*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. ИУ5-53Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Перлин Л.В.

(Подпись, дата) (И.О. Фамилия)

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

(Подпись, дата) (И.О. Фамилия)

Москва, 2020

**Задание**

**Вариант В**Вариант предметной области

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

1. «Оркестр» и «Музыкальное произведение» связаны соотношением один-ко-многим.

Вывод списка всех музыкальных произведений, у которых название начинается с буквы «Л», и названия Оркестров, которые его исполняют.

1. «Оркестр» и «Музыкальное произведение» связаны соотношением один-ко-многим.

Вывод списка оркестров с минимальной продолжительностью композиции, отсортированный по минимальной продолжительности композиции (в секунднах).

1. «Оркестр» и «Музыкальное произведение» связаны соотношением многие-ко-многим.

Вывод списка всех связанных исполняемых композиций и оркестров, отсортированный по композициям, сортировка по композициям произвольная.

**Текст программы**

from operator import itemgetter

class compositionorchestra:

    #Музыкальные произведения оркестров (многие-ко-многим)

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

        self.orchestra\_id = orchestra\_id

        self.composition\_id = composition\_id

class composition:

    #Музыкальная композиция

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

        self.id = id

        self.title = title

        self.autor = autor

        self.length = length

        self.orchestra\_id = orchestra\_id

class orchestra:

    #Оркестр

    def \_\_init\_\_(self, id, name):

        self.id = id

        self.name = name

# Оркестры

orchestras = [

    orchestra(1, 'Royal Concertgebouw Orchestra'),

    orchestra(2, 'Berlin Philharmonic Orchestra'),

    orchestra(3, 'Vienna Philharmonic Orchestra'),

]

# 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 - композиции

orchestra\_composition = [

    compositionorchestra(1, 1),

    compositionorchestra(2, 1),

    compositionorchestra(3, 2),

    compositionorchestra(2, 4),

    compositionorchestra(1, 5),

    compositionorchestra(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 А1')

    res\_11 = list(filter(lambda x: x[0].startswith('Л'), one\_to\_many))

    print(res\_11)

    print('\nЗадание А2')

    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\_min = min(s\_length)

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

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

    print(res\_12)

    print('\nЗадание А3')

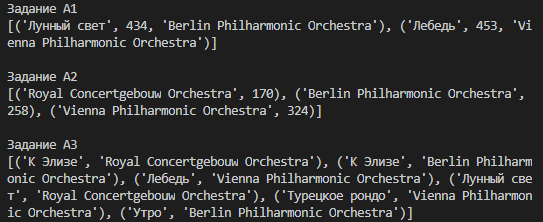
    res\_13 = sorted(many\_to\_many, key=itemgetter(0))

    print(res\_13)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**Результат работы программы**

****