Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени 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»

Выполнил:

студент группы ИУ5-35Б

Терентьев Д. А.

2021 г.

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).

Вариант Д.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов»,
и названия их отделов.

2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате (отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений).

3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «А», и список работающих в них сотрудников.

Вариант предметной области 15.

Файл-Каталог

В соответствии с предметной областью, задание было немного изменено:

3. «Каталог» и «Файл» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «П», и список работающих в них сотрудников.

```
# используется для сортировки
from operator import itemgetter


class File:
    """Файл"""

    def __init__(self, id, namef, size, cat_id):
        self.id = id
        self.namef = namef
        self.size = size
        self.cat_id = cat_id


class Cat:
    """Каталог"""

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


class CatFile:
    """
    'Файлы каталога' для реализации
    связи многие-ко-многим
    """

    def __init__(self, cat_id, file_id):
        self.cat_id = cat_id
        self.file_id = file_id


catalogs = [
    Cat(1, 'Рабочий стол'),
    Cat(2, 'Панель управления'),
    Cat(3, 'Папка1'),
    # для связи многие-ко-многим:
    Cat(11, 'Папка2'),
    Cat(22, 'Работы'),
    Cat(33, 'ДЗ'),
]

files = [
    File(1, 'photo.pdf', 3, 1),
    File(2, 'image.pdf', 2, 2),
    File(3, 'image.jpg', 5, 2),
    File(4, 'image0.pdf', 6.1, 3),
    File(5, 'lib1.jpg', 10, 3),


]

files_cats = [
    CatFile(3, 4),
```

```
    CatFile(3, 5),
    CatFile(2, 3),
    CatFile(2, 2),
    CatFile(1, 1),


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

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

    # Соединение данных один-ко-многим
    one_to_many = [(f.namef, f.size, c.namec)
                   for c in catalogs
                   for f in files
                   if f.cat_id == c.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(c.namec, fc.cat_id, fc.file_id)

                         for c in catalogs
                             for fc in files_cats
                             if c.id == fc.cat_id]

    many_to_many = [(f.namef, f.size, cat_name)
                    for cat_name, cat_id, file_id in many_to_many_temp
                        for f in files if b.id == file_id]

    print('Задание D1')
    res1 = list(filter(lambda x: x[0].endswith(".jpg"), one_to_many))
    print(res1)

    print('\nЗадание D2')
    res2unsorted = []
    # Перебираем все каталоги
    for c in catalogs:
        # Список файлов в каталоге
        filess = list(filter(lambda i: i[2] == c.namec, one_to_many))
            # Если в каталоге есть файл
        if len(filess) > 0:
                    # Все размеры файлов в каталоге
                    allSizes = [size for _, size, _ in filess]
                    # Средний размер файла в каталоге
                    averageSizes = round(sum(allSizes) / len(allSizes), 2)
                    res2unsorted.append((c.namec, averageSizes))

    # Сортировка по среднему размеру
    res2 = sorted(res2unsorted, key=itemgetter(1), reverse=True)
    print(res2)
    print('\nЗадание D3')
```

```
    res3 = {}
    for c in catalogs:
        if c.namec.startswith("П"):
                # Список файлов в каталоге
            filess = list(filter(lambda i: i[2] == c.namec, many_to_many))
                # Только имя файла
            filesNames = [x for x, _, _ in filess]
                # Добавляем результат в словарь
                # ключ - каталог, значение - список названий файлов
            res3[c.namec] = filesNames

    print(res3)

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

Результаты:

Задание D1
[('image.jpg', 5, 'Панель управления'), ('lib1.jpg', 10, 'Папка1')]

Задание D2
[('Рабочий стол', 3.0), ('Панель управления', 3.5), ('Папка1', 8.05)]

Задание D3
{'Панель управления': ['image.pdf', 'image.jpg'], 'Папка1': ['image0.pdf', 'lib1.jpg'], 'Папка2': ['photo.pdf']}