Московский государственный технический университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Система обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Рубежный контроль №2

Вариант Г №20

Выполнил:

Студент ИУ5-34Б

Фролов М. К.

Подпись и Дата:

Проверил:

Преподаватель каф. ИУ5

Гапанюк Ю. Е.

Подпись и Дата:

Москва, 2023 г

ТЕКСТ ПРОГРАММЫ

Файл rk1v2.py (Файл из рк1 после рефакторинга):

```
# Вариант Г.
# 1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список
всех отделов,
# у которых название начинается с буквы «А», и список работающих в
нихсотрудников.
# 2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список
# отделов с максимальной зарплатой сотрудников в каждом отделе, отсортированный
по максимальной зарплате.
# 3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите
# список всех связанных сотрудников и отделов, отсортированный по отделам,
# сортировка по сотрудникам произвольная.
# вариант 20 - деталь и поставщик
# используется для сортировки
from operator import itemgetter
class Detail:                                                    #класс-
реализация деталей
    def __init__(self, id, name, price, sup_id):
        self.id = id
        self.name = name
        self.price = price
        self.sup_id = sup_id
class Suppliers:                                                 #класс-
реализация поставщиков
    def __init__(self, id, name):
        self.id = id
        self.name = name
class EmpDep:                                                    # класс
для реализации связи многие-ко-многим
    def __init__(self, sup_id, det_id):
        self.sup_id = sup_id
        self.det_id = det_id

def a1_solution(one_to_many):
    res_11 = sorted(one_to_many, key=itemgetter(2))
    i = 0
    flags = []
    while i < len(res_11):
        if res_11[i][2][0] == "A":
            flags.append(i)
        i += 1
    return [res_11[i] for i in flags]

def a2_solution(one_to_many):
    res_12_unsorted = []
    for d in suppliers:
        d_dets = list(filter(lambda i: i[2]==d.name, one_to_many))
        if len(d_dets) > 0:
```

```
            d_prices = [price for _,price,_ in d_dets]
            d_sals_sum = sum(d_prices)
            res_12_unsorted.append((d.name, d_sals_sum))
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return res_12

def a3_solution(many_to_many):
    res_13 = {}
    for d in suppliers:
        d_emps = list(filter(lambda i: i[2]==d.name, many_to_many))
        d_emps_names = [x for x,_,_ in d_emps]
        res_13[d.name] = d_emps_names
    return dict(sorted(res_13.items()))
suppliers = [                                                    #
сведения о поставщиках
    Suppliers(1, 'Всё для дома поставщик'),
    Suppliers(2, 'Поставкиру поставщик'),
    Suppliers(3, 'Аризона поставщик'),
    Suppliers(11, 'Всё для дома поставщик(другой)'),
    Suppliers(22, 'Поставкиру поставщик(другой)'),
    Suppliers(33, 'Аризона поставщик(другая)'),
]
det_names = ["Гвоздь", "Шестерня", "Крышка", "Втулка", "Винт"]
details = [                                                      #
сведения о деталях + случайными числами
    Detail(1, det_names[0], 5000, 1),
    Detail(2, det_names[1], 6000, 2),
    Detail(3, det_names[2], 7000, 3),
    Detail(4, det_names[3], 8000, 3),
    Detail(5, det_names[4], 9000, 3),
]
emps_deps = [
    EmpDep(1,1),
    EmpDep(2,2),
    EmpDep(3,3),
    EmpDep(3,4),
    EmpDep(3,5),
    EmpDep(11,1),
    EmpDep(22,2),
    EmpDep(33,3),
    EmpDep(33,4),
    EmpDep(33,5),
]
def main(): #сновная функция
    # Соединение данных один-ко-многим
    one_to_many = [(e.name, e.price, d.name) for d in suppliers for e in details
if e.sup_id==d.id]
    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.name, ed.sup_id, ed.det_id) for d in suppliers for ed
in emps_deps if d.id==ed.sup_id]
```

```
    many_to_many = [(e.name, e.price, sup_name) for sup_name, sup_id, det_id in
many_to_many_temp for e in details if e.id==det_id]
    print('Задание А1')
    print(a1_solution(one_to_many))
    print('\nЗадание А2')
    print(a2_solution(one_to_many))
    print('\nЗадание А3')
    print(a3_solution(many_to_many))
if __name__ == '__main__':
    main()
```

