Министерство науки и высшего образования Российской ФедерацииФедеральное государственное бюджетное образовательное учреждение высшего образования«Московский государственный технический университетимени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Рубежный контроль №1

по дисциплине «Базовые компоненты интернет-технологий»

Выполнила:
студентка группы ИУ5-33Б
Пересыпкина М.А.

Проверил:
Гапанюк Ю.Е.

2021 г.

Постановка задания:

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.

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».

Вариант В.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с минимальной зарплатой сотрудников в каждом отделе, отсортированный по минимальной зарплате.
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.

Предметная область:

11	Программа	Компьютер

Текст программы:

```
# используется для сортировки
from operator import itemgetter


class Prog:
    """Программа"""
```

```
    def __init__(self, id, nameProg, memory, comp_id):
        self.id = id
        self.nameProg = nameProg
        self.memory = memory
        self.comp_id = comp_id


class Comp:
    """Компьютер"""

    def __init__(self, id, name):
        self.id = id
        self.name = name


class ProgComp:
    """
    'Программы компьютера' для реализации
    связи многие-ко-многим
    """

    def __init__(self, comp_id, prog_id):
        self.comp_id = comp_id
        self.prog_id = prog_id


# Компьютеры
comps = [
    Comp(1, 'Asus'),
    Comp(2, 'Acer'),
    Comp(3, 'Xiaomi'),
    Comp(4, 'HP'),
    Comp(5, 'Apple'),
    Comp(6, 'Lenovo'),
]

# Программы
progs = [
    Prog(1, 'Adobe Premiere Pro', 1150, 1),
    Prog(2, 'Sony Vegas Pro', 970, 2),
    Prog(3, 'Discord', 200, 2),
    Prog(4, 'Figma', 450, 3),
    Prog(5, 'PyCharm Community', 580, 4),
]
# Программы и компьютеры для связи многие-ко-многим
progs_comps = [
    ProgComp(1, 1),
    ProgComp(2, 2),
    ProgComp(2, 3),
    ProgComp(3, 4),
    ProgComp(4, 5),

    ProgComp(5, 1),
    ProgComp(5, 2),
    ProgComp(5, 3),
    ProgComp(6, 4),
    ProgComp(6, 5),
]


def main():
    """Основная функция"""
```

```
    # Соединение данных один-ко-многим
    one_to_many = [(p.nameProg, p.memory, c.name)
                   for c in comps
                   for p in progs
                   if p.comp_id == c.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(c.name, pc.comp_id, pc.prog_id)
                         for c in comps
                         for pc in progs_comps
                         if c.id == pc.comp_id]

    many_to_many = [(p.nameProg, p.memory, comp_name)
                    for comp_name, comp_id, prog_id in many_to_many_temp
                    for p in progs if p.id == prog_id]

    print('Задание B1')
    task1 = []
    for nameProg, memory, name in one_to_many:
        if nameProg[0] == "A":
            task1.append((nameProg, name))
    print(task1)

    print('\nЗадание B2')
    task2_uns = []
    for c in comps:
        # все программы компьютера
        p_comp = list(filter(lambda i: i[2] == c.name, one_to_many))
        if len(p_comp) > 0:
            c_memory = [memory for _, memory, _ in p_comp]
            c_minMemory = min(c_memory)
            task2_uns.append((c.name, c_minMemory))
    task2 = sorted(task2_uns, key=itemgetter(1))
    print(task2)

    print('\nЗадание B3')
    task3_uns = []
    for nameProg, memory, name in many_to_many:
        task3_uns.append((nameProg, name))

    task3 = list(sorted(task3_uns, key=itemgetter(0)))
    print(task3)


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

Результаты выполнения программы:

Задание B1

[('Adobe Premiere Pro', 'Asus')]

Задание B2

[('Acer', 200), ('Xiaomi', 450), ('HP', 580), ('Asus', 1150)]

Задание В3

[('Adobe Premiere Pro', 'Asus'), ('Adobe Premiere Pro', 'Apple'), ('Discord', 'Acer'), ('Discord', 'Apple'), ('Figma', 'Xiaomi'), ('Figma', 'Lenovo'), ('PyCharm Community', 'HP'), ('PyCharm Community', 'Lenovo'), ('Sony Vegas Pro', 'Acer'), ('Sony Vegas Pro', 'Apple')]