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»

Курс «Базовые компоненты интернет-технологий»

Рубежный контроль №2

Вариант 17

(Вариант В, классы – Дирижёр, Оркестр)

Выполнил:

студент группы ИУ5-33Б

Александр Саргсян

Проверил:

к.т.н., доц., Ю. Е. Гапанюк

2022 г.

Разработка тестов

1) Провести рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создать модульные тесты с применением TDD – фреймворка (3 теста).

Текст программы

main_RK1.py:

```
from operator import itemgetter


class Cond:
    """Дирижёр"""

    def __init__(self, id, fio, sal, orch_id):
        self.id = id
        self.fio = fio
        self.sal = sal
        self.orch_id = orch_id

class Orch:
    """Оркестр"""

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

class CondOrch:
    def __init__(self, orch_id, cond_id):
        self.orch_id = orch_id
        self.cond_id = cond_id


def connections(orchs, conds, conds_orchs):
    # Соединение данных один-ко-многим
    one_to_many = [(c.fio, c.sal, o.name)
                   for o in orchs
                   for c in conds
                   if c.orch_id == o.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(o.name, co.orch_id, co.cond_id)
                         for o in orchs
                         for co in conds_orchs
                         if o.id == co.orch_id]

    many_to_many = [(c.fio, c.sal, orch_name)
                    for orch_name, orch_id, cond_id in many_to_many_temp
                    for c in conds if c.id == cond_id]
    return one_to_many, many_to_many

def task1(orchs, conds, conds_orchs):
    one_to_many, many_to_many = connections(orchs, conds, conds_orchs)
    res_11 = {}
    for c in conds:
        if c.fio[0] == 'A':
```

```
            c_orchs = list(filter(lambda i: i[0] == c.fio, one_to_many))
            c_orchs_names = c_orchs[0][2]
            res_11[c.fio] = c_orchs_names
    return (res_11)

def task2(orchs, conds, conds_orchs):
    one_to_many, many_to_many = connections(orchs, conds, conds_orchs)
    res_12_unsorted = []
    for o in orchs:
        o_conds = list(filter(lambda i: i[2] == o.name, one_to_many))
        if len(o_conds) > 0:
            o_sals = [sal for _, sal, _ in o_conds]
            min_sal = sorted(o_sals)
            res_12_unsorted.append((o.name, min_sal[0]))
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return(res_12)

def task3(orchs, conds, conds_orchs):
    one_to_many, many_to_many = connections(orchs, conds, conds_orchs)
    res_13 = sorted(many_to_many, key=itemgetter(0))
    return(res_13)

def main():
    orchs = [
        Orch(1, 'симфонический оркестр'),
        Orch(2, 'струнный оркестр'),
        Orch(3, 'духовой оркестр'),
        Orch(4, 'джазовый оркестр'),
        Orch(5, 'эстрадный оркестр'),
        Orch(6, 'камерный оркестр'),
    ]

    conds = [
        Cond(1, 'Андреев', 45000, 1),
        Cond(2, 'Иванов', 35000, 2),
        Cond(3, 'Станков', 45000, 3),
        Cond(4, 'Ноткин', 30000, 3),
        Cond(5, 'Алексеев', 50000, 3),
    ]

    conds_orchs = [
        CondOrch(1, 1),
        CondOrch(2, 2),
        CondOrch(3, 3),
        CondOrch(3, 4),
        CondOrch(3, 5),

        CondOrch(4, 1),
        CondOrch(5, 2),
        CondOrch(5, 3),
        CondOrch(6, 4),
        CondOrch(6, 5),
    ]

    print('Задание B1')
    print(task1(orchs, conds, conds orchs))
    print('\nЗадание B2')
    print(task2(orchs, conds, conds_orchs))
    print('\nЗадание B3')
    print(task3(orchs, conds, conds_orchs))

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

main.py:

```
rom main_RK1 import *
import unittest

class RK2_test(unittest.TestCase):
    orchs = [
        Orch(1, 'симфонический оркестр'),
        Orch(2, 'струнный оркестр'),
        Orch(3, 'духовой оркестр'),
        Orch(4, 'джазовый оркестр'),
        Orch(5, 'эстрадный оркестр'),
        Orch(6, 'камерный оркестр'),
    ]

    conds = [
        Cond(1, 'Андреев', 45000, 1),
        Cond(2, 'Иванов', 35000, 2),
        Cond(3, 'Станков', 45000, 3),
        Cond(4, 'Ноткин', 30000, 3),
        Cond(5, 'Алексеев', 50000, 3),
    ]

    conds_orchs = [
        CondOrch(1, 1),
        CondOrch(2, 2),
        CondOrch(3, 3),
        CondOrch(3, 4),
        CondOrch(3, 5),

        CondOrch(4, 1),
        CondOrch(5, 2),
        CondOrch(5, 3),
        CondOrch(6, 4),
        CondOrch(6, 5),
    ]

    def test1(self):
        result = {'Андреев': 'симфонический оркестр', 'Алексеев': 'духовой
оркестр'}
        self.assertEqual(task1(RK2_test.orchs, RK2_test.conds,
RK2_test.conds_orchs), result)

    def test2(self):
        result = [('симфонический оркестр', 45000), ('струнный оркестр',
35000), ('духовой оркестр', 30000)]
        self.assertEqual(task2(RK2_test.orchs, RK2_test.conds,
RK2_test.conds_orchs), result)

    def test3(self):
        result = [
            ('Алексеев', 50000, 'духовой оркестр'), ('Алексеев', 50000,
'камерный оркестр'),
            ('Андреев', 45000, 'симфонический оркестр'), ('Андреев', 45000,
'джазовый оркестр'),
            ('Иванов', 35000, 'струнный оркестр'), ('Иванов', 35000,
'эстрадный оркестр'),
            ('Ноткин', 30000, 'духовой оркестр'), ('Ноткин', 30000, 'камерный
оркестр'),
            ('Станков', 45000, 'духовой оркестр'), ('Станков', 45000,
'эстрадный оркестр')]
        self.assertEqual(task3(RK2_test.orchs, RK2_test.conds,
RK2_test.conds_orchs), result)
```

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

Результат выполнения

```
============================= test session starts ==============================
collecting ... collected 3 items

main.py::RK2_test::test1 PASSED                                          [ 33%]
main.py::RK2_test::test2 PASSED                                          [ 66%]
main.py::RK2_test::test3 PASSED                                          [100%]

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

Process finished with exit code 0
```