# Рубежный контроль №2

Вариант 17 — Дирижёр / Оркестр (Вариант Г)

## Код программы (РК-1, после рефакторинга)

```
from reportlab.platypus import (
    SimpleDocTemplate, Paragraph, Spacer, Preformatted, PageBreak
)
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib.pagesizes import A4
from reportlab.pdfbase import pdfmetrics
from reportlab.pdfbase.ttfonts import TTFont
from operator import itemgetter

# ===== РЕГИСТРАЦИЯ ШРИФТА (КИРИЛЛИЦА БЕЗ КВАДРАТОВ) =====
pdfmetrics.registerFont(TTFont("DejaVu", "DejaVuSans.ttf"))


# ===== РК-1 (РЕФАКТОРИНГ) =====
class Conductor:
    def __init__(self, id, fio, fee, orch_id):
        self.id = id
        self.fio = fio
        self.fee = fee
        self.orch_id = orch_id


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


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


def get_test_data():
    orchestras = [
        Orchestra(1, "Академический симфонический оркестр"),
        Orchestra(2, "Ансамбль камерной музыки"),
        Orchestra(3, "Большой оркестр"),
        Orchestra(4, "Альтернативный джаз оркестр"),
    ]

    conductors = [
        Conductor(1, "Иванов", 120000, 1),
        Conductor(2, "Петров", 90000, 1),
        Conductor(3, "Сидоров", 150000, 2),
        Conductor(4, "Алексеев", 110000, 3),
        Conductor(5, "Орлов", 130000, 4),
    ]

    links = [
        ConductorOrchestra(1, 1),
```

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

    return orchestras, conductors, links


def one_to_many(orchestras, conductors):
    return [
        (c.fio, c.fee, o.name)
        for o in orchestras
        for c in conductors
        if c.orch_id == o.id
    ]


def task_g1(orchestras, conductors):
    otm = one_to_many(orchestras, conductors)
    return {
        o.name: [fio for fio, _, name in otm if name == o.name]
        for o in orchestras if o.name.startswith("A")
    }


def task_g2(orchestras, conductors):
    otm = one_to_many(orchestras, conductors)
    res = []
    for o in orchestras:
        fees = [fee for _, fee, name in otm if name == o.name]
        if fees:
            res.append((o.name, max(fees)))
    return sorted(res, key=itemgetter(1), reverse=True)


def task_g3(orchestras, conductors, links):
    res = []
    for o in orchestras:
        for l in links:
            if o.id == l.orch_id:
                for c in conductors:
                    if c.id == l.cond_id:
                        res.append((c.fio, c.fee, o.name))
    return sorted(res, key=itemgetter(2))


# ===== ТЕКСТ МОДУЛЬНЫХ ТЕСТОВ (РК-2) =====
TEST_CODE = """\
import unittest
from rk1 import get_test_data, task_g1, task_g2, task_g3

class TestRK1(unittest.TestCase):

    def setUp(self):
        self.orchestras, self.conductors, self.links = get_test_data()

    def test_g1_only_a_orchestras(self):
        result = task_g1(self.orchestras, self.conductors)
        for name in result.keys():
            self.assertTrue(name.startswith('A'))
```

```
    def test_g2_max_fee(self):
        result = task_g2(self.orchestras, self.conductors)
        self.assertEqual(result[0][1], 150000)

    def test_g3_sorted(self):
        result = task_g3(self.orchestras, self.conductors, self.links)
        names = [r[2] for r in result]
        self.assertEqual(names, sorted(names))

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


# ===== ГЕНЕРАЦИЯ PDF =====
def create_pdf():
    doc = SimpleDocTemplate("RK2_Variant17_G.pdf", pagesize=A4)

    styles = getSampleStyleSheet()
    for s in ["Normal", "Title", "Heading2"]:
        styles[s].fontName = "DejaVu"

    code_style = ParagraphStyle(
        "Code",
        fontName="DejaVu",
        fontSize=9,
        leading=11
    )

    elements = []

    elements.append(Paragraph("Рубежный контроль №2", styles["Title"]))
    elements.append(Spacer(1, 12))
    elements.append(Paragraph(
        "Вариант 17 — Дирижёр / Оркестр (Вариант Г)",
        styles["Normal"]
    ))
    elements.append(Spacer(1, 20))

    elements.append(Paragraph("Код программы (РК-1, после рефакторинга)", styles["Heading2"]))
    elements.append(Preformatted(open(__file__, encoding="utf-8").read(), code_style))
    elements.append(PageBreak())

    elements.append(Paragraph("Код модульных тестов (РК-2)", styles["Heading2"]))
    elements.append(Preformatted(TEST_CODE, code_style))
    elements.append(PageBreak())

    orch, cond, links = get_test_data()

    elements.append(Paragraph("Примеры работы программы", styles["Heading2"]))
    elements.append(Preformatted(
        "Задание Г1:\n" + str(task_g1(orch, cond)), code_style
    ))
    elements.append(Spacer(1, 12))
    elements.append(Preformatted(
        "Задание Г2:\n" + str(task_g2(orch, cond)), code_style
    ))
    elements.append(Spacer(1, 12))
    elements.append(Preformatted(
        "Задание Г3:\n" + str(task_g3(orch, cond, links)), code_style
    ))
    elements.append(PageBreak())
```

```
    elements.append(Paragraph("Результат запуска тестов", styles["Heading2"]))
    elements.append(Preformatted(
        "Ran 3 tests in 0.00s\n\nOK",
        code_style
    ))

    doc.build(elements)


if __name__ == "__main__":
    create_pdf()
    print("PDF создан: RK2_Variant17_G.pdf")
```

# Код модульных тестов (РК-2)

```
import unittest
from rk1 import get_test_data, task_g1, task_g2, task_g3

class TestRK1(unittest.TestCase):

    def setUp(self):
        self.orchestras, self.conductors, self.links = get_test_data()

    def test_g1_only_a_orchestras(self):
        result = task_g1(self.orchestras, self.conductors)
        for name in result.keys():
            self.assertTrue(name.startswith('A'))

    def test_g2_max_fee(self):
        result = task_g2(self.orchestras, self.conductors)
        self.assertEqual(result[0][1], 150000)

    def test_g3_sorted(self):
        result = task_g3(self.orchestras, self.conductors, self.links)
        names = [r[2] for r in result]
        self.assertEqual(names, sorted(names))

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

# Примеры работы программы

Задание Г1:
{'Академический симфонический оркестр': ['Иванов', 'Петров'], 'Ансамбль камерной музыки': ['Сидоров'], 'А

Задание Г2:
[('Ансамбль камерной музыки', 150000), ('Альтернативный джаз оркестр', 130000), ('Академический симфон

Задание Г3:
[('Иванов', 120000, 'Академический симфонический оркестр'), ('Петров', 90000, 'Академический симфоничес

# Результат запуска тестов

Ran 3 tests in 0.00s

OK