Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждениевысшего образования «Московский государственный технический университетимени 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

```python
# 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"""
```

```python
    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 A1')
    print(Task_1(one_to_many))
    print('Задание A2')
    print(Task_2(one_to_many))
    print('Задание A3')
    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']}
```