music.py

```
from operator import itemgetter


class MusicalPiece:
    """Музыкальное произведение"""

    def __init__(self, id, title, duration, orchestra_id):
        self.id = id
        self.title = title
        self.duration = duration
        self.orchestra_id = orchestra_id


class Orchestra:
    """Оркестр"""

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


class MusicOrch:
    """Музыкальные произведения оркестра для реализации связи
многие-ко-многим"""

    def __init__(self, orchestra_id, music_id):
        self.orchestra_id = orchestra_id
        self.music_id = music_id


def get_one_to_many(orch, mus):
    return [(m.title, m.duration, o.name)
            for o in orch
            for m in mus
            if m.orchestra_id == o.id]


def get_many_to_many(orch, mus, music_orch):
    many_to_many_temp = [(o.name, mo.orchestra_id, mo.music_id)
                         for o in orch
                         for mo in music_orch
                         if o.id == mo.orchestra_id]

    return [(m.title, m.duration, orchestra_name)
            for orchestra_name, orchestra_id, music_id in
many_to_many_temp
```

```
            for m in mus if m.id == music_id]


def task_a1(one_to_many):
    return sorted(one_to_many, key=itemgetter(2))


def task_a2(one_to_many, orch):
    result = []
    for orc in orch:
        temp_orc = list(filter(lambda i: i[2] == orc.name,
one_to_many))
        total_duration = sum(i[1] for i in temp_orc)
        if temp_orc:
            result.append((orc.name, total_duration))

    return sorted(result, key=itemgetter(1), reverse=True)


def task_a3(many_to_many, orch):
    result = {}
    for o in orch:
        if 'Оркестр' in o.name:
            this_orc_music = list(filter(lambda i: i[2] == o.name,
many_to_many))
            title_list = [i[0] for i in this_orc_music]
            result[o.name] = title_list

    return result


def main(orch, mus, music_orch):
    one_to_many = get_one_to_many(orch, mus)
    many_to_many = get_many_to_many(orch, mus, music_orch)

    return {
        "task_a1": task_a1(one_to_many),
        "task_a2": task_a2(one_to_many, orch),
        "task_a3": task_a3(many_to_many, orch),
    }
```

test.py

```
import unittest
from music import MusicalPiece, Orchestra, MusicOrch,
get_one_to_many, get_many_to_many, task_a1, task_a2, task_a3
```

```python
class TestMusic(unittest.TestCase):
    def setUp(self):
        """Инициализация данных для тестов"""
        self.orch = [
            Orchestra(1, 'Оркестр имени Чайковского'),
            Orchestra(2, 'Народный Оркестр'),
            Orchestra(3, 'Камерный оркестр Карелии'),
        ]

        self.mus = [
            MusicalPiece(1, 'Симфония №1', 120, 1),
            MusicalPiece(2, 'Симфония №3', 40, 2),
            MusicalPiece(3, 'Увертюра', 15, 3),
        ]

        self.music_orch = [
            MusicOrch(1, 1),
            MusicOrch(2, 2),
            MusicOrch(3, 3),
        ]

    def test_task_a1(self):
        one_to_many = get_one_to_many(self.orch, self.mus)
        result = task_a1(one_to_many)
        expected = sorted([
            ('Симфония №1', 120, 'Оркестр имени Чайковского'),
            ('Симфония №3', 40, 'Народный Оркестр'),
            ('Увертюра', 15, 'Камерный оркестр Карелии'),
        ], key=lambda x: x[2])  # Ожидаемый результат отсортирован
по названию оркестра
        self.assertEqual(result, expected)

    def test_task_a2(self):
        one_to_many = get_one_to_many(self.orch, self.mus)
        result = task_a2(one_to_many, self.orch)
        expected = sorted([
            ('Оркестр имени Чайковского', 120),
            ('Народный Оркестр', 40),
            ('Камерный оркестр Карелии', 15),
        ], key=lambda x: x[1], reverse=True)  # Сортировка по
длительности произведений
        self.assertEqual(result, expected)

    def test_task_a3(self):
        many_to_many = get_many_to_many(self.orch, self.mus,
self.music_orch)
        result = task_a3(many_to_many, self.orch)
        expected = {
```

```
            'Оркестр имени Чайковского': ['Симфония №1'],
            'Народный Оркестр': ['Симфония №3'],
        }  # Учитываются только оркестры, содержащие "Оркестр" в названии
        self.assertDictEqual(result, expected)


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

результат
Testing started at 19:13 ...
Launching unittests with arguments python -m unittest C:\Users\annfr\PycharmProjects\rk1_Orlova_IU5-31B\test.py in C:\Users\annfr\PycharmProjects\rk1_Orlova_IU5-31B

Ran 3 tests in 0.002s

OK

Process finished with exit code 0