Условия рубежного контроля №2 по курсу ПиК ЯП

Рубежный контроль представляет собой разработку тестов на языке Python.

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

Вариант 5:

Класс 1: Музыкант
Класс 2: Оркестр

Вариант Д.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате (*отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений*).
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «А», и список работающих в них сотрудников.

Текст программы:

```
import unittest


class Orchestra:
    def __init__(self, name):
        self.name = name
        self.musicians = []

    def mus_add(self, musician):
```

```
        if musician not in self.musicians:
            self.musicians.append(musician)
            musician.orch_add(self)

    def avrg_salary(self):
        if not self.musicians:
            return 0
        united_salary = sum(musician.salary for musician in self.musicians)
        mus_count = len(self.musicians)
        return united_salary / mus_count


class Musician:
    def __init__(self, surname, salary):
        self.surname = surname
        self.salary = salary
        self.orchs = []

    def orch_add(self, orchestra):
        if orchestra not in self.orchs:
            self.orchs.append(orchestra)


def mus_endswith_ov(orchs):
    result = []
    for orchestra in orchs:
        for musician in orchestra.musicians:
            if musician.surname.endswith("ов"):
                result.append((musician.surname, orchestra.name))
    return result


def avrg_orch_salary(orchs):
    avrg_salaries = []
    for orchestra in orchs:
        avrg = orchestra.avrg_salary()
        avrg_salaries.append((orchestra.name, avrg))
    avrg_salaries.sort(key=lambda x: x[1], reverse=True)
    return avrg_salaries


def orch_startswith_A(orchs):
    result = []
    for orchestra in orchs:
        if orchestra.name.startswith("A"):
            result.append((orchestra.name, [musician.surname for musician in
```

```
orchestra.musicians]))
    return result


class TestOrchestraFunctions(unittest.TestCase):

    def setUp(self):

        self.orchestra1 = Orchestra("Академический оркестр")
        self.orchestra2 = Orchestra("Симфонический оркестр")
        self.orchestra3 = Orchestra("Аркестр Анатолия")


        self.musician1 = Musician("Иванов", 50000)
        self.musician2 = Musician("Петров", 60000)
        self.musician3 = Musician("Сидоров", 70000)
        self.musician4 = Musician("Мушкарин", 180000)


        self.orchestra1.mus_add(self.musician1)
        self.orchestra1.mus_add(self.musician2)
        self.orchestra3.mus_add(self.musician3)
        self.orchestra3.mus_add(self.musician4)

    def test_mus_endswith_ov(self):
        """Тестирует фильтрацию музыкантов по фамилии, заканчивающейся на
'ов'."""
        result = mus_endswith_ov([self.orchestra1, self.orchestra2,
self.orchestra3])
        expected = [('Иванов', 'Академический оркестр'),('Петров',
'Академический оркестр') ,('Сидоров', 'Аркестр Анатолия')]
        self.assertEqual(result, expected)

    def test_avrg_orch_salary(self):
        """Тестирует вычисление средней зарплаты оркестров."""
        result = avrg_orch_salary([self.orchestra1, self.orchestra2,
self.orchestra3])
        expected_orchestra1_average = (50000 + 60000) / 2  # 55000
        expected_orchestra3_average = (70000 + 180000) / 2  # 125000

        self.assertEqual(len(result), 3)  # Мы ожидаем 3 оркестра в списке
        self.assertAlmostEqual(result[0][1], expected_orchestra3_average,
places=2)  # highest average first
        self.assertAlmostEqual(result[1][1], expected_orchestra1_average,
places=2)  # then orchestra1
        self.assertEqual(result[2][1], 0)  # orchestra2 has no musicians
```

```
    def test_orch_startswith_A(self):
        """Тестирует фильтрацию оркестров, названия которых начинаются на
'A'."""
        result = orch_startswith_A([self.orchestra1, self.orchestra2,
self.orchestra3])
        expected = [
            ("Академический оркестр", ["Иванов", "Петров"]),
            ("Аркестр Анатолия", ["Сидоров", "Мушкарин"])
        ]
        self.assertEqual(result, expected)  # Проверяем, совпадает ли
результат с ожидаемым


if __name__ == '__main__':
    unittest.RK2()
```