

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

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

Курс «Базовые компоненты интернет-технологий»
Отчет по рубежному контролю №1

Выполнил:

студент группы ИУ5-34Б
Барabanщиков Лев

Проверил:

преподаватель каф. ИУ5
Гапанюк Ю.Е.

Подпись и дата: 24.10.



Подпись и дата:

Москва, 2021 г.

Постановка задачи

Рубежный контроль представляет собой разработку программы на языке Python, которая выполняет следующие действия:

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).

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.

Класс 1 — студент, класс 2 — группа.

Текст программы (main.py)

```
# используется для сортировки  
from operator import itemgetter
```

```
class Student:
```

```
    """Студент"""
```

```
    def __init__(self, id, fio, scholarship, group_id):
```

```
        self.id = id
```

```
        self.fullname = fio
```

```
        self.scholarship = scholarship
```

```
        self.group_id = group_id
```

```
class Group:
```

```
    """Группа"""
```

```
    def __init__(self, id, name):
```

```
        self.id = id
```

```
        self.name = name
```

```
class StudentGroup:
```

```
    """
```

```
    'Студенты группы' для реализации
```

```
    связи многие-ко-многим
```

```
    """
```

```
    def __init__(self, group_id, stud_id):
```

```
        self.group_id = group_id
```

```
        self.stud_id = stud_id
```

```
# Группы
```

```
groups = [
```

```
    Group(1, 'AK1-31'),
```

```
    Group(2, 'AK3-52'),
```

```
    Group(3, 'ФН12-32Б'),
```

```
    Group(4, 'AK1-32'),
```

```
    Group(5, 'AK3-51'),
```

```
    Group(6, 'ФН12-31Б'),
```

```
    Group(7, 'ИУ5-34Б'),
```

```
    Group(8, 'PREP-12')
```

```
]
```

Студенты

```
students = [  
    Student(1, 'Абуховский', 3500, 1),  
    Student(2, 'Бондаренко', 9000, 2),  
    Student(3, 'Козлов', 0, 3),  
    Student(4, 'Гордеев', 0, 7),  
    Student(5, 'Коновалов', 100000, 5),  
    Student(6, 'Барабанщиков', 12000, 7),  
    Student(7, "Милевич", 3500, 4)  
]
```

```
studs_groups = [  
    StudentGroup(1, 1),  
    StudentGroup(2, 2),  
    StudentGroup(3, 3),  
    StudentGroup(4, 7),  
    StudentGroup(5, 5),  
    StudentGroup(7, 4),  
    StudentGroup(7, 6),  
    StudentGroup(8, 6)  
]
```

def main():

"""Основная функция"""

Соединение данных один-ко-многим

```
one_to_many = [(s.fullname, s.scholarship, g.name)  
    for g in groups  
    for s in students  
    if s.group_id == g.id]
```

Соединение данных многие-ко-многим

```
many_to_many_temp = [(g.name, sg.group_id, sg.stud_id)  
    for g in groups  
    for sg in studs_groups  
    if g.id == sg.group_id]
```

```
many_to_many = [(s.fullname, s.scholarship, group_name)  
    for group_name, group_id, stud_id in many_to_many_temp  
    for s in students if s.id == stud_id]
```

print('Задание Г1')

res1 = {}

Перебираем все группы

```
res1 = {g.name: [x for x, _, _ in list(filter(lambda i: i[2] == g.name, one_to_many))]} for g in groups if  
    'A' == g.name[0]
```

print(res1)

print('\nЗадание Г2')

res_2_unsorted = []

```

# Перебираем все группы
for g in groups:
    # Список студентов группы
    g_studs = list(filter(lambda i: i[2] == g.name, one_to_many))
    # Если группа не пустая
    if len(g_studs) > 0:
        # Считаем максимальные стипендии
        g_scholarship_max = max(g_studs, key=lambda x: x[1])[1]
        res_2_unsorted.append((g.name, g_scholarship_max))

# Сортировка по максимальной стипендии
res_2 = sorted(res_2_unsorted, key=itemgetter(1), reverse=True)
print(res_2)

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

if __name__ == '__main__':
    main()

```

Результат выполнения

```
/usr/bin/python3.9 /home/levus/PycharmProjects/BKIT_2021/labs/rk1/main.py
```

```
Задание Г1
```

```
{'AK1-31': ['Абуховский'], 'AK3-52': ['Бондаренко'], 'AK1-32': ['Милевич'], 'AK3-51': ['Коновалов']}
```

```
Задание Г2
```

```
[('AK3-51', 100000), ('ИУ5-34Б', 12000), ('AK3-52', 9000), ('AK1-31', 3500), ('AK1-32', 3500), ('ФН12-32Б', 0)]
```

```
Задание Г3
```

```
[('Барабанчиков', 12000, 'PREP-12'), ('Абуховский', 3500, 'AK1-31'), ('Милевич', 3500, 'AK1-32'), ('Коновалов', 100000, 'AK3-51'), ('Бондаренко', 9000, 'AK3-52'), ('Гордеев', 0, 'ИУ5-34Б'), ('Барабанчиков', 12000, 'ИУ5-34Б'), ('Козлов', 0, 'ФН12-32Б')]
```

```
Process finished with exit code 0
```