**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 «Системы обработки информации и управления»**

**Рубежный контроль №2.**

по предмету

«Базовые компоненты интернет-технологий»

Выполнил:

студент группы ИУ5-31Б

Изибаев Андрей

Проверил:

Преподаватель кафедры ИУ-5

Гапанюк Юрий

2022 г.

**Условия РК №1**

**Вариант А. Предметная область 4.**

1. «Компьютер» и «Дисплейный класс» связаны соотношением один-ко-многим. Выведите список всех связанных компьютеров и дисплейных классов, отсортированный по классам, сортировка по компьютерам произвольная
2. «Компьютер» и «Дисплейный класс» связаны соотношением один-ко-многим. Выведите список дисплейных классов с суммарной стоимостью в каждом классе, отсортированный по суммарной стоимости.
3. «Компьютер» и «Дисплейный класс» связаны соотношением многие-ко-многим. Выведите список всех дисплейных классов, у которых в названии отсутствует слово «просто», и список находящихся в них файлов.

**Условия РК №2**

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**

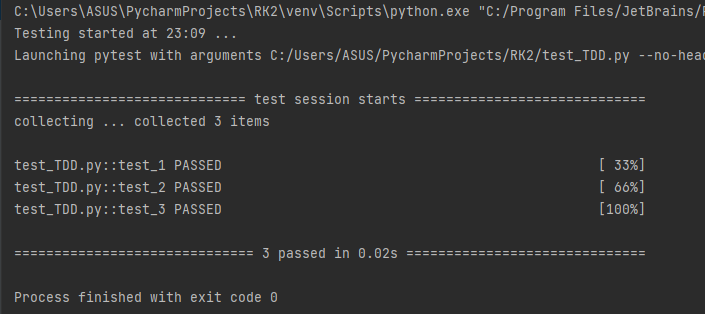
**main.py**

# используется для сортировки  
from operator import itemgetter  
  
  
class Comp:  
 *"""Компьютер"""* def \_\_init\_\_(self, id, name, price, disp\_id):  
 self.id = id  
 self.name = name  
 self.price = price  
 self.disp\_id = disp\_id  
  
  
class Disp:  
 *"""Дисплейный класс"""* def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
  
class CompDisp:  
 *"""  
 'Компьютер в дисплейном классе' для реализации  
 связи многие-ко-многим  
 """* def \_\_init\_\_(self, disp\_id, comp\_id):  
 self.disp\_id = disp\_id  
 self.comp\_id = comp\_id  
  
  
# Дисплейные классы  
rooms = [  
 Disp(1, 'киберспортивный класс'),  
 Disp(2, 'яблочный класс'),  
 Disp(3, 'просто класс'),  
]  
  
# Компьютеры  
comps = [  
 Comp(1, 'ASUS ROG', 120000, 1),  
 Comp(2, 'ASUS TUF', 100000, 1),  
 Comp(3, 'MacBook Air m1', 80000, 2),  
 Comp(4, 'MacBook Air m2', 110000, 2),  
 Comp(5, 'Asus Vivobook', 70000, 3),  
 Comp(6, 'Asus ZenBook', 85000, 4)  
]  
  
comps\_rooms = [  
 CompDisp(1, 1),  
 CompDisp(1, 2),  
 CompDisp(2, 3),  
 CompDisp(2, 4),  
 CompDisp(3, 5),  
 CompDisp(3, 6)  
]  
  
  
def connect():  
 # Соединение данных один-ко-многим  
 one\_to\_many = [(c.name, c.price, r.name)  
 for r in rooms  
 for c in comps  
 if c.disp\_id == r.id]  
  
 # Соединение данных многие-ко-многим  
 many\_to\_many\_temp = [(r.name, cr.disp\_id, cr.comp\_id)  
 for r in rooms  
 for cr in comps\_rooms  
 if r.id == cr.disp\_id]  
  
 many\_to\_many = [(c.name, c.price, room\_name)  
 for room\_name, room\_id, comp\_id in many\_to\_many\_temp  
 for c in comps if c.id == comp\_id]  
 return one\_to\_many, many\_to\_many  
  
def Task\_1(one\_to\_many):  
 return sorted(one\_to\_many, key=itemgetter(2))  
  
def Task\_2(one\_to\_many):  
 res\_12\_unsorted = []  
 # Перебираем все кабинеты  
 for r in rooms:  
 # Список компьютеров кабинета  
 r\_rooms = list(filter(lambda i: i[2] == r.name, one\_to\_many))  
 # Если компьютеров > 0  
 if len(r\_rooms) > 0:  
 # Стоимости компьютеров в кабинете  
 r\_price = [price for \_, price, \_ in r\_rooms]  
 # Суммарная стоимость компьютеров в кабинете  
 r\_price\_sum = sum(r\_price)  
 res\_12\_unsorted.append((r.name, r\_price\_sum))  
  
 return sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)  
  
def Task\_3(one\_to\_many):  
 res\_13 = {}  
 # Перебираем все кабинеты  
 for r in rooms:  
 if 'просто' not in r.name:  
 # Список ноутбуков кабинета  
 r\_rooms = list(filter(lambda i: i[2] == r.name, many\_to\_many))  
 # Только названия ноутбуков  
 r\_room\_names = [x for x, \_, \_ in r\_rooms]  
 # Добавляем результат в словарь  
 # ключ - кабинет, значение - список компьютеров  
 res\_13[r.name] = r\_room\_names  
  
 return res\_13  
  
one\_to\_many, many\_to\_many = connect()  
  
def main():  
 print('Задание А1')  
 print(Task\_1(one\_to\_many))  
 print('Задание А2')  
 print(Task\_2(one\_to\_many))  
 print('Задание А3')  
 print(Task\_3(one\_to\_many))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Test\_TDD.py**

import pytest  
  
from main import \*  
  
def test\_1():  
 temp = Task\_1(one\_to\_many)  
  
 assert temp[0] == ('ASUS ROG', 120000, 'киберспортивный класс')  
 assert temp[1] == ('ASUS TUF', 100000, 'киберспортивный класс')  
 assert temp[2] == ('Asus Vivobook', 70000, 'просто класс')  
 assert temp[3] == ('MacBook Air m1', 80000, 'яблочный класс')  
 assert temp[4] == ('MacBook Air m2', 110000, 'яблочный класс')  
  
def test\_2():  
 temp = Task\_2(one\_to\_many)  
 assert temp[0] == ('киберспортивный класс', 220000)  
 assert temp[1] == ('яблочный класс', 190000)  
 assert temp[2] == ('просто класс', 70000)  
  
def test\_3():  
 temp = Task\_3(one\_to\_many)  
 assert temp == {'киберспортивный класс': ['ASUS ROG', 'ASUS TUF'], 'яблочный класс': ['MacBook Air m1', 'MacBook Air m2']}

**Результат тестирования**

****