*Федеральное государственное бюджетное образовательное учреждение* *высшего профессионального образования*

|  |  |
| --- | --- |
|  | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления (ИУ5)

**Отчет**

**по рубежному контролю №1**

**Дисциплина: Разработка Интернет-Приложений**

Студент гр. ИУ5-54Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Самойлов А.М.

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Гапанюк Ю.Е.

(Подпись, дата) (И.О. Фамилия)

Москва, 2020

**Задание**

**Вариант Г**Вариант предметной области

|  |  |  |
| --- | --- | --- |
| 18 | Музыкальное произведение | Оркестр |

1. «Оркестр» и «Музыкальное произведение» связаны соотношением один-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «А», и список работающих в них сотрудников.
2. «Оркестр» и «Музыкальное произведение» связаны соотношением один-ко-многим. Выведите список отделов с максимальной зарплатой сотрудников в каждом отделе, отсортированный по максимальной зарплате.
3. «Оркестр» и «Музыкальное произведение» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по отделам, сортировка по сотрудникам произвольная.

**Текст программы**

from operator import itemgetter

# Самойлов Алексей Михайлович, группа ИУ5-54Б, Вариант Г-18

tasks = [

    "— Список оркестров на букву 'А' и их композиций",                          # Г1

    "— Список оркестров с максимальной длиной композиции, сортировка по длине", # Г2

    "— Список всех связанных композиций и оркестров, сортировка по оркестрам"   # Г3

]

class composition:

    #Музыкальная композиция

    def \_\_init\_\_(self, id, title, author, length, orchestra\_id):

        self.id = id

        self.title = title

        self.author = author

        self.length = length

        self.orchestra\_id = orchestra\_id

class orchestra:

    #Оркестр

    def \_\_init\_\_(self, id, name):

        self.id = id

        self.name = name

class CompOrch:

    """

    'Музыкальное произведение оркестров' для реализации

    связи многие-ко-многим

    """

    def \_\_init\_\_(self, orchestra\_id, composition\_id):

        self.orchestra\_id = orchestra\_id

        self.composition\_id = composition\_id

# Оркестры

orchestras = [

    orchestra(1, 'Большой симфонический оркестр'),

    orchestra(2, 'Симфонический оркестр Большого театра'),

    orchestra(3, 'Академический симфонический оркестр'),

]

# Композиции

compositions = [

    composition(1, 'К Элизе', 'Людвиг ван Бетховен', 170, 1),

    composition(2, 'Турецкое рондо', 'Вольфганг Амадей Моцарт', 258, 2),

    composition(3, 'Аве Мария', 'Франц Шуберт', 324, 3),

    composition(4, 'Утро', 'Эдвард Григ', 563, 1),

    composition(5, 'Лунный свет', 'Клод Дебюсси', 434, 2),

    composition(6, 'Лебедь', 'Камиль Сен-Санс', 453, 3),

]

# Оркестры - музыкальные произведения

orchestra\_composition = [

    CompOrch(1, 1),

    CompOrch(2, 2),

    CompOrch(3, 3),

    CompOrch(1, 4),

    CompOrch(2, 5),

    CompOrch(3, 6)

]

def main():

    """Основная функция"""

    #один ко многим

    one\_to\_many = [(b.title, b.length, s.name)

                   for s in orchestras

                   for b in compositions

                   if b.orchestra\_id == s.id]

    #многие ко многим

    many\_to\_many\_temp = [(s.name, bs.orchestra\_id, bs.composition\_id)

                         for s in orchestras

                         for bs in orchestra\_composition

                         if s.id == bs.orchestra\_id]

    many\_to\_many = [(b.title, orchestra\_name)

                    for orchestra\_name, orchestra\_id, composition\_id in many\_to\_many\_temp

                    for b in compositions if b.id == composition\_id]

    print('— Задание Г1'  + "\n" + tasks[0])

    res\_11 = list(filter(lambda x: x[2].startswith('А'), one\_to\_many))

    for i in res\_11:

        print(i)

    print('\n\n— Задание Г2' + "\n" + tasks[1])

    res\_12\_unsorted = []

    for s in orchestras:

        s\_compositions = list(filter(lambda i: i[2] == s.name, one\_to\_many))

        if len(s\_compositions) > 0:

            s\_length = [length for \_, length, \_ in s\_compositions]

            s\_length\_max = max(s\_length)

            res\_12\_unsorted.append((s.name, s\_length\_max))

    res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)

    print(res\_12)

    print('\n\n— Задание Г3'  + "\n" + tasks[2])

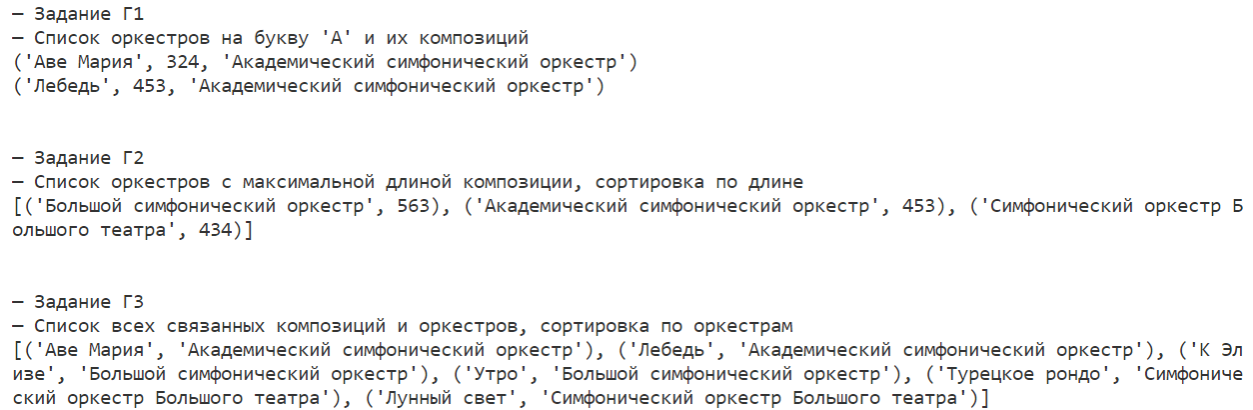
    res\_13 = sorted(many\_to\_many, key=itemgetter(1))

    print(res\_13)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**Результат работы программы**

****