**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: | Проверил: |
| студент группы ИУ5-34Б |  |
| Елизавета Рысьева |  |
| Подпись и дата:  13.12.21 | Подпись и дата: |

Москва, 2021г.

# 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).

# Файл main.py:

from operator import itemgetter  
  
  
class Mus:  
  
 def \_\_init\_\_(self, id, name, long, orch\_id):  
 self.id = id  
 self.name = name  
 self.long = long  
 self.orch\_id = orch\_id  
  
  
class Orch:  
  
 def \_\_init\_\_(self, id, instr):  
 self.id = id  
 self.instr = instr  
  
  
class MusOrch:  
  
 def \_\_init\_\_(self, orch\_id, mus\_id):  
 self.mus\_id = mus\_id  
 self.orch\_id = orch\_id  
  
  
orch = [  
 Orch(1, 'Piano'),  
 Orch(2, 'Flute'),  
 Orch(3, 'Violin'),  
  
 Orch(11, 'Piano other'),  
 Orch(22, 'Flute other'),  
 Orch(33, 'Violin other'),  
  
]  
  
mus = [  
 Mus(1, 'Moonlight Sonata', 60, 1),  
 Mus(2, 'Swan Lack', 10, 2),  
 Mus(3, 'Turkish March ', 3, 3),  
  
]  
  
mus\_orch = [  
 MusOrch(1, 1),  
 MusOrch(2, 2),  
 MusOrch(3, 3),  
 MusOrch(1, 2),  
 MusOrch(2, 3),  
  
 MusOrch(11, 1),  
 MusOrch(22, 2),  
 MusOrch(33, 3),  
 MusOrch(11, 3),  
  
]  
  
  
def sorting\_by\_name(table):  
 return sorted(table, key=itemgetter(2))  
  
  
def sorting\_by\_len\_of\_mus(table, orch):  
 result\_unsorted = []  
  
 for o in orch:  
  
 o\_mus = list(filter(lambda i: i[2] == o.instr, table))  
  
 if len(o\_mus) > 0:  
 o\_long = [long for \_, long, \_ in o\_mus]  
  
 o\_long\_sum = sum(o\_long)  
 result\_unsorted.append((o.instr, o\_long\_sum))  
  
 return sorted(result\_unsorted, key=itemgetter(1), reverse=True)  
  
  
def output\_mus\_of\_orchs\_with\_FLUTE(table, orch):  
 result = {}  
  
 for o in orch:  
 if 'Flute' in o.instr:  
 o\_mus = list(filter(lambda i: i[2] == o.instr, table))  
  
 o\_mus\_instr = [x for x, \_, \_ in o\_mus]  
  
 result[o.instr] = o\_mus\_instr  
 return result  
  
def main():  
 one\_to\_many = [(m.name, m.long, o.instr)  
 for o in orch  
 for m in mus  
 if m.orch\_id == o.id]  
  
 many\_to\_many\_temp = [(o.instr, mo.orch\_id, mo.mus\_id)  
 for o in orch  
 for mo in mus\_orch  
 if o.id == mo.orch\_id]  
  
 many\_to\_many = [(m.name, m.long, dep\_name)  
 for dep\_name, dep\_id, emp\_id in many\_to\_many\_temp  
 for m in mus if m.id == emp\_id]  
  
 print('A1')  
 print(sorting\_by\_name(one\_to\_many))  
  
 print('\nA2')  
 print(sorting\_by\_len\_of\_mus(one\_to\_many, orch))  
  
 print('\nA3')  
 print(output\_mus\_of\_orchs\_with\_FLUTE(many\_to\_many, orch))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Файл Tests.py:**

from main import Mus, Orch, MusOrch, sorting\_by\_name, sorting\_by\_len\_of\_mus, output\_mus\_of\_orchs\_with\_FLUTE  
import unittest  
  
  
class Tests(unittest.TestCase):  
 def setUp(self):  
 self.orch = [  
 Orch(1, 'Piano'),  
 Orch(2, 'Flute'),  
 Orch(3, 'Violin'),  
  
 Orch(11, 'Piano other'),  
 Orch(22, 'Flute other'),  
 Orch(33, 'Violin other'),  
  
 ]  
  
 self.mus = [  
 Mus(1, 'Moonlight Sonata', 60, 1),  
 Mus(2, 'Swan Lack', 10, 2),  
 Mus(3, 'Turkish March ', 3, 3),  
  
 ]  
  
 self.mus\_orch = [  
 MusOrch(1, 1),  
 MusOrch(2, 2),  
 MusOrch(3, 3),  
 MusOrch(1, 2),  
 MusOrch(2, 3),  
  
 MusOrch(11, 1),  
 MusOrch(22, 2),  
 MusOrch(33, 3),  
 MusOrch(11, 3),  
  
 ]  
  
 self.one\_to\_many = [(m.name, m.long, o.instr)  
 for o in self.orch  
 for m in self.mus  
 if m.orch\_id == o.id]  
  
 self.many\_to\_many\_temp = [(o.instr, mo.orch\_id, mo.mus\_id)  
 for o in self.orch  
 for mo in self.mus\_orch  
 if o.id == mo.orch\_id]  
  
 self.many\_to\_many = [(m.name, m.long, dep\_name)  
 for dep\_name, dep\_id, emp\_id in self.many\_to\_many\_temp  
 for m in self.mus if m.id == emp\_id]  
  
 def test\_sorting\_by\_name(self):  
 result = sorting\_by\_name(self.one\_to\_many)  
 desired\_result = [('Swan Lack', 10, 'Flute'), ('Moonlight Sonata', 60, 'Piano'), ('Turkish March ', 3, 'Violin')]  
 self.assertEqual(result, desired\_result)  
  
 def test\_sorting\_by\_len(self):  
 result = sorting\_by\_len\_of\_mus(self.one\_to\_many, self.orch)  
 desired\_result = [('Piano', 60), ('Flute', 10), ('Violin', 3)]  
 self.assertEqual(result, desired\_result)  
  
 def test\_output\_FLUTE(self):  
 result = output\_mus\_of\_orchs\_with\_FLUTE(self.many\_to\_many, self.orch)  
 desired\_result = {'Flute': ['Swan Lack', 'Turkish March '], 'Flute other': ['Swan Lack']}  
 self.assertEqual(result, desired\_result)

# Результат выполнения

# 