Московский государственный технический университет
им. Н.Э. Баумана

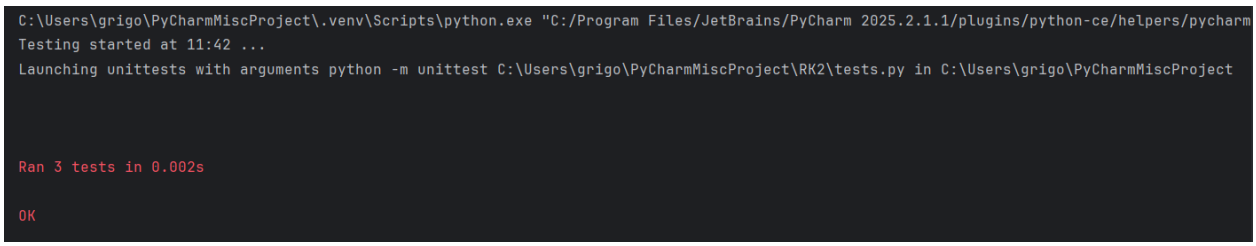
Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю №2
“Вариант Д-18”

Выполнил:
студент группы ИУ5-36Б
Юдин Григорий Олегович
Проверил:
Гапанюк Юрий Евгеньевич

Москва, 2025



RK1_refactored.py

```
from operator import itemgetter


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


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


class WorkOrchestra:
    def __init__(self, orchestra_id, work_id):
        self.orchestra_id = orchestra_id
        self.work_id = work_id


# Тестовые данные
orchestras = [
    Orchestra(1, "Академический оркестр"),
    Orchestra(2, "Симфонический оркестр"),
    Orchestra(3, "Камерный ансамбль"),
    Orchestra(4, "Альтернативный оркестр"),
]

works = [
    MusicalWork(1, "Вальсов", 7.5, 1),
    MusicalWork(2, "Маршов", 5.0, 2),
    MusicalWork(3, "Реквием", 12.0, 3),
    MusicalWork(4, "Концертов", 10.0, 1),
    MusicalWork(5, "Сюита", 8.0, 2),
]

works_orchestras = [
    WorkOrchestra(1, 1),
    WorkOrchestra(1, 4),
```

```
    WorkOrchestra(2, 2),
    WorkOrchestra(2, 5),
    WorkOrchestra(3, 3),
    WorkOrchestra(4, 1),
    WorkOrchestra(4, 2),
]


def get_one_to_many(orchestras, works):
    return [
        (w.title, w.duration, o.name)
        for o in orchestras
        for w in works
        if w.orchestra_id == o.id
    ]


def get_many_to_many(orchestras, works, works_orchestras):
    temp = [
        (o.name, wo.orchestra_id, wo.work_id)
        for o in orchestras
        for wo in works_orchestras
        if o.id == wo.orchestra_id
    ]

    return [
        (w.title, w.duration, orchestra_name)
        for orchestra_name, _, work_id in temp
        for w in works
        if w.id == work_id
    ]


# Задание Д1
def query_d1(one_to_many):
    return [
        (title, orchestra)
        for title, _, orchestra in one_to_many
        if title.endswith("ов")
    ]


# Задание Д2
def query_d2(one_to_many, orchestras):
    result = []
    for o in orchestras:
        o_works = list(filter(lambda x: x[2] == o.name, one_to_many))
        if o_works:
            durations = [dur for _, dur, _ in o_works]
            avg = sum(durations) / len(durations)
            result.append((o.name, avg))
    return sorted(result, key=itemgetter(1))
```

```
# Задание Д3
def query_d3(many_to_many, orchestras):
    result = {}
    for o in orchestras:
        if o.name.startswith("А"):
            o_works = list(filter(lambda x: x[2] == o.name, many_to_many))
            result[o.name] = [title for title, _, _ in o_works]
    return result
```

tests.py

```
import unittest
from RK2.RK1_refactored import (
    orchestras, works, works_orchestras,
    get_one_to_many, get_many_to_many,
    query_d1, query_d2, query_d3
)


class TestMusicQueries(unittest.TestCase):

    def setUp(self):
        self.one_to_many = get_one_to_many(orchestras, works)
        self.many_to_many = get_many_to_many(orchestras, works, works_orchestras)

    def test_query_d1_titles_ending_with_ов(self):
        result = query_d1(self.one_to_many)

        expected = {
            ("Вальсов", "Академический оркестр"),
            ("Маршов", "Симфонический оркестр"),
            ("Концертов", "Академический оркестр"),
        }

        self.assertEqual(set(result), expected)

    def test_query_d2_average_duration(self):
        result = query_d2(self.one_to_many, orchestras)

        orch_dict = dict(result)

        self.assertAlmostEqual(orch_dict["Камерный ансамбль"], 12.0)
        self.assertAlmostEqual(orch_dict["Симфонический оркестр"], 6.5)
        self.assertAlmostEqual(orch_dict["Академический оркестр"], 8.75)

    def test_query_d3_orchestras_starting_with_а(self):
        result = query_d3(self.many_to_many, orchestras)
```

```
        self.assertIn("Академический оркестр", result)
        self.assertIn("Альтернативный оркестр", result)
        self.assertIn("Вальсов", result["Академический оркестр"])


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