Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана»

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Отчёт по РК №2

Выполнила:
студентка группы ИУ5-35Б
Сухова Мария Андреевна
Подпись:

Дата:

Проверил:
Преподаватель кафедры ИУ5
Гапанюк Юрий Евгеньевич
Подпись:

Дата:

2022 г.

Полученное задание:
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 Д 18

18	Музыкальное произведение	Оркестр

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 «Р», и список работающих в них сотрудников.
Текст программы
Программа разделена на несколько файлов

1) Файл classes.py содержит классы для реализации задания:

```
class music:
    """музыкальное произведение"""

    def __init__(self, id, author, name, music_id, duration):
        self.id = id
        self.author = author
        self.name = name
        self.music_id = music_id
        self.duration = duration

class orchestra:
    """оркестр"""

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


class orchMusic:
    def __init__(self, id_music, id_orch):
        self.id_music = id_music
        self.id_orch = id_orch
```

2) Файл connections.py содержит функции для создания связей один-комногим и многие-ко-многим:

```
def one_to_many(musics, orchectras):
    return [(mus.name, mus.duration, orch.name)
            for mus in musics
            for orch in orchectras
            if mus.orch_id == orch.id]


def many_to_many(musics, orchectras, orchMusic):
    return [(mus.surname, mus.duration, orch.name)
            for oc in orchMusic
```

```
            for mus in musics
            for orch in orchectras
            if mus.id == oc.id_mus and orch.id == oc.id_orch]
```

3) Файл processing.py содержит функции для реализации требуемых заданий:

```
def task1(one_to_many):
    return [el for el in one_to_many if el[0].endswith('ов')]


def task2(one_to_many):
    orch = []
    duration = []
    for el in one_to_many:
        if el[2] not in orch:
            orch.append(el[2])
            duration.append((el[1], 1))
        else:
            idx = orch.index(el[2])
            duration[idx] = ((duration[idx][0] + el[1]) /
(duration[idx][1] + 1), duration[idx][1] + 1)
    salary = [i[0] for i in duration]
    return sorted(list(zip(orch, duration)), key=lambda p: p[1],
reverse=True)


def task3(many_to_many):
    orch_R = []
    mus_orch_R = []
    for el in many_to_many:
        if el[2].startswith('P'):
            if el[2] not in orch_R:
                orch_R.append(el[2])
                mus_orch_R.append((el[0],))
            else:
                mus_orch_R[orch_R.index(el[2])] += (el[0],)
    return list(zip(orch_R, mus_orch_R))
```

4) Файл testing.py реализует тестирование программы:

```
import unittest
from classes import music, orchestra, orchMusic
from connections import one_to_many, many_to_many
from processing import task1, task2, task3

class RK_test(unittest.TestCase):
    def setUp(self):
        orchestras = [
            orchestra(1, 'Российский национальный оркестр'),
            orchestra(2, 'Ансамбль песни и пляски'),
            orchestra(3, 'Большой симфонический оркестр'),
            orchestra(4, 'Русская филармония'),
            orchestra(5, 'Виртуозы Москвы'),
            orchestra(6, 'Местные ребята')
        ]
        musics = [
            music(1, 'Глинка', 60, 1),
            music(2, 'Чайковский', 38, 2),
            music(3, 'Шопен', 12, 4),
            music(4, 'Скрябин', 20, 6),
            music(5, 'Бобер', 28, 3),
            music(6, 'Рахманинов', 17, 5),
            music(7, 'Мусоргский', 33, 6),
            music(8, 'Прокофьев', 18, 4),
            music(9, 'Серов', 10, 2),
```

```
            music(10, 'Бетховен', 57, 1),
            music(11, 'Моцерт', 37, 5),
            music(12, 'Бах', 22, 3),
            music(13, 'Варламов', 7, 3)
        ]
        orchMusics = [
            orchMusic(1, 1),
            orchMusic(2, 1),
            orchMusic(3, 2),
            orchMusic(4, 2),
            orchMusic(5, 1),
            orchMusic(6, 3),
            orchMusic(7, 4),
            orchMusic(8, 4),
            orchMusic(9, 5),
            orchMusic(10, 5),
            orchMusic(11, 5),
            orchMusic(12, 6),
            orchMusic(13, 6),
            orchMusic(7, 1),
            orchMusic(8, 2),
            orchMusic(2, 3),
            orchMusic(3, 4),
            orchMusic(4, 5),
            orchMusic(5, 6)
        ]
        self.one_to_many = one_to_many(musics, orchestras)
        self.many_to_many = many_to_many(musics, orchestras,
orchMusics)

    def test_task1(self):
        expected_result = [('Рахманинов', 17, 'Виртуозы Москвы'),
                           ('Серов', 10, 'Ансамбль песни и пляски'),
('Варламов', 7, 'Большой симфонический оркестр')]
        result = task1(self.one_to_many)
        self.assertEqual(result, expected_result)

    def test_task2(self):
        expected_result = [('Российский национальный оркестр', (58.5,
2)), ('Виртуозы Москвы', (27.0, 2)),
                           ('Местные ребята', (26.5, 2)), ('Ансамбль
песни и пляски', (24.0, 2)),
                           ('Русская филармония', (15.0, 2)), ('Большой
симфонический оркестр', (10.666666666666666, 3))]
        result = task2(self.one_to_many)
        self.assertEqual(result, expected_result)

    def test_task3(self):
        expected_result = [('Российский национальный оркестр',
('Глинка', 'Чайковский', 'Бобер', 'Мусоргский')),
                           ('Русская филармония', ('Мусоргский',
'Прокофьев', 'Шопен'))]
        result = task3(self.many_to_many)
        self.assertEqual(result, expected_result)


def main():
    unittest.main()


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

Результат работы программы:

```
testing.py::RK_test::test_task1 PASSED                                   [  33%]
testing.py::RK_test::test_task2 PASSED                                   [  66%]
testing.py::RK_test::test_task3 PASSED                                   [ 100%]

============================== 3 passed in 0.01s ===============================

Process finished with exit code 0
```