МГТУ им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Рубежный контроль №2
«Базовые компоненты интернет-технологий»

Студент группы ИУ5-31Б:
Слепченкова Светлана Дмитриевна

Преподаватель кафедры ИУ5:
Гапанюк Юрий Евгеньевич

Москва, 2022

Условия РК1

Вариант А. Предметная область 19.

1. «Производитель» и «Деталь» связаны соотношением один-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по отделам, сортировка по сотрудникам произвольная.
2. «Производитель» и «Деталь» связаны соотношением один-ко-многим. Выведите список отделов с суммарной зарплатой сотрудников в каждом отделе, отсортированный по суммарной зарплате.
3. «Производитель» и «Деталь» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых в названии присутствует слово «ООО.», и список работающих в них сотрудников.

Условия РК2

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).

Листинг программы

RK2.py

```
# -*- coding: cp1251 -*-
from operator import itemgetter
class Detail: #Деталь
    def __init__(self, id, name, cost, count_det, det_id):
        self.id = id
        self.name = name
        self.cost = cost
        self.det_id = det_id
        self.count = count_det

class Manuf: #Dep
    #Производитель
    def __init__(self, id, name):
        self.id = id
        self.name = name

class DetManuf: #DetManuf
    #'производимые производителем детали' для реализации связи многие-ко-многим
    def __init__(self, dep_id, det_id):
        self.dep_id = dep_id
        self.det_id = det_id

# Производители
Manufs = [
    Manuf(1, 'ОАО."РЖД"'),
    Manuf(2, 'ООО."Яндекс"'),
    Manuf(3, 'ООО."СветОчка"'),
]

# Детали
Details = [
```

```
    Detail(1, "Гайка", 450, 400, 1),
    Detail(2, "Винт", 390, 1000, 1),

    Detail(3, "Винт", 600, 50, 2),

    Detail(4, "Гильза", 630, 100, 3),
    Detail(5, "Фланец", 880, 370, 3),
]

# Связи
Detail_Manuf = [
    DetManuf(1,1),
    DetManuf(2,2),
    DetManuf(3,3),
]
def connections():
    # Соединение данных один-ко-многим
    one_to_many = [(e.name, e.cost, d.name, e.count)
        for d in Manufs
        for e in Details
        if e.det_id == d.id]
    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.name, ed.dep_id, ed.det_id)
        for d in Manufs
        for ed in Detail_Manuf
        if d.id == ed.dep_id]
    many_to_many = [(e.name, e.cost, dep_name)
        for dep_name, dep_id, det_id in many_to_many_temp
        for e in Details if e.id == det_id]
    return one_to_many, many_to_many


# Переделанные задания из Rk1 для модульного тестирования
def Example_1(one_to_many):
    return sorted(one_to_many, key = itemgetter(2))
def Example_2(one_to_many):
    res_12_unsorted = []
    # Перебираем всех производителей
    for d in Manufs:
        # Список деталей рооизводителей
        d_details = list(filter(lambda i: i[2] == d.name, one_to_many))
        if len(d_details) > 0:
            # Стоимоть деталей у производителя
            d_sals = [cost*count for _,cost,_,count in d_details]
            # Суммарная стоимость деталей у производителя
            d_sals_sum = sum(d_sals)
            res_12_unsorted.append((d.name, d_sals_sum))
    # Сортировка по стоимости имеющихся деталей
    return sorted(res_12_unsorted, key = itemgetter(1), reverse = True)
def Example_3(one_to_many):
    res_13 = {}
    # Перебираем всех производителей (будем проверять наличие слова: "ООО.")
    for d in Manufs:
        if 'ООО.' in d.name:
            d_detailz = list(filter(lambda i: i[2] == d.name, one_to_many))
            if len(d_detailz) > 0:
                # Только название деталей
                d_detailz_names = [x for x,_,_,_ in d_detailz]
                # Добавляем результат в словарь
                # ключ - производитель, значение - список деталей
                res_13[d.name] = d_detailz_names
    return res_13

one_to_many, many_to_many = connections()
```

```
if __name__ == '__main__':
    print("Example A1\n", Example_1(one_to_many))
    print("Example A2\n", Example_2(one_to_many))
    print("Example A1\n", Example_3(one_to_many))
```

test_TDD.py

```
# -*- coding: cp1251 -*-
import pytest
from RK2 import Example_1, Example_2, Example_3, one_to_many

def test_result_example_1():
    temp = Example_1(one_to_many)

    assert temp[0] == ('Гильза', 630, 'ООО."СветОчка"', 100)
    assert temp[1] == ('Фланец', 880, 'ООО."СветОчка"', 370)
    assert temp[2] == ('Винт', 600, 'ООО."Яндекс"', 50)
    assert temp[3] == ('Гайка', 450, 'ОАО."РЖД"', 400)
    assert temp[4] == ('Винт', 390, 'ОАО."РЖД"', 1000)


def test_result_example_2():
    temp = Example_2(one_to_many)

    assert temp[0] == ('ОАО."РЖД"', 570000)
    assert temp[1] == ('ООО."СветОчка"', 388600)
    assert temp[2] == ('ООО."Яндекс"', 30000)

def test_result_example_3():
    temp = Example_3(one_to_many)

    assert temp == {'ООО."Яндекс"': ['Винт'], 'ООО."СветОчка"': ['Гильза',
'Фланец']}
```

Результаты выполнения модульного тестирования: