Московский Государственный Технический Университет им. Н.Э.Баумана
(МГТУ им Баумана)

Факультет: Информатика и системы управления (ИУ)
Кафедра: ИУ-5

ОТЧЕТ

По рубежному контролю №1
По дисциплине «Разработка интернет-приложений»
Объектно-ориентированные возможности языка Python.

Выполнил:
Дума Эмилия ИУ5-52Б

Проверил:
Гапанюк Ю. Е.

(подпись) (дата) (подпись) (дата)

Москва, 2020

_____ _____

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).

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

Результатом рубежного контроля является документ в формате PDF, который содержит текст программы и результаты ее выполнения.

Вариант Б.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с количеством сотрудников в каждом отделе, отсортированный по количеству сотрудников.
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.

№ варианта	Класс 1	Класс 2
10	Браузер	Компьютер

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


class Brz:
    """Браузер"""

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


class Cmp:
    """Компьютер"""

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



class BrzCmp:
    def __init__(self, brz_id, cmp_id):
        self.brz_id = brz_id
        self.cmp_id = cmp_id


# Отделы
brzs = [
    Brz(1, 'bbrz1', 51, 1),
    Brz(2, 'brz2', 52, 2),
    Brz(3, 'brz3', 53, 3),

    Brz(4, 'bbrz11', 31, 2),
    Brz(5, 'brz22', 32, 4),
    Brz(6, 'bbrz33', 33, 5),
]

# Сотрудники
cmps = [
    Cmp(1, 'ccmp1'),
    Cmp(2, 'cmp2q'),
    Cmp(3, 'ccmp3'),
    Cmp(4, 'cmp4'),
    Cmp(5, 'ccmp5q'),
]

cmps_brzs = {
    BrzCmp(1, 1),
    BrzCmp(2, 2),
    BrzCmp(3, 3),
    BrzCmp(3, 4),
    BrzCmp(3, 5),

    BrzCmp(4, 1),
    BrzCmp(4, 2),
    BrzCmp(5, 3),
    BrzCmp(5, 4),
    BrzCmp(5, 5),
}
```

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

    # Соединение данных один-ко-многим
    one_to_many = [(b.name, b.size, c.name)
                   for b in brzs
                   for c in cmps
                   if b.cmp_id == c.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(c.name, cd.brz_id, cd.cmp_id)
                         for c in cmps
                         for cd in cmps_brzs
                         if c.id == cd.cmp_id]

    many_to_many = [(b.name, b.size, b_name)
                    for b_name, b_id, cmp_id in many_to_many_temp
                    for b in brzs if b.id == cmp_id]

    print(many_to_many)
    print('Задание Б1')
    res_11 = sorted(one_to_many, key=itemgetter(0))
    print(res_11)

    print('\nЗадание Б2')
    res_12_unsorted = []

    for d in cmps:
        d_emps = list(filter(lambda i: i[2] == d.name, one_to_many))
        if len(d_emps) > 0:

            res_12_unsorted.append((d.name, len(d_emps)))

    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    print(res_12)

    print('\nЗадание Б3')
    res_13 = {}

    for c in cmps:
        if c.name[-1] == 'q':
            d_emps = list(filter(lambda i: i[2] == c.name, many_to_many))

            d_emps_names = [x for x, _, _ in d_emps]

            res_13[c.name] = d_emps_names

    print(res_13)


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

```
Задание Б1
[('bbrz1', 51, 'ccmp1'), ('bbrz11', 31, 'cmp2q'), ('bbrz33', 33, 'ccmp5q'), ('brz2', 52, 'cmp2q'), ('brz22', 32, 'cmp4'), ('brz3', 53, 'ccmp3')]

Задание Б2
[('cmp2q', 2), ('ccmp1', 1), ('ccmp3', 1), ('cmp4', 1), ('ccmp5q', 1)]

Задание Б3
{'cmp2q': ['brz2'], 'ccmp5q': ['brz22']}

Process finished with exit code 0
```