

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю №2
Вариант 19Д

Выполнил:
студент группы ИУ5-35Б
Ходырев Роман
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич
Подпись и дата:

Постановка задачи

Задача для РК1

1. «Деталь» и «Производитель» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их деталей.
2. «Деталь» и «Производитель» связаны соотношением один-ко-многим. Выведите список производителей с его средней зарплатой за одну деталь, отсортированный по убыванию зарплаты.
3. «Деталь» и «Производитель» связаны соотношением многие-ко-многим. Вывести список сотрудников, которых нужно выбрать для производства робота-пылесоса (детали, необходимые для производства робота пылесоса будут даны в листинге программы). Если одну деталь могут сделать несколько производителей, необходимо выбрать того, чья зарплата за одну деталь будет меньше. Так же вывести итоговые затраты на производство робота-пылесоса

Задача для РК2

1. Создать функцию test.py для тестирования основного кода. Функция test.py должна тестировать все 3 функции – task1, task2 и task3 – выполняющие задания 1-3 для РК1

Текст программы

classes.py

```
class Manufacturer:
    """Производитель"""
    def __init__(self, id, name, salary):
        self.id = id
        self.name = name
        self.salary = salary

class Detail:
    """Деталь"""
    def __init__(self, id, name, man_id):
        self.id = id
        self.name = name
        self.man_id = man_id

class ManufacturerDetail:
    """Реализуем связь многие ко многим"""
    def __init__(self, man_id, det_id):
        self.man_id = man_id
        self.det_id = det_id
```

