Московский государственный технический университет им. Н.Э. Баумана

Факультет ИУ
Кафедра ИУ5

Курс «Основы информатики»
Отчет по Рубежному контролю №2

Выполнил студент группы ИУ5-33Б:
Хасанова К.М.
Подпись и дата:

Проверил преподаватель каф.:
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2024 г

Постановка задачи

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).

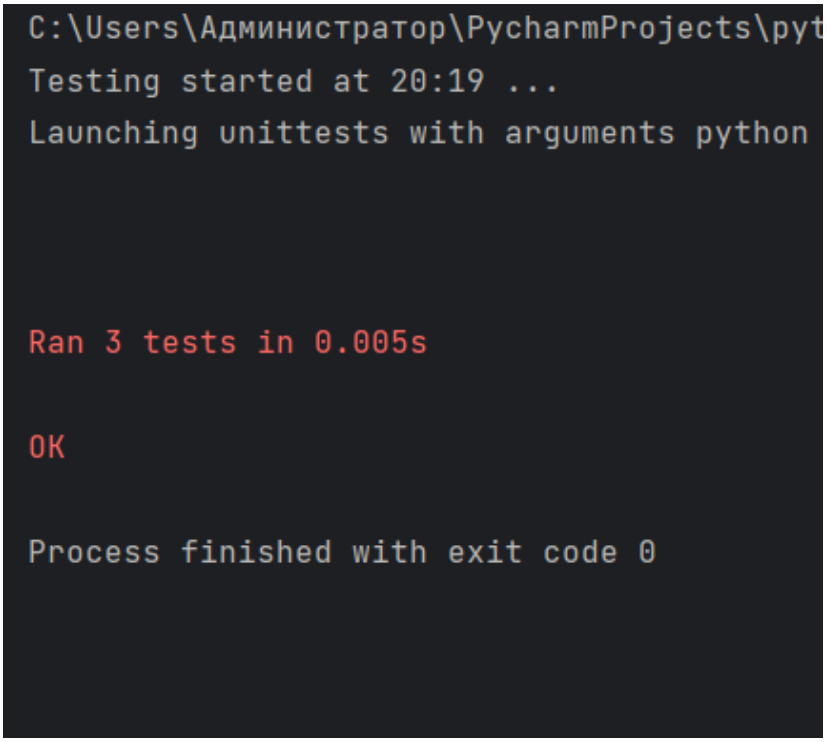
Текст программы

```
import unittest

class MusicalPiece:
    """Музыкальное произведение"""

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


class Orchestra:
    """Оркестр"""

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


class Performance:
    """
    'Выступления оркестров' для реализации
    связи многие-ко-многим
    """

    def __init__(self, orchestra_id, piece_id, date):
        self.orchestra_id = orchestra_id
        self.piece_id = piece_id
        self.date = date


pieces = [
    MusicalPiece(1, 'Симфония № 5', 'Классическая', 'Бетховен', 45),
    MusicalPiece(2, 'Болеро', 'Классическая', 'Равель', 15),
    MusicalPiece(3, 'Концерт для скрипки с оркестром', 'Классическая', 'Чайковский',
30),
    MusicalPiece(4, 'Времена года', 'Классическая', 'Вивальди', 25),
    MusicalPiece(5, 'Лунная соната', 'Романтическая', 'Бетховен', 20),
    MusicalPiece(6, 'Ноктюрн № 2', 'Романтическая', 'Шопен', 10),
    MusicalPiece(7, 'Вальс цветов', 'Романтическая', 'Чайковский', 15),
    MusicalPiece(8, 'Полька', 'Танцевальная', 'Штраус', 5),
    MusicalPiece(9, 'Танго', 'Танцевальная', 'Пьяццолла', 4),
    MusicalPiece(10, 'Маленькая ночная серенада', 'Классическая', 'Моцарт', 12),
]

# Оркестры
orchestras = [
    Orchestra(1, 'Берлинский филармонический оркестр', 'Германия'),
```

```
    Orchestra(2, 'Лондонский симфонический оркестр', 'Великобритания'),
    Orchestra(3, 'Филармония', 'Россия'),
    Orchestra(4, 'Виенский филармонический оркестр', 'Австрия'),
    Orchestra(5, 'Симфонический оркестр Мариинского театра', 'Россия'),
    Orchestra(6, 'Королевский оркестр Концертгебау', 'Нидерланды'),
]

# Связь оркестров и музыкальных произведений
performances = [
    Performance(1, 1, '2023-01-01'),
    Performance(1, 2, '2023-02-01'),
    Performance(2, 3, '2023-03-01'),
    Performance(3, 4, '2023-04-01'),
    Performance(4, 5, '2023-05-01'),
    Performance(5, 6, '2023-06-01'),
    Performance(6, 7, '2023-07-01'),
    Performance(1, 8, '2023-08-01'),
    Performance(2, 9, '2023-09-01'),
    Performance(3, 10, '2023-10-01'),
    Performance(4, 1, '2023-11-01'),
    Performance(5, 2, '2023-12-01'),
    Performance(6, 3, '2024-01-01'),
    Performance(1, 4, '2024-02-01'),
    Performance(2, 5, '2024-03-01'),
    Performance(3, 6, '2024-04-01'),
    Performance(4, 7, '2024-05-01'),
    Performance(5, 8, '2024-06-01'),
    Performance(6, 9, '2024-07-01'),
    Performance(1, 10, '2024-08-01'),
]


def get_pieces_longer_than_20_minutes(one_to_many):
    """Возвращает список произведений длительностью больше 20 минут и оркестров."""
    return [item for item in one_to_many if item[2] > 20]


def get_minimum_duration_orchestras(one_to_many):
    """Возвращает оркестры с минимальной длительностью произведений."""
    min_durations = {}
    for name, author, duration, orchestra_name in one_to_many:
        if orchestra_name not in min_durations or duration <
min_durations[orchestra_name]:
            min_durations[orchestra_name] = duration
