1. Класс «Файл», содержащий поля:
   - ID файла;
   - Имя файла;
   - Размер файла
   - ID записи о каталоге (для реализации связи один-ко-многим)
2. Класс «Каталог с файлами», содержащий поля:
   - ID каталога;
   - Имя каталога.
3. (Для реализации связи многие-ко-многим) Класс «Каталог файлов», содержащий поля:
   - ID записи о файле;
   - ID записи о каталоге.

Листинг программы:

```
# используется для сортировки
from operator import itemgetter

class File:
    """Файл"""
    def __init__(self, id, name, size, catalog_id):
        self.id = id
        self.name = name
        self.size = size
        self.catalog_id = catalog_id

class Catalog:
    """Каталог"""
    def __init__(self, id, catalog_name):
        self.id = id
        self.catalog_name = catalog_name

class Files_of_Catalogs:
    """
    'Каталог с файлами' для реализации
    связи многие-ко-многим
    """
    def __init__(self, file_id, catalog_id):
        self.file_id = file_id
        self.catalog_id = catalog_id

files = [
    File(1, 'txtA', 20, 1),
    File(2, 'txtB', 30, 1),
    File(3, 'program1', 50, 2),
```

```
    File(4, 'program2', 55, 2),
    File(5, 'pic1', 100, 3),
    File(6, 'pic2', 110, 3)
]

catalogs = [
    Catalog(1, 'texts'),
    Catalog(2, 'programs'),
    Catalog(3, 'pics'),
    Catalog(4, 'pics2')
]

files_catalogs = [
    Files_of_Catalogs(1,1),
    Files_of_Catalogs(2,1),
    Files_of_Catalogs(3,2),
    Files_of_Catalogs(4,2),
    Files_of_Catalogs(5,3),
    Files_of_Catalogs(6,3),
    Files_of_Catalogs(5,4),
    Files_of_Catalogs(6,4)
    ]

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

    # Соединение данных один-ко-многим
    one_to_many = [(f.name, f.size, cat.catalog_name)
        for cat in catalogs
        for f in files
        if f.catalog_id==cat.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(cat.catalog_name,
fcat.catalog_id, fcat.file_id)
        for cat in catalogs
        for fcat in files_catalogs
        if cat.id==fcat.catalog_id]

    many_to_many = [(f.name, f.size, catalog_name)
        for catalog_name, catalog_id, file_id in
many_to_many_temp
        for f in files if f.id==file_id]

    print('\nЗадание А1')
```

```
    res_11 = sorted(one_to_many, key=itemgetter(2))
    print(res_11)

    print('\nЗадание А2')
    res_12_unsorted = []
    # Перебираем все каталоги
    for c in catalogs:
        # Список файлов каталога
        cat_files = list(filter(lambda i:
i[2]==c.catalog_name, one_to_many))
        # Если каталог не пустой
        if len(cat_files) > 0:
            # Размеры файлов каталога
            cat_sizes = [size for _,size,_ in
cat_files]
            # Суммарный размер файлов каталога
            cat_sizes_sum = sum(cat_sizes)
            res_12_unsorted.append((c.catalog_name,
cat_sizes_sum))

    # Сортировка по суммарному размеру файлов
    res_12 = sorted(res_12_unsorted, key=itemgetter(1),
reverse=True)
    print(res_12)

    print('\nЗадание А3')
    res_13 = {}
    # Перебираем все каталоги
    for c in catalogs:
        if 'pics' in c.catalog_name:
            # Список файлов каталога
            cat_f = list(filter(lambda i:
i[2]==c.catalog_name, many_to_many))
            # Только название файлов
            cat_f_names = [x for x,_,_ in cat_f]
            # Добавляем результат в словарь
            # ключ - каталог, значение - список файлов
            res_13[c.catalog_name] = cat_f_names

    print(res_13)

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

Результаты работы программы:

Задание A1

[('pic1', 100, 'pics'), ('pic2', 110, 'pics'), ('program1', 50, 'programs'), ('program2', 55, 'programs'), ('txtA', 20, 'texts'), ('txtB', 30, 'texts')]

Задание A2

[('pics', 210), ('programs', 105), ('texts', 50)]

Задание A3

{'pics': ['pic1', 'pic2'], 'pics2': ['pic1', 'pic2']}