

Московский государственный технический университет им. Н.Э. Баумана

**Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»**

**Курс «Базовые компоненты интернет-технологий»
Отчет по рубежному контролю №2**

Выполнил:
студент группы ИУ5-31Б
Егошин Дмитрий
Павлович

Подпись: _____

Дата: _____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич

Подпись: _____

Дата: _____

Москва, 2021 г.

Лабораторная работа №3

Описание задания

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).

Код программы

Файл main.py(РК №1)

```
# используется для сортировки
from operator import itemgetter

class Musician:
    """Музыкант"""

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

class Orchestra:
    """Оркестр"""

    def __init__(self, id, name, num, mus_id):
        self.id = id
        self.name = name
        self.num = num # количество музыкантов в оркестре
        self.mus_id = mus_id

class MusOrch:
    """
    'Музыканты оркестра' для реализации
    СВЯЗИ МНОГИЕ-КО-МНОГИМ
    """

    def __init__(self, mus_id, orch_id):
        self.orch_id = orch_id
        self.mus_id = mus_id

# Оркестры
Orchs = [
    Orchestra(1, 'Великорусский оркестр ', 100, 1),
    Orchestra(2, 'Государственный духовой оркестр России', 150, 2),
    Orchestra(3, 'Музыка души', 200, 3),

    Orchestra(11, 'Молчат дома', 250, 4),
    Orchestra(22, 'Академический симфонический оркестр', 300, 5),
    Orchestra(33, 'Симфонический оркестр', 350, 5),
```

```

]

# Музыканты
Mus = [
    Musician(1, 'Жмышенко'),
    Musician(2, 'Александров'),
    Musician(3, 'Симонов'),
    Musician(4, 'Ленин'),
    Musician(5, 'Ищенко'),
]

Orchs_Mus = [
    MusOrch(1, 1),
    MusOrch(2, 2),
    MusOrch(3, 3),
    MusOrch(4, 11),
    MusOrch(5, 22),

    MusOrch(1, 33),
    MusOrch(2, 1),
    MusOrch(3, 2),
    MusOrch(4, 3),
    MusOrch(5, 11),
]

def sorting_by_name(table):
    return sorted(table, key=itemgetter(2))

def sorting_by_sum_musicians(table, mus):
    result_unsorted = []
    # Перебираем всех музыкантов
    for m in mus:
        # Список оркестров музыканта
        m_orchs = list(filter(lambda i: i[0] == m.fio, table))
        # Если список не пустой
        if len(m_orchs) > 0:
            # Количество музыкантов в оркестрах
            o_num = [num[1] for num in m_orchs]
            # Суммарное количество музыкантов
            o_num_sum = sum(o_num)
            result_unsorted.append((m.fio, o_num_sum))
    # Сортировка по суммарному количеству музыкантов в оркестрах
    return sorted(result_unsorted, key=itemgetter(1), reverse=True)

def output_musicians_of_orchs_with_ORKESTR(table, orchs):
    result = {}
    # Перебираем все оркестры
    for o in orchs:
        if 'оркестр' in o.name:
            # Список музыкантов оркестра
            o_mus = list(filter(lambda i: i[0] == o.name, table))
            # Только ФИО музыкантов
            o_mus_names = [x[2] for x in o_mus]
            # Добавляем результат в словарь
            # ключ - оркестр, значение - список фамилий
            result[o.name] = o_mus_names
    return result

def main():

```

```

"""Основная функция"""

# Соединение данных один-ко-многим
one_to_many = [(m.fio, o.num, o.name)
                for m in Mus
                for o in Orchs
                if o.mus_id == m.id]

# Соединение данных многие-ко-многим
many_to_many_temp = [(m.fio, om.orch_id, om.mus_id)
                      for m in Mus
                      for om in Orchs_Mus
                      if m.id == om.mus_id]

many_to_many = [(o.name, o.num, mus_name)
                 for mus_name, orch_id, mus_id in many_to_many_temp
                 for o in Orchs if o.id == orch_id]

print('Задание A1')
print(sorting_by_name(one_to_many))

print('\nЗадание A2')
print(sorting_by_sum_musicians(one_to_many, Mus))

print('\nЗадание A3')
print(output_musicians_of_orchs_with_ORKESTR(many_to_many, Orchs))

if __name__ == '__main__':
    main()

```

Test.py(РК №2)

```

from main import Orchestra, Musician, MusOrch, sorting_by_name, \
    sorting_by_sum_musicians, \
    output_musicians_of_orchs_with_ORKESTR
import unittest

class Tests(unittest.TestCase):
    def setUp(self):
        # Оркестры
        self.Orchs = [
            Orchestra(1, 'Великорусский оркестр ', 100, 1),
            Orchestra(2, 'Государственный духовой оркестр России', 150, 2),
            Orchestra(3, 'Музыка души', 200, 3),

            Orchestra(11, 'Молчат дома', 250, 4),
            Orchestra(22, 'Академический симфонический оркестр', 300, 5),
            Orchestra(33, 'Симфонический оркестр', 350, 5),
        ]

        # Музыканты
        self.Mus = [
            Musician(1, 'Жмышенко'),
            Musician(2, 'Александров'),
            Musician(3, 'Симонов'),
            Musician(4, 'Ленин'),
            Musician(5, 'Ищенко'),
        ]

```

```

self.Orchs_Mus = [
    MusOrch(1, 1),
    MusOrch(2, 2),
    MusOrch(3, 3),
    MusOrch(4, 11),
    MusOrch(5, 22),

    MusOrch(1, 33),
    MusOrch(2, 1),
    MusOrch(3, 2),
    MusOrch(4, 3),
    MusOrch(5, 11),
]
# Соединение данных один-ко-многим
self.one_to_many = [(m.fio, o.num, o.name)
                     for m in self.Mus
                     for o in self.Orchs
                     if o.mus_id == m.id]

# Соединение данных многие-ко-многим
self.many_to_many_temp = [(m.fio, om.orch_id, om.mus_id)
                           for m in self.Mus
                           for om in self.Orchs_Mus
                           if m.id == om.mus_id]

self.many_to_many = [(o.name, o.num, mus_name)
                      for mus_name, orch_id, mus_id in
self.many_to_many_temp
                      for o in self.Orchs if o.id == orch_id]

def test_sorting_by_name(self):
    result = sorting_by_name(self.one_to_many)
    desired_result = [('Ищенко', 300, 'Академический симфонический
оркестр'),
                      ('Жмышенко', 100, 'Великорусский оркестр '),
                      ('Александров', 150, 'Государственный духовой
оркестр России'), ('Ленин', 250, 'Молчат дома'),
                      ('Симонов', 200, 'Музыка души'), ('Ищенко', 350,
'Симфонический оркестр')]
    self.assertEqual(result, desired_result)

def test_sorting_by_sum(self):
    result = sorting_by_sum_musicians(self.one_to_many, self.Mus)
    desired_result = [('Ищенко', 650), ('Ленин', 250), ('Симонов', 200),
('Александров', 150), ('Жмышенко', 100)]
    self.assertEqual(result, desired_result)

def test_output_PAPKA2(self):
    result = output_musicians_of_orchs_with_ORKESTR(self.many_to_many,
self.Orchs)
    desired_result = {'Великорусский оркестр ': ['Жмышенко',
'Александров'],
                      'Государственный духовой оркестр России':
['Александров', 'Симонов'],
                      'Академический симфонический оркестр': ['Ищенко'],
'Симфонический оркестр': ['Жмышенко']}
    self.assertEqual(result, desired_result)

```

Результат выполнения программы

```
"C:\Users\Дмитрий Павлович\AppData\Local\Programs\Python\Python38-32\python.exe" "C:\Program Files\JetBrains\PyCharm Community
Testing started at 21:34 ...
Launching unittests with arguments python -m unittest tests.Tests in F:\BKIT\RK

Ran 3 tests in 0.003s

OK

Process finished with exit code 0
```