create_object.py

```
from classes import Detail, Manufacturer, ManufacturerDetail

#Производители
manufacturers = [
    Manufacturer(1, "Абалуев", 35000),
    Manufacturer(2, "Андрест", 40000),
    Manufacturer(3, "Аннакулиева", 55000),
    Manufacturer(4, "Вопияшин", 18000),
    Manufacturer(5, "Гонов", 45000),
    Manufacturer(6, "Идрисов", 35000),
    Manufacturer(7, "Ларин", 65000),
    Manufacturer(8, "Лахин", 50000),
    Manufacturer(9, "Новицкий", 25000),
    Manufacturer(10, "Пермяков", 20000),
    Manufacturer(11, "Расулов", 100000),
    Manufacturer(12, "Сироткин", 58000),
    Manufacturer(13, "Стрельцов", 64000),
    Manufacturer(14, "Удалова", 39000),
    Manufacturer(15, "Ходырев", 235000),
    Manufacturer(16, "Шакиров", 85000),
    Manufacturer(17, "Шиленок", 105000),
    Manufacturer(18, "Якимова", 88000)
]

#Детали
details = [
    Detail(1, 'Подшипник', 1),
    Detail(2, 'Вал', 2),
    Detail(3, 'Кольцо', 3),
    Detail(4, 'Пружина', 4),
    Detail(5, 'Шестерня', 5),
    Detail(6, 'Магнит', 6),
    Detail(7, 'Ротор', 7),
    Detail(8, 'Статор', 8),
    Detail(9, 'Клапан', 9),
    Detail(10, 'Компрессор', 10),
    Detail(11, 'Шкив', 11),
    Detail(12, 'Ремень', 12),
    Detail(13, 'Шпиндель', 13),
    Detail(14, 'Плита', 14),
    Detail(15, 'Фильтр', 15),
    Detail(16, 'Сенсор', 16),
    Detail(17, 'Трубка', 17),
    Detail(18, 'Регулятор', 18),
    Detail(19, 'Датчик', 1),
    Detail(20, 'Крепеж', 2),
    Detail(21, 'Кабель', 3),
    Detail(22, 'Ручка', 4),
    Detail(23, 'Панель', 5),
    Detail(24, 'Датчик', 6),
    Detail(25, 'Вентиль', 7),
```

```
Detail(26, 'Ключ', 8),
Detail(27, 'Аккумулятор', 9),
Detail(28, 'Диск', 10),
Detail(29, 'Дверь', 11),
Detail(30, 'Замок', 12),
Detail(31, 'Болт', 13),
Detail(32, 'Гайка', 14),
Detail(33, 'Плата', 15),
Detail(34, 'Микросхема', 16),
Detail(35, 'Реле', 17),
Detail(36, 'Провод', 18),
Detail(37, 'Диод', 1),
Detail(38, 'Компонент', 2),
Detail(39, 'Батарейка', 3),
Detail(40, 'Мотор', 4),
Detail(41, 'Кнопка', 5),
Detail(42, 'Шланг', 6),
Detail(43, 'Штекер', 7),
Detail(44, 'Разъем', 8),
Detail(45, 'Контроллер', 9),
Detail(46, 'Пульт', 10),
Detail(47, 'Экран', 11),
Detail(48, 'Рамка', 12),
Detail(49, 'Винт', 13),
Detail(50, 'Ручка-кран', 14)
```

```
]
```

```
#Связь многие-ко-многим
```

```
manufacturer_details = [
    ManufacturerDetail(1,1),
    ManufacturerDetail(2,2),
    ManufacturerDetail(3,3),
    ManufacturerDetail(4,4),
    ManufacturerDetail(5,5),
    ManufacturerDetail(6,6),
    ManufacturerDetail(7,7),
    ManufacturerDetail(8,8),
    ManufacturerDetail(9,9),
    ManufacturerDetail(10,10),
    ManufacturerDetail(11,11),
    ManufacturerDetail(12,12),
    ManufacturerDetail(13,13),
    ManufacturerDetail(14,14),
    ManufacturerDetail(15,15),
    ManufacturerDetail(16,16),
    ManufacturerDetail(17,17),
    ManufacturerDetail(18,18),
    ManufacturerDetail(1,19),
    ManufacturerDetail(2,20),
    ManufacturerDetail(3,21),
    ManufacturerDetail(4,22),
    ManufacturerDetail(5,23),
    ManufacturerDetail(6,24),
```

```
ManufacturerDetail(7,25),
ManufacturerDetail(8,26),
ManufacturerDetail(9,27),
ManufacturerDetail(10,28),
ManufacturerDetail(11,29),
ManufacturerDetail(12,30),
ManufacturerDetail(13,31),
ManufacturerDetail(14,32),
ManufacturerDetail(15,33),
ManufacturerDetail(16,34),
ManufacturerDetail(17,35),
ManufacturerDetail(18,36),
ManufacturerDetail(1,37),
ManufacturerDetail(2,38),
ManufacturerDetail(3,39),
ManufacturerDetail(4,40),
ManufacturerDetail(5,41),
ManufacturerDetail(6,42),
ManufacturerDetail(7,43),
ManufacturerDetail(8,44),
ManufacturerDetail(9,45),
ManufacturerDetail(10,46),
ManufacturerDetail(11,47),
ManufacturerDetail(12,48),
ManufacturerDetail(13,49),
ManufacturerDetail(14,50),
ManufacturerDetail(4,44),
ManufacturerDetail(8,16),
ManufacturerDetail(8,39),
ManufacturerDetail(3,50),
ManufacturerDetail(8,12),
ManufacturerDetail(13,49),
ManufacturerDetail(14,39),
ManufacturerDetail(15,11),
ManufacturerDetail(5,42),
ManufacturerDetail(3,43),
ManufacturerDetail(3,45),
ManufacturerDetail(17,26),
ManufacturerDetail(10,50),
ManufacturerDetail(18,39),
ManufacturerDetail(15,13),
ManufacturerDetail(9,3),
ManufacturerDetail(4,27),
ManufacturerDetail(13,38),
ManufacturerDetail(3,45),
ManufacturerDetail(5,42),
ManufacturerDetail(12,32),
ManufacturerDetail(5,28),
ManufacturerDetail(16,46),
ManufacturerDetail(1,41),
ManufacturerDetail(12,10),
ManufacturerDetail(4,3),
ManufacturerDetail(18,30),
```

```
ManufacturerDetail(17,40),
ManufacturerDetail(13,43),
ManufacturerDetail(13,18),
ManufacturerDetail(10,34),
ManufacturerDetail(10,47),
ManufacturerDetail(17,12),
ManufacturerDetail(16,12),
ManufacturerDetail(13,11),
ManufacturerDetail(4,46),
ManufacturerDetail(1,21),
ManufacturerDetail(16,48),
ManufacturerDetail(9,41),
ManufacturerDetail(15,34),
ManufacturerDetail(12,27),
ManufacturerDetail(15,11),
ManufacturerDetail(8,15),
ManufacturerDetail(7,25),
ManufacturerDetail(15,46),
ManufacturerDetail(12,36),
ManufacturerDetail(16,49),
ManufacturerDetail(6,37),
ManufacturerDetail(5,14),
ManufacturerDetail(10,21)
```

```
]
```

