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

# Рубежный контроль №2
# «Базовые компоненты интернет-технологий»

Студент группы ИУ5-31Б:
Строганов Георгий Константинович

Преподаватель кафедры ИУ5:
Гапанюк Юрий Евгеньевич

Москва, 2022

## Условия РК-1

### Вариант А. Предметная область 20.

1. «Поставщик» и «Деталь» связаны соотношением один-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по отделам, сортировка по сотрудникам произвольная.
2. «Поставщик» и «Деталь» связаны соотношением один-ко-многим. Выведите список отделов с суммарной зарплатой сотрудников в каждом отделе, отсортированный по суммарной зарплате.
3. «Поставщик» 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 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).

## Листинг программы

### RK_2.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 Provider:
    #Поставщик
    def __init__(self, id, name):
        self.id = id
        self.name = name

class DetProv:
    #'Поставляемые поставщиком детали' для реализации связи многие-ко-многим
    def __init__(self, dep_id, det_id):
        self.dep_id = dep_id
        self.det_id = det_id

# Поставщики
Providers = [
    Provider(1, 'ООО."Рога и копыта"'),
    Provider(2, 'ООО."Моя оборона"'),
    Provider(3, 'ИП."Рога и подковы"'),
]
```

```python
# Детали
Details = [
    Detail(1, "Болт", 250, 300, 1),
    Detail(2, "Саморез", 375, 200, 1),

    Detail(3, "Вкладыш", 450, 50, 2),
    Detail(4, "Бобышка", 350, 60, 2),

    Detail(5, "Штуцер", 500, 100, 3),
]

# Связи деталей и поставщиков
Detail_Provider = [
    DetProv(1,1),
    DetProv(2,2),
    DetProv(3,3),
]

# Все связи
def connections():
    # Соединение данных один-ко-многим
    one_to_many = [(e.name, e.cost, d.name, e.count)
        for d in Providers
        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 Providers
        for ed in Detail_Provider
        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

# Вывод результатов заданий
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 поставщиков
    for d in Providers:
        # Список деталей поставщиков
        d_details = list(filter(lambda i: i[2] == d.name, one_to_many))
        if len(d_details) > 0:
            # Стоимоть деталей у постащика
            d_sals = [cost*count for _,cost,_,count in d_details]
            # Суммарная стоимость деталей у поставщика
            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 поставщиков (будем проверять наличие слова: "ООО.")
    for d in Providers:
        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 - поставщик, значение - список деталей
```

```
                res_13[d.name] = d_detailz_names
    return res_13

one_to_many, many_to_many = connections()

def main():
    print('Example A1')
    print(Example_1(one_to_many))
    print('Example A2')
    print(Example_2(one_to_many))
    print('Example A3')
    print(Example_3(one_to_many))

if __name__ == '__main__':
    main()
```

## Test_TDD_RK_2.py

```
# -*- coding: cp1251 -*-
import pytest
from RK_2 import Example_1
from RK_2 import Example_2
from RK_2 import Example_3
from RK_2 import one_to_many

#Разделил на элементы для удобства чтения

def test_result_example_1():
    temp = Example_1(one_to_many)

    assert temp[0] == ('Вкладыш', 450, 'ООО."Моя оборона"', 50)
    assert temp[1] == ('Бобышка', 350, 'ООО."Моя оборона"', 60)
    assert temp[2] == ('Болт', 250, 'ООО."Рога и копыта"', 300)
    assert temp[3] == ('Саморез', 375, 'ООО."Рога и копыта"', 200)
    assert temp[4] == ('Штуцер', 500, 'ИП."Рога и подковы"', 100)

def test_result_example_2():
    temp = Example_2(one_to_many)

    assert temp[0] == ('ООО."Рога и копыта"', 150000)
    assert temp[1] == ('ИП."Рога и подковы"', 50000)
    assert temp[2] == ('ООО."Моя оборона"', 43500)

def test_result_example_3():
    temp = Example_3(one_to_many)

    assert temp == {'ООО."Рога и копыта"': ['Болт', 'Саморез'], 'ООО."Моя оборона"':
['Вкладыш', 'Бобышка']}
```

## Результаты выполнения модульного тестирования: