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).

Изначальное задание(РК1):
Вариант Г.

«Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «А», и список работающих в них сотрудников.

«Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с максимальной зарплатой сотрудников в каждом отделе, отсортированный по максимальной зарплате.

«Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по отделам, сортировка по сотрудникам произвольная.

30 Факультет Университет

Мушкарин Дмитрий Евгеньевич
ИУ5-35Б

```python
from operator import itemgetter

You, last week | 1 author (You)
class Факультет:
    """Факультет"""
    def __init__(self, id, название, зарплата_преподавателя, университет_id):
        self.id = id
        self.название = название
        self.зарплата_преподавателя = зарплата_преподавателя
        self.университет_id = университет_id

You, last week | 1 author (You)
class Университет:
    """Университет"""
    def __init__(self, id, название):
        self.id = id
        self.название = название

You, 2 hours ago | 1 author (You)
class ФакультетыУниверситета:
    """Связь многие-ко-многим"""
    def __init__(self, университет_id, факультет_id):
        self.университет_id = университет_id
        self.факультет_id = факультет_id

def one_to_many(университеты, факультеты):
    return [(f.название, f.зарплата_преподавателя, u.название)
            for u in университеты
            for f in факультеты
            if f.университет_id == u.id]

def many_to_many(университеты, факультеты, факультеты_университетов):
    temp = [(u.название, fu.университет_id, fu.факультет_id)
            for u in университеты
            for fu in факультеты_университетов
            if u.id == fu.университет_id]
    return [(f.название, f.зарплата_преподавателя, университет_название)
            for университет_название, университет_id, факультет_id in temp
            for f in факультеты if f.id == факультет_id]

def задание_Г1(data):
    return [item for item in data if item[2].startswith('A')]

def задание_Г2(университеты, one_to_many_data):
    result = []
    for u in университеты:
        факультеты = list(filter(lambda i: i[2] == u.название, one_to_many_data))
        if факультеты:
            зарплаты = [зарплата for _, зарплата, _ in факультеты]
            max_зарплата = max(зарплаты)
            result.append((u.название, max_зарплата))
    return sorted(result, key=itemgetter(1), reverse=True)

def задание_Г3(университеты, many_to_many_data):
    result = {}
    for u in университеты:
        факультеты = list(filter(lambda i: i[2] == u.название, many_to_many_data))
        факультеты_названия = [название for название, _, _ in факультеты]
        result[u.название] = факультеты_названия
    return result
```

```python
import unittest
from main import ФакультетыУниверситета, Университет, Факультет, one_to_many, задание_Г1, задание_Г2,  задание_Г3, many_to_many
```

```python
class TestФакультеты(unittest.TestCase):
    def setUp(self):
        self.университеты = [
            Университет(1, 'МГУ'),
            Университет(2, 'СПбГУ'),
            Университет(3, 'МФТИ'),
            Университет(4, 'АГУ')
        ]
        self.факультеты = [
            Факультет(1, 'Математический', 50000, 1),
            Факультет(2, 'Физический', 60000, 1),
            Факультет(3, 'Исторический', 45000, 2),
            Факультет(4, 'Биологический', 55000, 3),
            Факультет(5, 'Астрономический', 70000, 4)
        ]
        self.факультеты_университетов = [
            ФакультетыУниверситета(1, 1),
            ФакультетыУниверситета(1, 2),
            ФакультетыУниверситета(2, 3),
            ФакультетыУниверситета(3, 4),
            ФакультетыУниверситета(4, 5)
        ]
        self.one_to_many_data = one_to_many(self.университеты, self.факультеты)
        self.many_to_many_data = many_to_many(self.университеты, self.факультеты, self.факультеты_университетов)

    def test_задание_Г1(self):
        expected = [('Астрономический', 70000, 'АГУ')]
        result = задание_Г1(self.one_to_many_data)
        self.assertEqual(result, expected)

    def test_задание_Г2(self):
        expected = [('АГУ', 70000),('МГУ', 60000),  ('МФТИ', 55000), ('СПбГУ', 45000)]
        result = задание_Г2(self.университеты, self.one_to_many_data)
        self.assertEqual(result, expected)

    def test_задание_Г3(self):
        expected = {
            'МГУ': ['Математический', 'Физический'],
            'СПбГУ': ['Исторический'],
            'МФТИ': ['Биологический'],
            'АГУ': ['Астрономический']
        }
        result = задание_Г3(self.университеты, self.many_to_many_data)
        self.assertEqual(result, expected)

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


```
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.3, pluggy-1.5.0
rootdir: /home/user/Documents/py_sem3/rk2
plugins: anyio-4.4.0
collected 3 items

test_main.py ...                                                         [100%]

============================== 3 passed in 0.01s ===============================
```