

**Федеральное государственное бюджетное образовательное  
учреждение высшего образования «Московский государственный  
технический университет имени Н.Э.Баумана  
(национальный исследовательский университет)»**



**Факультет «Информатика и системы управления»  
Курс «Базовые компоненты интернет-технологий»**

**Рубежный контроль №1**

**Выполнил:**

**студент группы ИУ5-33Б Николай Горкунов**

**подпись: \_\_\_\_\_, дата: \_\_\_\_\_**

**Проверил:**

**преподаватель Юрий Гапанюк**

**подпись: \_\_\_\_\_, дата: \_\_\_\_\_**

**2022 г.**

## Задание.

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

Для моей группы вариант запросов В, для меня вариант предметной области 4.

- Вариант В запросов:  
«Отдел» и «Сотрудник» связаны отношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.  
«Отдел» и «Сотрудник» связаны отношением один-ко-многим. Выведите список отделов с минимальной зарплатой сотрудников в каждом отделе, отсортированный по минимальной зарплате.  
«Отдел» и «Сотрудник» связаны отношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.
- Вариант 3 предметной области:  
Компьютер – Дисплейный класс.

## Исходный код.

```
# Используется для сортировки
from operator import itemgetter

class Computer:
    """Компьютер"""
    def __init__(self, id, name, cost, clas_id):
        self.id = id
        self.name = name
        self.cost = cost
        self.clas_id = clas_id

class DispClass:
    """Дисплейный класс"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class ComputerDispClass:
    """
    'Компьютеры дисплейного класса' для реализации
    связи многие-ко-многим
    """
    def __init__(self, clas_id, comp_id):
        self.clas_id = clas_id
        self.comp_id = comp_id

# Дисплейный классы
classes = [
    DispClass(1, 'Класс информатики'),
    DispClass(2, 'Класс генной инженерии'),
    DispClass(3, 'Класс 3Д моделирования'),

    DispClass(11, 'Класс информатики (другой)'),
    DispClass(22, 'Класс генной инженерии (другой)'),
    DispClass(33, 'Класс 3Д моделирования (другой)'),
]

# Компьютеры
computers = [
    Computer(1, 'Делл_АУМ01', 125000, 1),
    Computer(2, 'Эпл_АУМ02', 135000, 2),
```

```

Computer(3, 'Асус_АУМ03', 145000, 3),
Computer(4, 'Делл_АУМ04', 135000, 3),
Computer(5, 'Асус_АУМ05', 125000, 3),
]

computers_classes = [
    ComputerDispClass(1, 1),
    ComputerDispClass(2, 2),
    ComputerDispClass(3, 3),
    ComputerDispClass(3, 4),
    ComputerDispClass(3, 5),

    ComputerDispClass(11, 1),
    ComputerDispClass(22, 2),
    ComputerDispClass(33, 3),
    ComputerDispClass(33, 4),
    ComputerDispClass(33, 5),
]

def main():
    # Соединение данных один-ко-многим
    one_to_many = [(comp.name, comp.cost, clas.name)
                    for clas in classes
                    for comp in computers
                    if comp.clas_id == clas.id]

    # Соединение данных многие-ко-многим
    many_to_many_tmp = [(clas.name, compclas.clas_id,
compclas.comp_id)
                        for clas in classes
                        for compclas in computers_classes
                        if clas.id == compclas.clas_id]

    many_to_many = [(comp.name, comp.cost, clas_name)
                    for clas_name, clas_id, comp_id in many_to_many_tmp
                    for comp in computers if comp.id == comp_id]

    print('Задание B1')
    res_11 = list(filter(lambda x: x[0].find('A') == 0,
one_to_many))
    print(res_11)

```

```

print('\nЗадание В2')
res_12_unsorted = []
# Перебираем все дисплейные классы
for clas in classes:
    # Список компьютеров дисплейного класса
    clas_comps = list(filter(lambda i: i[2] == clas.name,
one_to_many))
    # Если дисплейный класс не пустой
    if len(clas_comps) > 0:
        # Стоимости компьютеров дисплейного класса
        clas_costs = list(map(lambda x: x[1], clas_comps))
        # Минимальная стоимость среди компьютеров
дисплейного класса
        clas_costs_min = min(clas_costs)
        res_12_unsorted.append((clas.name, clas_costs_min))

# Сортировка по минимальной стоимости
res_12 = sorted(res_12_unsorted, key = itemgetter(1))
print(res_12)

print('\nЗадание В3')
# Сортировка по имени компьютера
res_13 = sorted(many_to_many, key = itemgetter(0))
print(res_13)

if __name__ == '__main__':
    main()

```

## Пример выполнения.

```

nор@nорc:~/Projects/bmstu_3sem/BKIT_2022$ python3.8 main.py

```

Задание В1

```
[('Асус_АУМ03', 145000, 'Класс 3Д моделирования'), ('Асус_АУМ05', 125000, 'Класс 3Д моделирования')]
```

Задание В2

```
[('Класс информатики', 125000), ('Класс 3Д моделирования', 125000), ('Класс генной инженерии', 135000)]
```

Задание В3

```
[('Асус_АУМ03', 145000, 'Класс 3Д моделирования'), ('Асус_АУМ03', 145000, 'Класс 3Д моделирования (другой)'), ('Асус_АУМ05', 125000, 'Класс 3Д моделирования'), ('Асус_АУМ05', 125000, 'Класс 3Д моделирования (другой)'), ('Делл_АУМ01', 125000, 'Класс информатики'), ('Делл_АУМ01', 125000, 'Класс информатики (другой)'), ('Делл_АУМ04', 135000, 'Класс 3Д моделирования'), ('Делл_АУМ04', 135000, 'Класс 3Д моделирования (другой)'), ('Эпл_АУМ02', 135000, 'Класс генной инженерии'), ('Эпл_АУМ02', 135000, 'Класс генной инженерии (другой)')]
```

```

nор@nорc:~/Projects/bmstu_3sem/BKIT_2022$ █

```