main.py (внесены изменения для тестирования)

```
from classes import Manufacturer, Detail, ManufacturerDetail
from create_object import manufacturers, details, manufacturer_details
from tabulate import tabulate
from operator import itemgetter

def task1(one_to_many):
    # Выводим производителей, их зарплаты и их детали если
    # фамилия сотрудника заканчивается на ов
    res_11_det = {}
    res_11_sal = {}
    for x in one_to_many:
        man_name = x[0]
        man_salary = x[1]
        det = x[2]

        res_11_det.setdefault(man_name, []).append(det)
        res_11_sal[man_name] = man_salary

    data = []
    for name, dets in res_11_det.items():
        if str(name).endswith('ов'):
            data.append([name, res_11_sal[name], dets])

    return data

def task2(manufacturers, one_to_many):
```

```

# Отсортируем по убыванию стоимость детали у одного производителя
# то есть делим зарплату на количество деталей сотрудника
res_12_uns = []
for m in manufacturers:
    #Список деталей сотрудника
    m_details = list(filter(lambda i:i[0]==m.name, one_to_many))

    if len(m_details)>0:
        res_12_uns.append((m.name, round(m.salary/len(m_details), 2)))

#Сортировка по цене за деталь
return res_12_uns

# Для создания робота-пылесоса нужно сделать эти детали.
robot_vacuum = [
    'Мотор',
    'Датчик',
    'Кнопка',
    'Ключ',
    'Крепеж',
    'Реле',
    'Контроллер',
    'Дисплей',
    'Панель',
    'Фильтр',
]
# Опреде

def task3(robot_vacuum, many_to_many, manufacturers, one_to_many):
    d_emps = {}
    for d in robot_vacuum:
        for detail, name in many_to_many:
            if d == detail:
                d_emps.setdefault(d, set()).add(name)

    cheap_mans = sorted(task2(manufacturers, one_to_many), key=itemgetter(1))
    res_13 = {}
    sum_prod = []
    for dtl, mans in d_emps.items():
        for cheap_man in cheap_mans:
            if cheap_man[0] in mans:
                res_13[dtl] = cheap_man[0]
                sum_prod.append(float(cheap_man[1]))
                break

    data = []
    i = 0
    for dtl, man in res_13.items():
        data.append([dtl, man, sum_prod[i]])
        i += 1

    total_cost = sum(sum_prod)
    return data, total_cost

```

```

def main():
    """Основная функция"""
    print('Задание Д1')
    table = table = tabulate(task1(one_to_many), headers=["Производитель", "Зарплата",
"Детали"], tablefmt="pretty")
    print(table)

    print('\nЗадание Д2')
    res_12 = sorted(task2(manufacturers, one_to_many), key=itemgetter(1), reverse=True)
    table2 = tabulate(res_12, headers=["Производитель", "Стоимость одной детали"],
tablefmt="pretty")
    print(table2)

    print('\nЗадание Д3')
    data3, total = task3(robot_vacuum, many_to_many, manufacturers, one_to_many)
    table3 = tabulate(data3, headers=["Деталь", "Производитель", "Стоимость"],
tablefmt="pretty")
    print(table3)
    print(f"Затраты на производство робота пылесоса: {total}")

    a = input()

if __name__ == "__main__":
    # Соединение данных один-ко-многим
    one_to_many = [(m.name, m.salary, d.name)
        for m in manufacturers
        for d in details
        if m.id==d.man_id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(m.name, md.man_id, md.det_id)
        for m in manufacturers
        for md in manufacturer_details
        if m.id==md.man_id]

    many_to_many = [(d.name, man_name)
        for man_name, man_id, det_id in many_to_many_temp
        for d in details
        if d.id==det_id]

    main()

```


test.py (само тестирование для PK2)

```
import unittest
from main import task1, task2, task3
from classes import Manufacturer, Detail, ManufacturerDetail

class TestTasks(unittest.TestCase):
    def setUp(self):
        # Создаем данные для тестирования
        self.manufacturers = [
            Manufacturer(1, "Иванов", 100),
            Manufacturer(2, "Петров", 200),
        ]

        self.details = [
            Detail(1, 'X', 1),
            Detail(2, 'Y', 2),
        ]

        self.manufacturer_details = [
            ManufacturerDetail(1, 1),
            ManufacturerDetail(2, 2),
        ]

        self.robot_vacuum = ['X', 'Y']

    def test_task1(self):
        # Тестирование task1
        one_to_many = [(m.name, m.salary, d.name) for m in self.manufacturers for d in
self.details if m.id == d.man_id]
        expected_result = [
            ['Иванов', 100, ['X']],
            ['Петров', 200, ['Y']],
        ]
        result = task1(one_to_many)
        self.assertEqual(result, expected_result)

    def test_task2(self):
        # Тестирование task2
        one_to_many = [(m.name, m.salary, d.name) for m in self.manufacturers for d in
self.details if m.id == d.man_id]
        expected_result = [('Иванов', 100), ('Петров', 200)]
        result = task2(self.manufacturers, one_to_many)
        self.assertEqual(result, expected_result)

    def test_task3(self):
        # Тестирование task3
        one_to_many = [(m.name, m.salary, d.name) for m in self.manufacturers for d in
self.details if m.id == d.man_id]
        many_to_many = [(d.name, m.name) for m in self.manufacturers for md in
self.manufacturer_details for d in self.details if m.id == md.man_id and md.det_id ==
d.id]
        expected_result = [
            ['X', 'Иванов', 100.0],
```

```

        ['Y', 'Петров', 200.0],
    ]
    result, cost = task3(self.robot_vacuum, many_to_many, self.manufacturers,
one_to_many)
    self.assertEqual(cost, 300.0)
    self.assertEqual(result, expected_result)

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

```

Анализ результатов тестирования

```

PS F:\Рабочий стол\3 сем\PCPL\Labs> & C:/User
...
-----
Ran 3 tests in 0.001s

OK

```

Анализ результатов выполнения основной функции

1) Задание Д1

Задание Д1			
Производитель	Зарплата	Детали	
Гонов	45000	['Шестерня', 'Панель', 'Кнопка']	
Идрисов	35000	['Магнит', 'Датчик', 'Шланг']	
Пермяков	20000	['Компрессор', 'Диск', 'Пульт']	
Расулов	100000	['Шкив', 'Дверь', 'Экран']	
Стрельцов	64000	['Шпиндель', 'Болт', 'Винт']	
Шакиров	85000	['Сенсор', 'Микросхема']	

2) Задание Д2

Задание Д2	
Производитель	Стоимость одной детали
Ходырев	117500.0
Шиленок	52500.0
Якимова	44000.0
Шакиров	42500.0
Расулов	33333.33
Ларин	21666.67
Стрельцов	21333.33
Сироткин	19333.33
Аннакулиева	18333.33
Лахин	16666.67
Гонов	15000.0
Андрест	13333.33
Удалова	13000.0
Абалуев	11666.67
Идрисов	11666.67
Новицкий	8333.33
Пермяков	6666.67
Вопияшин	6000.0

3) Задание Д3

Задание Д3		
Деталь	Производитель	Стоимость
Мотор	Вопияшин	6000.0
Датчик	Абалуев	11666.67
Кнопка	Новицкий	8333.33
Ключ	Лахин	16666.67
Крепеж	Андрест	13333.33
Реле	Шиленок	52500.0
Контроллер	Новицкий	8333.33
Панель	Гонов	15000.0
Фильтр	Лахин	16666.67
Затраты на производство робота-пылесоса: 148500.0		