        self.assertEqual(result, [('Московский автопарк', 45000), ('Нижегородский автопарк', 35000), ('Костромской
автопарк', 20000)])

    # Тестирование задания №3
    def test_task_3(self):
        word = 'Московский'
        result = task_3(self.many_to_many, word)
        self.assertEqual(result, {'Московский автопарк': ['Сидоров', 'Больков'], 'Московский (другой) автопарк':
['Петров', 'Абакумов']})

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

Результаты выполнения.

Результаты выполнения файла rk_1_refactored.py



```
Задание А1
«Автопарк» и «Водитель» связаны соотношением один-ко-многим.
Список всех связанных водителей и автопарков, отсортированный по автопаркам, сортировка по водителям произвольная:
('Иванов', 20000, 'Костромской автопарк')
('Сидоров', 15000, 'Московский автопарк')
('Больков', 30000, 'Московский автопарк')
('Петров', 10000, 'Нижегородский автопарк')
('Абакумов', 25000, 'Нижегородский автопарк')

Задание А2
«Автопарк» и «Водитель» связаны соотношением один-ко-многим.
Список отделов с суммарной зарплатой водителей в каждом автопарке, отсортированный по суммарной зарплате:
('Московский автопарк', 45000)
('Нижегородский автопарк', 35000)
('Костромской автопарк', 20000)

Задание А3
«Автопарк» и «Водитель» связаны соотношением многие-ко-многим.
Список всех автопарков, у которых в названии присутствует слово «Московский», и список работающих в них водителей:
Московский автопарк (['Сидоров', 'Больков'])
Московский (другой) автопарк (['Петров', 'Абакумов'])
```

Результаты выполнения файла rk_2.py