

# РК1 Воробьев Егор ИУ5-32Б Вариант № 30/А

## Листинг кода:

```
"""
Классы данных:
1) Faculty (Факультет)
2) University (Университет)
3) FacultyUniversity - для связи многие-ко-многим

Связи:
1) Один-ко-многим: Один факультет может иметь много университетов
2) Многие-ко-многим: Университет может относиться к нескольким факультетам

Запросы:
A1: Список всех связанных университетов и факультетов, отсортированный по факультетам
A2: Список факультетов с суммарным количеством студентов, отсортированный по убыванию
A3: Словарь факультетов (содержащих "факультет" в названии) и их университетов
"""

from operator import itemgetter

class Faculty:
    """Факультет"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class University:
    """Университет"""
    def __init__(self, id, name, students_count, faculty_id):
        self.id = id
        self.name = name
        self.students_count = students_count # Количественный признак
        self.faculty_id = faculty_id # FK Для связи один-ко-многим

class FacultyUniversity:
    """
    Факультеты университета для реализации
    связи многие-ко-многим
    """
    def __init__(self, faculty_id, university_id):
        self.faculty_id = faculty_id
        self.university_id = university_id

# Факультеты
faculties = [
    Faculty(1, 'Факультет информатики'),
    Faculty(2, 'Экономический факультет'),
    Faculty(3, 'Юридический факультет'),
    Faculty(4, 'Факультет иностранных языков'),
    Faculty(5, 'Медицинский факультет'),
]

# Университеты
universities = [
    University(1, 'МГУ', 1500, 1),
    University(2, 'МГИМО', 1200, 2),
```

```

University(3, 'МФТИ', 800, 1),
University(4, 'ВШЭ', 2000, 2),
University(5, 'МГЮА', 900, 3),
University(6, 'МГЛУ', 700, 4),
University(7, 'ПМГМУ', 1100, 5),
University(8, 'МГТУ', 600, 1),
]

```

# Факультеты и университеты (многие-ко-многим)

```

faculties_universities = [
    FacultyUniversity(1, 1), # Факультет информатики -> МГУ
    FacultyUniversity(1, 3), # Факультет информатики -> МФТИ
    FacultyUniversity(1, 8), # Факультет информатики -> МГТУ
    FacultyUniversity(2, 2), # Экономический факультет -> МГИМО
    FacultyUniversity(2, 4), # Экономический факультет -> ВШЭ
    FacultyUniversity(3, 5), # Юридический факультет -> МГЮА
    FacultyUniversity(4, 6), # Факультет иностранных языков -> МГЛУ
    FacultyUniversity(5, 7), # Медицинский факультет -> ПМГМУ

```

# Добавляем связи многие-ко-многим

```

FacultyUniversity(1, 2), # Факультет информатики -> МГИМО
FacultyUniversity(2, 1), # Экономический факультет -> МГУ

```

def main():

"""Основная функция"""

# --- Подготовка данных ---

# Соединение данных один-ко-многим (Университет -> Факультет)

```

one_to_many = [(u.name, u.students_count, f.name)
                for f in faculties
                for u in universities
                if u.faculty_id == f.id]

```

# Соединение данных многие-ко-многим

# 1. Промежуточное соединение

```

many_to_many_temp = [(f.name, fu.faculty_id, fu.university_id)
                      for f in faculties
                      for fu in faculties_universities
                      if f.id == fu.faculty_id]

```

# 2. Основное соединение

```

many_to_many = [(u.name, u.students_count, fac_name)
                 for fac_name, fac_id, uni_id in many_to_many_temp
                 for u in universities if u.id == uni_id]

```

# --- Выполнение запросов (Вариант А) ---

print('Задание А1')

# Выводим список всех связанных университетов и факультетов,

# отсортированный по количеству студентов

```
res_1 = sorted(one_to_many, key=itemgetter(1))
```

```
print(res_1)
```

print('\nЗадание А2')

# Выводим список факультетов с суммарным количеством студентов в каждом факультете,

# отсортированный по суммарному количеству студентов

```

res_2_unsorted = []
# Перебираем все факультеты
for f in faculties:
    # Список университетов в текущем факультете (из one_to_many)
    f_universities = list(filter(lambda i: i[2] == f.name, one_to_many))
    # Если факультет не пустой
    if len(f_universities) > 0:
        # Количество студентов в университетах факультета
        f_students = [students for _, students, _ in f_universities]
        # Суммарное количество студентов
        f_students_sum = sum(f_students)
        res_2_unsorted.append((f.name, f_students_sum))

# Сортировка по суммарному количеству студентов (по убыванию)
res_2 = sorted(res_2_unsorted, key=itemgetter(1), reverse=True)
print(res_2)

print("\nЗадание А3")
# Выводим список всех факультетов, у которых в названии присутствует
# слово «факультет», и список их университетов (из many-to-many)
res_3 = {}
# Перебираем все факультеты
for f in faculties:
    # Ищем слово "факультет" в названии (без учета регистра)
    if 'факультет' in f.name.lower():
        # Список университетов в этом факультете (из many_to_many)
        f_universities = list(filter(lambda i: i[2] == f.name, many_to_many))
        # Только названия университетов
        f_universities_names = [name for name, _, _ in f_universities]
        # Добавляем результат в словарь
        res_3[f.name] = f_universities_names
print(res_3)

if __name__ == '__main__':
    main()

```

## Вывод программы:

### Задание А1

[('МГТУ', 600, 'Факультет информатики'), ('МГЛУ', 700, 'Факультет иностранных языков'), ('МФТИ', 800, 'Факультет информатики'), ('МГЮА', 900, 'Юридический факультет'), ('ПМГМУ', 1100, 'Медицинский факультет'), ('МГИМО', 1200, 'Экономический факультет'), ('МГУ', 1500, 'Факультет информатики'), ('ВШЭ', 2000, 'Экономический факультет')]

### Задание А2

[('Экономический факультет', 3200), ('Факультет информатики', 2900), ('Медицинский факультет', 1100), ('Юридический факультет', 900), ('Факультет иностранных языков', 700)]

### Задание А3

{ 'Факультет информатики': ['МГУ', 'МФТИ', 'МГТУ', 'МГИМО'], 'Экономический факультет': ['МГИМО', 'ВШЭ', 'МГУ'], 'Юридический факультет': ['МГЮА'], 'Факультет иностранных языков': ['МГЛУ'], 'Медицинский факультет': ['ПМГМУ'] }