Московский государственный технический университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»
Отчет по рубежному контролю №2
«**Вариант В, 17**»

Выполнила:
Студентка группы ИУ5-34Б
Глозман Варвара

Проверил:
Гапанюк Ю. Е.

2025 г.

Листинг программы
rk1.py(после рефакторинга):

```python
from operator import itemgetter

class Conductor:
    def __init__(self, id, fio, salary, orchestra_id):
        self.id = id
        self.fio = fio
        self.salary = salary
        self.orchestra_id = orchestra_id

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

class ConductorOrchestra:
    def __init__(self, orchestra_id, conductor_id):
        self.orchestra_id = orchestra_id
        self.conductor_id = conductor_id

orchestras = [
    Orchestra(1, "Симфонический оркестр"),
    Orchestra(2, "Филармонический оркестр"),
    Orchestra(3, "Камерный оркестр"),
    Orchestra(4, "Академический оркестр"),
]

conductors = [
    Conductor(1, "Антонов", 50000, 1),
    Conductor(2, "Петров", 60000, 2),
    Conductor(3, "Алексеев", 55000, 1),
    Conductor(4, "Сидоров", 70000, 3),
    Conductor(5, "Андреев", 45000, 2),
]

conductors_orchestras = [
    ConductorOrchestra(1, 1),
    ConductorOrchestra(2, 2),
    ConductorOrchestra(1, 3),
    ConductorOrchestra(3, 4),
    ConductorOrchestra(2, 5),
    ConductorOrchestra(4, 1),
    ConductorOrchestra(4, 4),
]

def get_one_to_many(orchestras, conductors):
    return [(c.fio, c.salary, o.name)
            for o in orchestras
            for c in conductors
            if c.orchestra_id == o.id]
```

```python
def get_many_to_many(orchestras, conductors_orchestras, conductors):
    many_to_many_temp = [(o.name, co.orchestra_id, co.conductor_id)
                         for o in orchestras
                         for co in conductors_orchestras
                         if o.id == co.orchestra_id]

    return [(c.fio, c.salary, orchestra_name)
            for orchestra_name, orchestra_id, conductor_id in many_to_many_temp
            for c in conductors if c.id == conductor_id]


def solve_b1(one_to_many):
    return list(filter(lambda i: i[0].startswith('A'), one_to_many))

def solve_b2(one_to_many, orchestras):
    res_2_unsorted = []
    for o in orchestras:
        o_salaries = list(filter(lambda i: i[2] == o.name, one_to_many))
        if len(o_salaries) > 0:
            min_salary = min([sal for _, sal, _ in o_salaries])
            res_2_unsorted.append((o.name, min_salary))

    return sorted(res_2_unsorted, key=itemgetter(1))

def solve_b3(many_to_many):
    return sorted(many_to_many, key=itemgetter(0))

def main():
    one_to_many = get_one_to_many(orchestras, conductors)
    many_to_many = get_many_to_many(orchestras, conductors_orchestras, conductors)

    print("Задание B1:")
    print(solve_b1(one_to_many))

    print("\nЗадание B2:")
    print(solve_b2(one_to_many, orchestras))

    print("\nЗадание B3:")
    print(solve_b3(many_to_many))

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

test_rk2.py:

```python
import unittest
from rk1 import orchestras, conductors, conductors_orchestras
from rk1 import get_one_to_many, get_many_to_many
from rk1 import solve_b1, solve_b2, solve_b3

class TestRk2(unittest.TestCase):
```

```
    @classmethod
    def setUpClass(cls):
        cls.one_to_many = get_one_to_many(orchestras, conductors)
        cls.many_to_many = get_many_to_many(orchestras, conductors_orchestras,
conductors)

    def test_b1_filter_by_name(self):
        result = solve_b1(self.one_to_many)
        for item in result:
            self.assertTrue(item[0].startswith('A'), f"Фамилия {item[0]} не начинается
на 'A'")

        expected_surnames = ['Антонов', 'Алексеев', 'Андреев']
        actual_surnames = [item[0] for item in result]
        self.assertEqual(actual_surnames, expected_surnames)

    def test_b2_min_salary(self):
        result = solve_b2(self.one_to_many, orchestras)

        expected = [
            ('Филармонический оркестр', 45000),
            ('Симфонический оркестр', 50000),
            ('Камерный оркестр', 70000)
        ]
        self.assertEqual(result, expected)

        salaries = [r[1] for r in result]
        self.assertEqual(salaries, sorted(salaries), "Список не отсортирован по
зарплате")

    def test_b3_sort_by_name(self):
        result = solve_b3(self.many_to_many)
        names = [item[0] for item in result]
        self.assertEqual(names, sorted(names), "Список не отсортирован по фамилиям")

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

Результат выполнения:

```
Задание B1:
[('Антонов', 50000, 'Симфонический оркестр'), ('Алексеев', 55000, 'Симфонический оркестр'), ('Андреев', 45000, 'Филармонический оркестр')]

Задание B2:
[('Филармонический оркестр', 45000), ('Симфонический оркестр', 50000), ('Камерный оркестр', 70000)]

Задание B3:
[('Алексеев', 55000, 'Симфонический оркестр'), ('Андреев', 45000, 'Филармонический оркестр'), ('Антонов', 50000, 'Симфонический оркестр'), ('Антонов', 50000, 'Академический оркестр'),
('Петров', 60000, 'Филармонический оркестр'), ('Сидоров', 70000, 'Камерный оркестр'), ('Сидоров', 70000, 'Академический оркестр')]
```

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```