    return sorted(min_durations.items(), key=lambda x: x[1])


def get_sorted_performances(many_to_many):
    """Возвращает отсортированные музыкальные произведения и оркестры."""
    return sorted(many_to_many, key=lambda x: (x[0], x[1]))


def main():
    # один ко многим
    one_to_many = [(p.name, p.author, p.duration, o.name)
                   for o in orchestras
                   for p in pieces
                   if p.id == o.id]

    # многие ко многим
    many_to_many_temp = [(o.name, v.orchestra_id, v.piece_id)
                         for o in orchestras
                         for v in performances
                         if o.id == v.orchestra_id]

    many_to_many = [(p.name, p.author, p.duration, o_name)
                    for o_name, o_id, p_id in many_to_many_temp
                    for p in pieces if p.id == p_id]
```

```
    # Задание B1
    print('Задание B1')
    result = get_pieces_longer_than_20_minutes(one_to_many)
    for i in result:
        print(i[0], i[3])

    # Задание B2
    print('Задание B2')
    min_durations = get_minimum_duration_orchestras(one_to_many)
    for orchestra_name, duration in min_durations:
        print(f"Оркестр: {orchestra_name}, Длительность самого короткого произведения:
{duration}")

    # Задание B3
    print('Задание B3')
    sorted_productions = get_sorted_performances(many_to_many)
    for production in sorted_productions:
        print(production[0], production[3])


# Модульные тесты

class TestMusicalFunctions(unittest.TestCase):

    def setUp(self):
        self.one_to_many = [
            ('Симфония № 5', 'Бетховен', 45, 'Берлинский филармонический оркестр'),
            ('Болеро', 'Равель', 15, 'Лондонский симфонический оркестр'),
            ('Концерт для скрипки с оркестром', 'Чайковский', 30, 'Филармония'),
            ('Времена года', 'Вивальди', 25, 'Виенский филармонический оркестр'),
            ('Лунная соната', 'Бетховен', 20, 'Симфонический оркестр Мариинского
театра'),
            ('Ноктюрн № 2', 'Шопен', 10, 'Королевский оркестр Концертгебау'),
        ]

        self.many_to_many = [
            ('Симфония № 5', 'Берлинский филармонический оркестр'),
            ('Болеро', 'Лондонский симфонический оркестр'),
            ('Концерт для скрипки с оркестром', 'Филармония'),
            ('Времена года', 'Виенский филармонический оркестр'),
            ('Лунная соната', 'Симфонический оркестр Мариинского театра'),
            ('Ноктюрн № 2', 'Королевский оркестр Концертгебау'),
        ]

    def test_get_pieces_longer_than_20_minutes(self):
        result = get_pieces_longer_than_20_minutes(self.one_to_many)
        self.assertEqual(len(result), 3)  # Исправлено на 3
        self.assertIn(('Симфония № 5', 'Берлинский филармонический оркестр'), [(i[0],
i[3]) for i in result])

    def test_get_minimum_duration_orchestras(self):
        result = get_minimum_duration_orchestras(self.one_to_many)
        self.assertEqual(result[0], ('Королевский оркестр Концертгебау', 10))

    def test_get_sorted_performances(self):
        result = get_sorted_performances(self.many_to_many)
        self.assertEqual(result[0], ('Болеро', 'Лондонский симфонический оркестр'))


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

Анализ результатов


C:\Users\Администратор\PycharmProjects\pyt
Testing started at 20:19 ...
Launching unittests with arguments python
Ran 3 tests in 0.005s
OK
Process finished with exit code 0