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

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Разработка интернет-приложений»

Отчет по рубежному контролю №1
Вариант Г9

Выполнил:
студент группы ИУ5-54Б
Киреев А.А.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Ю.Е. Гапанюк.
Подпись и дата:

Москва, 2021 г.

Условия рубежного контроля №1 по курсу РИП

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.

Задание по варианту.

9	Операционная система	Компьютер

Текст программы.

```
# используется для сортировки
from operator import itemgetter


class Os:
    """Операционная система"""
    def __init__(self, id, name, start, comp_id):
        self.id = id
        self.name = name
        self.start = start
        self.comp_id = comp_id

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


class OsComp:
    """
    'Операционные системы в компьютере' для реализации
    связи многие-ко-многим
    """
    def __init__(self, os_id, comp_id):
        self.os_id = os_id
        self.comp_id = comp_id


# Компьютеры
Comps = [
    Computer(1, 'Настольный ПК'),
    Computer(2, 'Ноутбук'),
    Computer(3, 'Игровой ПК'),
]

# Операционные системы
OSs = [
    Os(1, 'Windows', 1239, 3),
    Os(2, 'Mint', 267, 1),
    Os(3, 'MacOS', 873, 2),
    Os(4, 'Ubuntu', 365, 3),
    Os(5, 'Debian', 89, 1),
    Os(6, 'Chrome OS', 99, 3),
    Os(7, 'Fedora', 17, 1),
]

oss_comps = [
    OsComp(1, 3),
    OsComp(2, 1),
    OsComp(3, 2),
    OsComp(4, 3),
    OsComp(5, 1),
    OsComp(6, 3),
    OsComp(7, 1),
]
```

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

    # Соединение данных один-ко-многим
    one_to_many = [(os_temp.name, os_temp.start, comp_temp.name)
                   for comp_temp in Comps
                   for os_temp in OSs
                   if os_temp.comp_id == comp_temp.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(comp_temp.name, os_temp.comp_id, os_temp.os_id)
                         for comp_temp in Comps
                         for os_temp in oss_comps
                         if comp_temp.id == os_temp.comp_id]

    many_to_many = [(os_temp.name, os_temp.start, comp_name)
                    for comp_name, comp_id, os_id in many_to_many_temp
                    for os_temp in OSs if os_temp.id == os_id]

    print('Задание Г1')
    res_11 = [(os_temp.name, comp_temp.name)
              for comp_temp in Comps
              for os_temp in OSs
              if (os_temp.comp_id == comp_temp.id)&(comp_temp.name[:13] ==
"Настольный ПК")]
    print(res_11)



    print('\nЗадание Г2')
    res_12_unsorted = []
    # Перебираем все компьютеры
    for comp_temp in Comps:
        # Список ОС в компьютерах
        c_oses = list(filter(lambda i: i[2] == comp_temp.name, one_to_many))
        # Если компьютер не пуст
        if len(c_oses) > 0:
            # Сколько раз запускали ОС
            c_starts = [start for  _, start, _ in c_oses]
            # Максимальное число запусков
            c_starts_max = max(c_starts)
            res_12_unsorted.append((comp_temp.name, c_starts_max))

    # Сортировка по максимальному числу запусков
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    print(res_12)

    print('\nЗадание Г3')
    res_13 = sorted(many_to_many, key=itemgetter(2))
    print(res_13)

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

Результаты работы программы.