Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий**»**

Рубежный контроль №1

Выполнил:		Проверил:
студент группы ИУ5-32Б		преподаватель каф. ИУ5
Ткаченко В. Л.		Гапонюк Ю. Е.
Подпись и дата		Подпись и дата

Москва, 2021 г.

Задание, Вариант Б:

1. «Музыкальное произведение» и «Оркестр» связаны соотношением один-ко-многим. Выведите список всех связанных оркестров и музыкальных произведений, отсортированный по оркестрам, сортировка по музыкальным произведениям произвольная.
2. «Музыкальное произведение» и «Оркестр» связаны соотношением один-ко-многим. Выведите список мызукальных произведений с количеством оркестров, которые играют это произведение, отсортированный по количеству оркестров.
3. «Музыкальное произведение» и «Оркестр» связаны соотношением многие-ко-многим. Выведите список всех оркестров, в которых играют 10 человек, и названия их музыкальных произведений.

Листинг программы:

```
from operator import itemgetter
from pprint import pprint


class Table():
    id: int = 0

    def __init__(self):
        self.incId()

    @classmethod
    def getId(cls):
        return cls.id

    @classmethod
    def incId(cls):
        cls.id += 1


class Link:
    ONE_TO_MANY: int = 1
    MANY_TO_ONE: int = 2
    MANY_TO_MANY: int = 3


class Relationship:
```

```
    def __init__(self, tableFirst: Table, tableSecond: Table,
relation, fk=''):
        self.tableFirst = tableFirst
        self.tableSecond = tableSecond
        self.relation = relation
        self.fk = fk
        self.many_to_many = []

    def link(self, objectFirst, objectSecond):
        if self.relation == Link.ONE_TO_MANY:
            assert self.fk, 'Set foreign key'
            setattr(objectSecond, self.fk, objectFirst.id)

        elif self.relation == Link.MANY_TO_MANY:
            self.many_to_many.append([objectFirst.id,
objectSecond.id])

    def iter(self, tableFirst, tableSecond):
        result = []

        if self.relation == Link.ONE_TO_MANY:
            for recordFirst in tableFirst:
                for recordSecond in tableSecond:
                    if getattr(recordSecond, self.fk) ==
recordFirst.id:
                        result.append([recordFirst, recordSecond])

        elif self.relation == Link.MANY_TO_MANY:
            for idFirst, idSecond in self.many_to_many:
                for recordFirst in tableFirst:
                    for recordSecond in tableSecond:
                        if idFirst == recordFirst.id and idSecond
== recordSecond.id:
                            result.append([recordFirst,
recordSecond])

        return result


class Music(Table):
    """Музыкальное произведение"""
    def __init__(self, name):
```

```
        super().__init__()
        self.id = self.getId()
        self.name = name

    def __iter__(self):
        return iter((self.name,))


class Orchestra(Table):
    """Оркестр"""
    fk_music = 'music_id'

    def __init__(self, name, amount):
        super().__init__()
        self.id = self.getId()
        self.name = name
        self.amount = amount
        self.music_id = 0

    def __iter__(self):
        return iter((self.name, self.amount))


musics = [
    imagine := Music('Imagine'),
    stone := Music('Like a Rolling Stone'),
    respect := Music('Respect'),
    vibrations := Music('Good Vibrations'),
    jude := Music('Hey Jude'),
    generation := Music('My Generation'),
]

orcs = [
    miller := Orchestra('Оркестр Гленна Миллера', 2),
    moria := Orchestra('Оркестр под управлением Поля Мориа',
3),
    czech := Orchestra('Чешский филармонический оркестр', 10),
    cleveland := Orchestra('Кливлендский оркестр', 15),
    philadelphia := Orchestra('Филадельфийский оркестр', 10),
]

rel_otm = Relationship(Music, Orchestra, Link.ONE_TO_MANY,
```

```
fk=Orchestra.fk_music)
rel_otm.link(imagine, miller)
rel_otm.link(stone, moria)
rel_otm.link(respect, czech)
rel_otm.link(imagine, cleveland)
rel_otm.link(imagine, philadelphia)

rel_mtm = Relationship(Music, Orchestra, Link.MANY_TO_MANY)
rel_mtm.link(imagine, miller)
rel_mtm.link(stone, moria)
rel_mtm.link(respect, czech)
rel_mtm.link(imagine, cleveland)
rel_mtm.link(imagine, philadelphia)
rel_mtm.link(stone, philadelphia)
rel_mtm.link(jude, cleveland)

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

    print('Задание A1')

    arr = [(o_name, m_name) for (m_name,), (o_name, _) in
rel_otm.iter(musics, orcs)]
    sorted_arr = sorted(arr, key=itemgetter(0))

    pprint(sorted_arr)


    print('\nЗадание A2')

    m_names = [m_name for (m_name,), (o_name, _) in
rel_otm.iter(musics, orcs)]
    count = {}

    for m_name in m_names:
        count[m_name] = count[m_name] + 1 if count.get(m_name)
else 1

    sorted_count = sorted(count.items(), key=itemgetter(1))
    [print(m_name, o_count) for m_name, o_count in sorted_count]
```

```
    print('\nЗадание A3')
    arr = [(m_name, o_name, o_amount) for (m_name,), (o_name,
o_amount) in rel_mtm.iter(musics, orcs)]

    for (m_name, o_name, o_amount) in arr:
        if o_amount == 10:
            print(m_name, o_name)


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

Пример работы:

```
→  rk1 git:(main) ✗ python main.py
Задание A1
[('Кливлендский оркестр', 'Imagine'),
 ('Оркестр Гленна Миллера', 'Imagine'),
 ('Оркестр под управлением Поля Мориа', 'Like a Rolling Stone'),
 ('Филадельфийский оркестр', 'Imagine'),
 ('Чешский филармонический оркестр', 'Respect')]

Задание A2
Like a Rolling Stone 1
Respect 1
Imagine 3

Задание A3
Respect Чешский филармонический оркестр
Imagine Филадельфийский оркестр
Like a Rolling Stone Филадельфийский оркестр
→  rk1 git:(main) ✗
```