МГТУ им. Н.Э. Баумана

Кафедра «Системы обработки информации иуправления»

Рубежный контроль №2 «Базовые компоненты интернет-технологий»

Студентка группы ИУ5-31Б:

Саркисян Стелла Зограбовна

Преподаватель кафедры ИУ5:Гапанюк Юрий Евгеньевич

Москва, 2022

текст программы рк1:

```
from operator import itemgetter

class Cond:
    """Дирижер"""
    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:
    """
    'Дирижеры оркестра' для реализации
    связи многие-ко-многим
    """
    def __init__(self, orch_id, cond_id):
        self.orch_id = orch_id
        self.cond_id = cond_id

orchs = [
    Orch(1, 'филармонический'),
    Orch(2, 'народный оркестр'),
    Orch(3, 'хоровой'),

    Orch(11, 'духовой оркестр'),
    Orch(22, 'джазисты'),
    Orch(33, 'симфонический'),
]

conds = [
    Cond(1, 'Сидоров', 25000, 1),
    Cond(2, 'Петров', 35000, 2),
    Cond(3, 'Иваненко', 45000, 3),
    Cond(4, 'Сорокина', 35000, 3),
    Cond(5, 'Иванин', 25000, 3),
]

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

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

def one_to_many(orchs, conds):
    return [(c.fio, c.sal, o.name)
        for o in orchs
        for c in conds
        if c.orch_id == o.id]

def many_to_many(orchs, conds):
    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]
    return [(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]

def A1(orchs, conds) -> list:
    res_11 = sorted(one_to_many(orchs, conds), key=itemgetter(2))
    return(res_11)

def A2(orchs, conds) -> list:
    res_12_unsorted = []
    for o in orchs:
        o_conds = list(filter(lambda i: i[2] == o.name,
one_to_many(orchs, conds)))
        if len(o_conds) > 0:
            o_sals = [sal for _,sal,_ in o_conds]
            o_sals_sum = sum(o_sals)
            res_12_unsorted.append((o.name, o_sals_sum))

    res_12 = sorted(res_12_unsorted, key=itemgetter(1),
reverse=True)
    return(res_12)

def A3(orchs, conds, str_to_find) -> list:
    res_13 = {}
    for o in orchs:
        if str_to_find in o.name:
            o_conds = list(filter(lambda i: i[2]==o.name,
many_to_many(orchs, conds)))
            o_conds_names = [x for x,_,_ in o_conds]
            res_13[o.name] = o_conds_names

    return(res_13)

if __name__ == '__main__':
```

```
    print('Задание А1')
    print(A1(orchs, conds))

    print('\nЗадание А2')
    print(A2(orchs, conds))

    print('\nЗадание А3')
    print(A2(orchs, conds, 'оркестр'))
```

текст программы рк2:

```
import unittest
from rk1_copy import Cond, Orch, CondOrch, A1, A2, A3

class test(unittest.TestCase):

    def setUp(self):
        self.streets = [
            Orch(1, 'филармонический'),
            Orch(2, 'народный оркестр'),
            Orch(3, 'хоровой'),

            Orch(11, 'духовой оркестр'),
            Orch(22, 'джазисты'),
            Orch(33, 'симфонический'),
        ]
        self.houses = [
            Cond(1, 'Сидоров', 25000, 1),
            Cond(2, 'Петров', 35000, 2),
            Cond(3, 'Иваненко', 45000, 3),
            Cond(4, 'Сорокина', 35000, 3),
            Cond(5, 'Иванин', 25000, 3),
        ]
        self.houses_streets = [
            CondOrch(1,1),
            CondOrch(2,2),
            CondOrch(3,3),
            CondOrch(3,4),
            CondOrch(3,5),
            CondOrch(11,1),
            CondOrch(22,2),
            CondOrch(33,3),
            CondOrch(33,4),
            CondOrch(33,5),
        ]


    def test_A1(self):
        expected_result = [
            ('Петров', 35000, 'народный оркестр'),
```

```
            ('Сидоров', 25000, 'филармонический'),
            ('Иваненко', 45000, 'хоровой'),
            ('Сорокина', 35000, 'хоровой'),
            ('Иванин', 25000, 'хоровой')
        ]

        result = A1(self.streets, self.houses)
        self.assertEqual(result, expected_result)

    def test_A2(self):
        expected_result = [
            ('хоровой', 105000),
            ('народный оркестр', 35000),
            ('филармонический', 25000)
        ]
        result = A2(self.streets, self.houses)
        self.assertEqual(result, expected_result)

    def test_A3(self):
        expected_result = {'народный оркестр': ['Петров'],
'духовой оркестр': ['Сидоров']}
        result = A3(self.streets, self.houses, 'оркестр')
        self.assertEqual(result, expected_result)

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

Результат выполнения:

```
Ran 3 tests in 0.001s

OK
```