ИУ5-32Б

Егоров Виктор

Рубежный контроль 2

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).

Текст программы

main.py

```
from operator import itemgetter


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


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


class MusOrc:
    def __init__(self, mus_id, orc_id):
        self.mus_id = mus_id
        self.orc_id = orc_id


def first_task(one_to_many):
    """Сортировка по первому элементу (id музыканта)."""
    return sorted(one_to_many, key=lambda x: x[0])


def second_task(one_to_many):
    """Подсчет количества оркестров каждого типа."""
    temp_dict = {}
    for _, orchestra_type, _ in one_to_many:
        if orchestra_type in temp_dict:
            temp_dict[orchestra_type] += 1
        else:
            temp_dict[orchestra_type] = 1
    # Сортировка по убыванию количества
    return sorted(temp_dict.items(), key=itemgetter(1), reverse=True)


def third_task(many_to_many, end_ch):
    """Фильтрация оркестров, имена которых заканчиваются на заданную
букву."""
```

```
    return [(orc_name, orc_type) for orc_name, orc_type, _ in many_to_many if
orc_name.endswith(end_ch)]


def main():
    # Данные для работы программы
    musicians = [
        Musician(1, "Бетховен"),
        Musician(2, "Чайковский"),
        Musician(3, "Прокофьев"),
    ]

    orcs = [
        Orchestra(1, "Симфонический оркестр города Гусь Хрустальный",
"Симфонический оркестр", 1),
        Orchestra(2, "Военный оркестр Александровского полка", "Военный
оркестр", 2),
        Orchestra(3, "Bad Apple", "Джазовый оркестр", 3),
        Orchestra(4, "Масленичное чучело", "Оркестр народных инструментов",
3),
        Orchestra(5, "Дудочка", "Духовой оркестр", 1),
        Orchestra(5, "Оркестр имени Петра Ильича Чайковского", "Симфонический
оркестр", 2)
    ]

    mus_orc = [
        MusOrc(1, 1),
        MusOrc(2, 2),
        MusOrc(3, 3),
        MusOrc(1, 4),
        MusOrc(2, 5),
    ]

    # Формирование one_to_many
    one_to_many = [(orc.name, orc.orchestra_type, m.name)
                   for m in musicians
                   for orc in orcs
                   if orc.mus_id == m.id]

    # Формирование many_to_many
    many_to_many_temp = [(m.name, mo.mus_id, mo.orc_id)
                         for m in musicians
                         for mo in mus_orc
                         if mo.mus_id == m.id]

    many_to_many = [(orc.name, orc.orchestra_type, mus_name)
                    for mus_name, mus_id, orc_id in many_to_many_temp
                    for orc in orcs if orc.id == orc_id]

    # Выполнение заданий
    print('Задание Б1')
    print(first_task(one_to_many))

    print("\nЗадание Б2")
    print(second_task(one_to_many))

    print("\nЗадание Б3")
    print(third_task(many_to_many, 'a'))


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

test_main.py

```
import unittest
from main import first_task, second_task, third_task

class TestTasks(unittest.TestCase):

    def test_first_task(self):
        """
        Тест для функции first_task.
        Проверяет, что список отсортирован по первому элементу.
        """
        # Тестовые данные
        one_to_many = [
            ("Чайковский", "Военный оркестр", "Чайковский"),
            ("Бетховен", "Симфонический оркестр", "Бетховен"),
            ("Прокофьев", "Джазовый оркестр", "Прокофьев"),
        ]
        # Ожидаемый результат (сортировка по первому элементу)
        expected = [
            ("Бетховен", "Симфонический оркестр", "Бетховен"),
            ("Прокофьев", "Джазовый оркестр", "Прокофьев"),
            ("Чайковский", "Военный оркестр", "Чайковский"),
        ]
        # Проверка
        self.assertEqual(first_task(one_to_many), expected)

    def test_second_task(self):
        """
        Тест для функции second_task.
        Проверяет, что подсчитывается количество оркестров каждого типа и
сортируется по убыванию.
        """
        # Тестовые данные
        one_to_many = [
            ("Симфонический оркестр", "Симфонический оркестр", "Бетховен"),
            ("Военный оркестр", "Военный оркестр", "Чайковский"),
            ("Джазовый оркестр", "Джазовый оркестр", "Прокофьев"),
            ("Оркестр народных инструментов", "Оркестр народных
инструментов", "Прокофьев"),
            ("Духовой оркестр", "Духовой оркестр", "Бетховен"),
            ("Симфонический оркестр", "Симфонический оркестр", "Чайковский"),
        ]
        # Ожидаемый результат (подсчет по типам оркестров)
        expected = [
            ("Симфонический оркестр", 2),
            ("Военный оркестр", 1),
            ("Джазовый оркестр", 1),
            ("Оркестр народных инструментов", 1),
            ("Духовой оркестр", 1),
        ]
        # Проверка
        self.assertEqual(second_task(one_to_many), expected)

    def test_third_task(self):
        """
        Тест для функции third_task.
        Проверяет, что фильтруются оркестры, имена которых заканчиваются на
заданную букву.
        """
        # Тестовые данные
        many_to_many = [
            ("Симфонический оркестр", "Симфонический оркестр", "Бетховен"),
            ("Военный оркестр", "Военный оркестр", "Чайковский"),
```

```
            ("Джазовый оркестр", "Джазовый оркестр", "Прокофьев"),
            ("Оркестр народных инструментов", "Оркестр народных
инструментов", "Прокофьев"),
            ("Духовой оркестр", "Духовой оркестр", "Бетховен"),
            ("Симфонический оркестр", "Симфонический оркестр", "Чайковский"),
        ]
        # Ожидаемый результат (фильтрация по последней букве 'a')
        expected = []
        # Проверка
        self.assertEqual(third_task(many_to_many, 'a'), expected)

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

Результат

```
Ran 3 tests in 0.002s

OK
```