ПРОГРАММА ТЕСТОВ

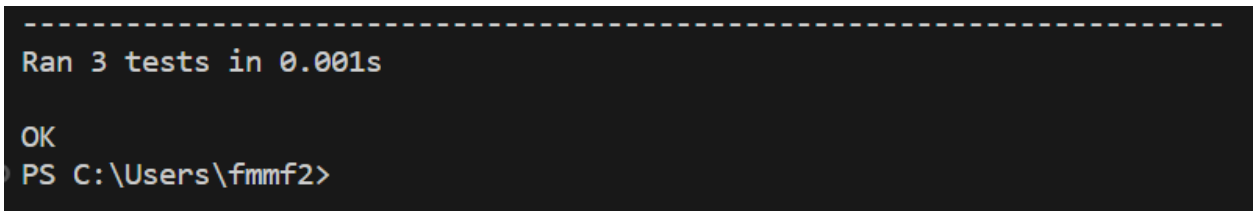
файл rk2.py

```
import unittest
from rk1v2 import *
class Test_Program(unittest.TestCase):

    suppliers = [                                                    #
сведения о поставщиках
        Suppliers(1, 'Всё для дома поставщик'),
        Suppliers(2, 'Поставкиру поставщик'),
        Suppliers(3, 'Аризона поставщик'),
        Suppliers(11, 'Всё для дома поставщик(другой)'),
        Suppliers(22, 'Поставкиру поставщик(другой)'),
        Suppliers(33, 'Аризона поставщик(другая)'),
    ]

    det_names = ["Гвоздь", "Шестерня", "Крышка", "Втулка", "Винт"]

    details = [                                                      #
сведения о деталях + случайными числами
        Detail(1, det_names[0], 5000, 1),
        Detail(2, det_names[1], 6000, 2),
        Detail(3, det_names[2], 7000, 3),
        Detail(4, det_names[3], 8000, 3),
        Detail(5, det_names[4], 9000, 3),
    ]

    emps_deps = [
        EmpDep(1,1),
        EmpDep(2,2),
        EmpDep(3,3),
        EmpDep(3,4),
        EmpDep(3,5),
        EmpDep(11,1),
        EmpDep(22,2),
        EmpDep(33,3),
```

```
        EmpDep(33,4),
        EmpDep(33,5),
    ]
    def test_a1(self):
        one_to_many = [(e.name, e.price, d.name)
                       for d in suppliers
                       for e in details
                       if e.sup_id==d.id]
        self.assertEqual(a1_solution(one_to_many), [('Крышка', 7000, 'Аризона
поставщик'),
                                                    ('Втулка', 8000, 'Аризона
поставщик'),
                                                    ('Винт', 9000, 'Аризона
поставщик')])
    def test_a2(self):
         one_to_many = [(e.name, e.price, d.name)
                        for d in suppliers
                        for e in details
                        if e.sup_id==d.id]
         self.assertEqual(a2_solution(one_to_many), [('Аризона поставщик',
24000),
                                                     ('Поставкиру поставщик',
6000),
                                                     ('Всё для дома поставщик',
5000)])
    def test_a3(self):
        many_to_many_temp = [(d.name, ed.sup_id, ed.det_id)
                             for d in suppliers
                             for ed in emps_deps
                             if d.id==ed.sup_id]
        many_to_many = [(e.name, e.price, sup_name)
                        for sup_name, sup_id, det_id in many_to_many_temp
                        for e in details
                        if e.id==det_id]
        self.assertDictEqual(a3_solution(many_to_many), {'Аризона поставщик':
['Крышка', 'Втулка', 'Винт'],
                                                         'Аризона
поставщик(другая)': ['Крышка', 'Втулка', 'Винт'],
                                                         'Всё для дома
поставщик': ['Гвоздь'],
                                                         'Всё для дома
поставщик(другой)': ['Гвоздь'],
                                                         'Поставкиру поставщик':
['Шестерня'],
                                                         'Поставкиру
поставщик(другой)': ['Шестерня']})

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

РЕЗУЛЬТАТЫ ТЕСТОВ

При правильном прохождении:



При неправильном прохождении тестов: