Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Рубежный контроль №2

по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:
студент группы ИУ5-33Б
Зорькин А.В.

Проверил:
Гапанюк Ю.Е.

2021 г.

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).

Текст программы:

music.py:

```
import operator
from operator import itemgetter
from collections import defaultdict


class Musician:
    """Музыкант"""

    def __init__(self, id, surname, salary, orchestra_id):
        self.id = id
        self.surname = surname
        self.salary = salary
        self.orchestra_id = orchestra_id


class Orchestra:
    """Оркестр"""

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


class MusOrc:
    """
    'Музыканты оркестра' для реализации
    связи многие-ко-многим
    """

    def __init__(self, orchestra_id, musician_id):
        self.orchestra_id = orchestra_id
        self.musician_id = musician_id


def orchestra_id():
    i = 0
    while True:
        i += 1
        yield i


def musician_id():
    i = 0
    while True:
        i += 1
        yield i
```

```python
def task_1(one_to_many):
    return [(x[0], x[2]) for x in one_to_many if x[0][0] == 'A']


def task_2(one_to_many):  # реализация кажется более оптимизированной, чем при использовании функций высших порядков
    result = {}
    for x in one_to_many:
        if x[2] in result:
            if x[1] < result[x[2]]:
                result[x[2]] = x[1]
        else:
            result[x[2]] = x[1]
    return sorted(result.items(), key=operator.itemgetter(1))


def task_3(many_to_many):
    result = defaultdict(list)
    for x in many_to_many:
        result[x[0]].append(x[1])
    return sorted(result.items(), key=operator.itemgetter(0))


# Оркестры
orc_id = orchestra_id()
orchestras = [
    Orchestra(next(orc_id), 'Российский национальный оркестр'),
    Orchestra(next(orc_id), 'Лондонский симфонический оркестр'),
    Orchestra(next(orc_id), 'Королевский филармонический оркестр'),
    Orchestra(next(orc_id), 'Российский национальный оркестр(другой)'),
    Orchestra(next(orc_id), 'Лондонский симфонический оркестр(другой)'),
    Orchestra(next(orc_id), 'Королевский филармонический оркестр(другая)'),
]
mus_id = musician_id()
# Музыканты
musicians = [
    Musician(next(mus_id), 'Абрамов', 250000, 1),
    Musician(next(mus_id), 'Ахтамбаев', 315000, 2),
    Musician(next(mus_id), 'Некрасов', 453000, 2),
    Musician(next(mus_id), 'Зорькин', 432000, 3),
    Musician(next(mus_id), 'Прошкина', 343000, 3),
]

musorc = {
    MusOrc(1, 1),
    MusOrc(2, 2),
    MusOrc(3, 3),
    MusOrc(3, 4),
    MusOrc(3, 5),
    MusOrc(4, 1),
    MusOrc(5, 2),
    MusOrc(6, 3),
    MusOrc(6, 4),
    MusOrc(6, 5),
}


def main():
    """Основная функция"""
    # Соединение данных один-ко-многим
    one_to_many = [(m.surname, m.salary, o.name) for o in orchestras for m in musicians if m.orchestra_id == o.id]
```

```
    # Соединение данных многие-ко-многим
    many_to_many_temp = [(o.name, r.orchestra_id, r.musician_id) for o in
orchestras for r in musorc if
                         o.id == r.orchestra_id]

    many_to_many = [(m.surname, o_name) for o_name, orchestra_id, musician_id
in many_to_many_temp for m in musicians if
                    m.id == musician_id]
    print("Задание №1:", '\n', task_1(one_to_many))
    print("Задание №2:", '\n', task_2(one_to_many))
    print("Задание №3:", '\n', task_3(many_to_many))


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

test.py:

```
import unittest
import sys, os

sys.path.append(os.getcwd())
from music import *


class Test_task_1(unittest.TestCase):
    def test_First_letter(self):
        one_to_many = [(m.surname, m.salary, o.name) for o in orchestras for
m in musicians if m.orchestra_id == o.id]
        self.assertEqual(task_1(one_to_many), [('Абрамов', 'Российский
национальный оркестр'),
                                               ('Ахтамбаев', 'Лондонский
симфонический оркестр')])


class Test_task_2(unittest.TestCase):
    def test_min_many_everyone_department(self):
        one_to_many = [(m.surname, m.salary, o.name) for o in orchestras for
m in musicians if m.orchestra_id == o.id]
        self.assertEqual(task_2(one_to_many),
                         [('Российский национальный оркестр', 250000),
('Лондонский симфонический оркестр', 315000),
                          ('Королевский филармонический оркестр', 343000)])


class Test_task_3(unittest.TestCase):
    def test_worker_many_departments(self):
        many_to_many_temp = [(o.name, r.orchestra_id, r.musician_id) for o in
orchestras for r in musorc if
                             o.id == r.orchestra_id]
        many_to_many = [(m.surname, o_name) for o_name, orchestra_id,
musician_id in many_to_many_temp for m in
                        musicians if
                        m.id == musician_id]
        self.assertEqual(task_3(many_to_many),
                         [('Абрамов', ['Российский национальный оркестр',
'Российский национальный оркестр(другой)']), (
                             'Ахтамбаев',
                             ['Лондонский симфонический оркестр', 'Лондонский
симфонический оркестр(другой)']),
                          ('Зорькин',
                           ['Королевский филармонический оркестр',
'Королевский филармонический оркестр(другая)']), (
```

```
                                'Некрасов',
                                ['Королевский филармонический оркестр',
'Королевский филармонический оркестр(другая)']), (
                                'Прошкина',
                                ['Королевский филармонический оркестр',
'Королевский филармонический оркестр(другая)']])])
```

Результат выполнения программы:

```
Launching unittests with arguments python -m unittest (



Ran 3 tests in 0.011s

OK

Process finished with exit code 0
```