**Московский государственный технический университет им. Н.Э. Баумана.**

## Факультет «Информатика и управление»

Кафедра ИУ5 «Системы обработки информации и управления»

Рубежный контроль №2

по дисциплине **«**Базовые компоненты интернет-технологий**»**

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| студент группы ИУ5-33 | преподаватель каф.  ИУ5 |
| Стебунов А.И. | Гапанюк Ю.Е. |

2021

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: main.py

## используется для сортировки  
from operator import itemgetter  
  
class Composition:  
 *"""Музыкальное произведение"""* def \_\_init\_\_(self, id, naz, count):  
 self.id = id  
 self.naz = naz  
 self.count = count  
  
  
class Orchestra:  
 *"""Оркестр"""* def \_\_init\_\_(self, id,name):  
 self.id = id  
 self.name = name # название  
  
class OrchestraComposition:  
 *"""  
 'Исполнители произведения' для реализации  
 связи многие-ко-многим  
 """* def \_\_init\_\_(self, orc\_id, comp\_id):  
 self.orc\_id = orc\_id  
 self.comp\_id = comp\_id  
  
  
# Оркестр  
orc = [  
 Orchestra(1, 'народный'),  
 Orchestra(2, 'камерный'),  
 Orchestra(3, 'военный'),  
 Orchestra(4, 'эстрадный'),  
 Orchestra(5, 'симфонический'),  
]  
  
# Музыкальные композиции  
comp = [  
 Composition(1, 'Лунная соната', 3),  
 Composition(2, 'Танец рыцарей', 4),  
 Composition(3, 'Марш Мендельсона', 2),  
 Composition(4, 'Пятая симфония', 5),  
 Composition(5, 'Щелкунчик', 4),  
 Composition(6, 'Новый мерин', 3),  
 Composition(7, 'Ария', 2),  
 Composition(8, 'Кан-кан', 1)  
]  
  
оrchestro\_сomposition = [  
 OrchestraComposition(1, 1),  
 OrchestraComposition(2, 2),  
 OrchestraComposition(3, 4),  
 OrchestraComposition(4, 1),  
 OrchestraComposition(5, 3),  
 OrchestraComposition(6, 1),  
 OrchestraComposition(7, 2),  
 OrchestraComposition(8, 3),  
 OrchestraComposition(7, 5),  
 OrchestraComposition(6, 2),  
]  
def Task1(one\_to\_many):  
 print('Задание В1')  
 res\_11 = []  
 for naz, count, orc\_name in one\_to\_many:  
 if 'А' in naz[0]:  
 res\_11.append((naz, orc\_name))  
 return res\_11  
def Task2(one\_to\_many):  
 print('Задание В2')  
 buff = []  
 for o in orc:  
 # список видов транспорта  
 o\_names = list(filter(lambda i: i[2] == o.name, one\_to\_many))  
 if len(o\_names) > 0:  
 o\_count = [count for \_, count, \_ in o\_names]  
 min\_count = min(o\_count)  
 buff.append((o.name, min\_count))  
 res\_12 = sorted(buff, key=itemgetter(1))  
 return res\_12  
def Task3(many\_to\_many):  
 print('Задание В3')  
 buff = []  
 for naz, count, orc\_name in many\_to\_many:  
 buff.append((naz, orc\_name))  
 res\_13 = list(sorted(buff, key=itemgetter(0)))  
 return res\_13  
  
def main():  
 *"""Основная функция"""* # Соединение данных один-ко-многим  
 one\_to\_many = [(c.naz, c.count, o.name)  
 for o in orc  
 for c in comp  
 if c.count == o.id]  
  
 # Соединение данных многие-ко-многим  
 many\_to\_many\_temp = [(o.name, oc.orc\_id, oc.comp\_id)  
 for o in orc  
 for oc in оrchestro\_сomposition  
 if o.id == oc.orc\_id]  
  
 many\_to\_many = [(c.naz, c.count, orc\_name)  
 for orc\_name, orc\_id, comp\_id in many\_to\_many\_temp  
 for c in comp if c.id == comp\_id]  
  
  
 print('Test')#вывод списков со связями 1-м, м-м  
 res\_0 = (one\_to\_many)  
 print(res\_0)  
 res\_01 = (many\_to\_many)  
 print(res\_01)  
  
 Task1(one\_to\_many)  
  
 Task2(one\_to\_many)  
  
 Task3(many\_to\_many)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

## test.py

import unittest  
import sys, os  
  
sys.path.append(os.getcwd())  
from main import \*  
  
  
class Test\_task\_1(unittest.TestCase):  
 def test\_task\_1(self):  
 one\_to\_many = [(c.naz, c.count, o.name)  
 for o in orc  
 for c in comp  
 if c.count == o.id]  
 self.assertEqual(Task1(one\_to\_many), [('Ария', 'камерный')])  
  
class Test\_task\_2(unittest.TestCase):  
 def test\_task\_2(self):  
 one\_to\_many = [(c.naz, c.count, o.name)  
 for o in orc  
 for c in comp  
 if c.count == o.id]  
 self.assertEqual(Task2(one\_to\_many),[('народный', 1), ('камерный', 2), ('военный', 3), ('эстрадный', 4), ('симфонический', 5)])  
  
  
class Test\_task\_3(unittest.TestCase):  
 def test\_task\_3(self):  
 many\_to\_many\_temp = [(o.name, oc.orc\_id, oc.comp\_id)  
 for o in orc  
 for oc in оrchestro\_сomposition  
 if o.id == oc.orc\_id]  
 many\_to\_many = [(c.naz, c.count, orc\_name)  
 for orc\_name, orc\_id, comp\_id in many\_to\_many\_temp  
 for c in comp if c.id == comp\_id]  
 self.assertEqual(Task3(many\_to\_many),  
 [('Лунная соната', 'народный'), ('Лунная соната', 'эстрадный'), ('Марш Мендельсона', 'симфонический'), ('Пятая симфония', 'военный'), ('Танец рыцарей', 'камерный')])  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()

# Результат выполнения программы:

