

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
им. Н.Э. Баумана

Факультет “Информатика и системы управления”  
Кафедра “Системы обработки информации и управления”



Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по Рубежному Контролю №1  
Вариант Г 17

**Выполнил:**  
Студент группы ИУ5-35Б  
Костылев М.С.  
**Преподаватель:**  
Гапанюк Ю.Е.

Москва 2025

## Задание

1) Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.

Пример классов данных для предметной области Сотрудник-Отдел:

1. Класс «Сотрудник», содержащий поля:
  - ID записи о сотруднике;
  - Фамилия сотрудника;
  - Зарплата (количественный признак);
  - ID записи об отделе. (для реализации связи один-ко-многим)
2. Класс «Отдел», содержащий поля:
  - ID записи об отделе;
  - Наименование отдела.
3. (Для реализации связи многие-ко-многим) Класс «Сотрудники отдела», содержащий поля:
  - ID записи о сотруднике;
  - ID записи об отделе.

2) Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.

3) Необходимо разработать запросы в соответствии с Вашим вариантом. Запросы сформулированы в терминах классов «Сотрудник» и «Отдел», которые используются в примере. Вам нужно перенести эти требования в Ваш вариант предметной области. При разработке запросов необходимо по возможности использовать функциональные возможности языка Python (list/dict comprehensions, функции высших порядков).

Для реализации запроса №2 введите в класс, находящийся на стороне связи «много», произвольный количественный признак, например, «зарплата сотрудника».

## Задание варианта

### 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 них сотрудников.
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.

## Код

```
from operator import itemgetter

#Вариант Г 17

class Dirigent:
    """Дирижёр"""
    def __init__(self, id, fio, salary, orchestra_id):
        self.id = id
        self.fio = fio
        self.salary = salary
        self.orchestra_id = orchestra_id

class Orchestra:
    """Оркестр"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class DirigentOrchestra:
    """Связь многие-ко-многим между дирижёрами и оркестрами"""
    def __init__(self, dirigent_id, orchestra_id):
        self.dirigent_id = dirigent_id
        self.orchestra_id = orchestra_id
```

```
orchestras = [  
    Orchestra(1, 'Академический оркестр Москвы'),  
    Orchestra(2, 'Балтийский оркестр'),  
    Orchestra(3, 'Альфа-оркестр Санкт-Петербурга'),  
    Orchestra(4, 'Оркестр камерной музыки'),  
]
```

```
dirigents = [  
    Dirigent(1, 'Артамонов', 85000, 1),  
    Dirigent(2, 'Белов', 90000, 2),  
    Dirigent(3, 'Андреев', 78000, 3),  
    Dirigent(4, 'Иванов', 95000, 1),  
    Dirigent(5, 'Кузнецов', 87000, 4),  
]
```

```
dir_orch = [  
    DirigentOrchestra(1, 1),  
    DirigentOrchestra(2, 2),  
    DirigentOrchestra(3, 3),  
    DirigentOrchestra(4, 1),  
    DirigentOrchestra(5, 4),  
    DirigentOrchestra(2, 4),  
]
```

```
def main():
```

```
    one_to_many = [(d.fio, d.salary, o.name)  
                    for o in orchestras  
                    for d in dirigents  
                    if d.orchestra_id == o.id]
```

```
    many_to_many_temp = [(o.name, do.orchestra_id, do.dirigent_id)  
                          for o in orchestras  
                          for do in dir_orch]
```

```

        if o.id == do.orchestra_id]

many_to_many = [(d.fio, d.salary, orch_name)
                 for orch_name, orch_id, dir_id in many_to_many_temp
                 for d in dirigents if d.id == dir_id]

print("Задание Г1")
res_1 = {}
for o in orchestras:
    if o.name.startswith('A'):
        o_dirigs = [x for x, _, orch in one_to_many if orch == o.name]
        res_1[o.name] = o_dirigs
print(res_1)

print("\nЗадание Г2")
res_2 = []
for o in orchestras:
    o_dirigs = list(filter(lambda i: i[2] == o.name, one_to_many))
    if o_dirigs:
        max_salary = max([sal for _, sal, _ in o_dirigs])
        res_2.append((o.name, max_salary))
res_2_sorted = sorted(res_2, key=itemgetter(1), reverse=True)
print(res_2_sorted)

print("\nЗадание Г3")
res_3 = sorted(many_to_many, key=itemgetter(2))
print(res_3)

if __name__ == "__main__":
    main()

```

**Скриншоты работы программы**

```
spiral@LAPTOP-L1VU569D:~/Kostylev_3SEM_25/RK1$ python3 code.py
Задание Г1
{'Академический оркестр Москвы': ['Артамонов', 'Иванов'], 'Альфа-оркестр Санкт-Петербурга': ['Андреев']}

Задание Г2
[('Академический оркестр Москвы', 95000), ('Балтийский оркестр', 90000), ('Оркестр камерной музыки', 87000), ('Альфа-оркестр Санкт-Петербурга', 78000)]

Задание Г3
[('Артамонов', 85000, 'Академический оркестр Москвы'), ('Иванов', 95000, 'Академический оркестр Москвы'), ('Андреев', 78000, 'Альфа-оркестр Санкт-Петербурга'), ('Белов', 90000, 'Балтийский оркестр'), ('Кузнецов', 87000, 'Оркестр камерной музыки'), ('Белов', 90000, 'Оркестр камерной музыки')]
spiral@LAPTOP-L1VU569D:~/Kostylev_3SEM_25/RK1$
```