**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по рубежному контролю №2

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-31Б |  | преподаватель каф. ИУ5 |
| Койбаев Тамерлан |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2022 г.

**Условия рубежного контроля №2 по курсу БКИТ**

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.

**Измененный код РК №1:**

# используется для сортировки  
from operator import itemgetter  
  
  
class Musician:  
 *"""Музыкант"""* def \_\_init\_\_(self, id, name, age, orch\_id):  
 self.id = id  
 self.name = name  
 self.age = age  
 self.orch\_id = orch\_id  
  
  
class Orchestra:  
 *"""Оркестр"""* def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
  
class MusOrch:  
 *"""  
 'Музыкант в оркестре' для реализации  
 связи многие-ко-многим  
 """* def \_\_init\_\_(self, orch\_id, mus\_id):  
 self.orch\_id = orch\_id  
 self.mus\_id = mus\_id  
  
  
# Оркестры  
orch = [  
 Orchestra(1, 'оркестр 1'),  
 Orchestra(2, 'оркестр 2'),  
 Orchestra(3, 'оркестр 3'),  
]  
  
# Музыканты  
mus = [  
 Musician(1, 'Холмогоров', 25, 1),  
 Musician(2, 'Пчелкин', 31, 1),  
 Musician(3, 'Белый', 19, 2),  
 Musician(4, 'Филатов', 44, 2),  
 Musician(5, 'Лапшин', 50, 3),  
 Musician(6, 'Шмидт', 22, 4)  
]  
  
mus\_orch = [  
 MusOrch(1, 1),  
 MusOrch(1, 2),  
 MusOrch(2, 3),  
 MusOrch(2, 4),  
 MusOrch(3, 5),  
 MusOrch(3, 6)  
]  
  
def one\_to\_many(orch,mus):  
 return [(c.name, c.age, r.name)  
 for r in orch  
 for c in mus  
 if c.orch\_id == r.id]  
  
def many\_to\_many(orch,mus):  
 many\_to\_many\_temp = [(r.name, cr.orch\_id, cr.mus\_id)  
 for r in orch  
 for cr in mus\_orch  
 if r.id == cr.orch\_id]  
 return [(c.name, c.age, orch\_name)  
 for orch\_name, orch\_id, mus\_id in many\_to\_many\_temp  
 for c in mus if c.id == mus\_id]  
  
def A1(orch,mus) ->list:  
 res\_11 = sorted(one\_to\_many(orch,mus), key=itemgetter(2))  
 return res\_11  
  
def A2(orch,mus) ->list:  
 res\_12\_unsorted = []  
 # Перебираем все оркестры  
 for r in orch:  
 # Список музыкантов оркестра  
 r\_orch = list(filter(lambda i: i[2] == r.name, one\_to\_many(orch,mus)))  
 # Если музыкантов > 0  
 if len(r\_orch) > 0:  
 # Возраст музыкантов в оркестре  
 r\_age = [age for \_, age, \_ in r\_orch]  
 # Суммарный возраст  
 r\_age\_sum = sum(r\_age)  
 res\_12\_unsorted.append((r.name, r\_age\_sum))  
  
 # Сортировка по суммарному возрасту  
 res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)  
 return res\_12  
def A3(orch,mus,str) ->list:  
 res\_13 = {}  
 # Перебираем все оркестры  
 for r in orch:  
 if str not in r.name:  
 # Список музыкантов оркестра  
 r\_orch = list(filter(lambda i: i[2] == r.name, many\_to\_many(orch,mus)))  
 # Только имена  
 r\_orch\_names = [x for x, \_, \_ in r\_orch]  
 # Добавляем результат в словарь  
 # ключ - оркестр, значение - имена  
 res\_13[r.name] = r\_orch\_names  
  
 return(res\_13)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print(A1(orch,mus))  
 print(A2(orch, mus))  
 print(A3(orch, mus))

**Тестирование:**

import unittest  
from rk1re import Musician, Orchestra, MusOrch, A1, A2, A3  
  
  
class test(unittest.TestCase):  
  
 def setUp(self):  
 self.orch = [  
 Orchestra(1, 'оркестр 1'),  
 Orchestra(2, 'оркестр 2'),  
 Orchestra(3, 'оркестр 3'),  
]  
 self.mus = [  
 Musician(1, 'Холмогоров', 25, 1),  
 Musician(2, 'Пчелкин', 31, 1),  
 Musician(3, 'Белый', 19, 2),  
 Musician(4, 'Филатов', 44, 2),  
 Musician(5, 'Лапшин', 50, 3),  
 Musician(6, 'Шмидт', 22, 4)  
]  
 self.mus\_orch = [  
 MusOrch(1, 1),  
 MusOrch(1, 2),  
 MusOrch(2, 3),  
 MusOrch(2, 4),  
 MusOrch(3, 5),  
 MusOrch(3, 6)  
]  
  
 def test\_A1(self):  
 expected\_result = [('Холмогоров', 25, 'оркестр 1'),  
 ('Пчелкин', 31, 'оркестр 1'),  
 ('Белый', 19, 'оркестр 2'),  
 ('Филатов', 44, 'оркестр 2'),  
 ('Лапшин', 50, 'оркестр 3')]  
  
 result = A1(self.orch, self.mus)  
 self.assertEqual(result, expected\_result)  
  
 def test\_A2(self):  
 expected\_result = [('оркестр 2', 63),  
 ('оркестр 1', 56),  
 ('оркестр 3', 50)]  
 result = A2(self.orch, self.mus)  
 self.assertEqual(result, expected\_result)  
  
 def test\_A3(self):  
 expected\_result = {'оркестр 2': ['Белый', 'Филатов'], 'оркестр 3': ['Лапшин', 'Шмидт']}  
 result = A3(self.orch, self.mus,'1')  
 self.assertEqual(result, expected\_result)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

Результаты тестирования:

