Попов Александр, ИУ5-31Б

Текст программы РК1 после рефакторинга:

```
# 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 and texts'),
    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(3,3),
    Files_of_Catalogs(4,3),
    Files_of_Catalogs(5,4),
    Files_of_Catalogs(6,4)
    ]

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

```python
# Соединение данных многие-ко-многим
def many_to_many(files, catalogs):
    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]

    return [(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]

def A1(files, catalogs) -> list:
    res_11 = sorted(one_to_many(files, catalogs), key=itemgetter(2))
    return (list(res_11))

def A2(files, catalogs) -> list:
    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(files, catalogs)))
        # 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=False)
    return (list(res_12))

def A3(files, catalogs, str_find) -> dict:
    res_13 = {}
    # Перебираем все каталоги
    for c in catalogs:
        if str_find in c.catalog_name:
            # Список файлов каталога
            cat_f = list(filter(lambda i: i[2]==c.catalog_name, many_to_many(files, catalogs)))
            # 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
    return (res_13)

if __name__ == '__main__':
    print('Задание 1')
    print(A1(files,catalogs))
    print('Задание 2')
    print(A2(files,catalogs))
    print('Задание 3')
    print(A3(files,catalogs, 'texts'))
```

Текст программы тестирования РК2:

```
import unittest
from RK1_refactoring import File, Catalog, Files_of_Catalogs, A1, A2, A3

class Testing_RK1(unittest.TestCase):

    def setUp(self):
        self.files = [
            File(1, 'txtA', 20, 1),
            File(2, 'txtB', 30, 1),
            File(3, 'program1', 50, 2),
            File(4, 'program2', 55, 2)
            ]

        self.catalogs = [
            Catalog(1, 'texts'),
            Catalog(2, 'programs'),
            Catalog(3, 'pics and texts')
            ]

        self.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(3,3),
            Files_of_Catalogs(4,3)
            ]


    def test_A1(self):
        result_true = [
            ('program1', 50, 'programs'),
            ('program2', 55, 'programs'),
            ('txtA', 20, 'texts'),
            ('txtB', 30, 'texts')]
        result = A1(self.files, self.catalogs)
        self.assertEqual(result, result_true)

    def test_A2(self):
        result_true = [
            ('texts', 50),
            ('programs', 105)]
        result = A2(self.files, self.catalogs)
        self.assertEqual(result, result_true)

    def test_A3(self):
        result_true = {
            'texts' : ['txtA','txtB'],
            'pics and texts' : ['program1','program2']
            }
        result = A3(self.files, self.catalogs, 'texts')
        self.assertEqual(result, result_true)

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

Результат выполнения:

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========================= RESTART: D:/Python 38/rk2.py =========================
...
----------------------------------------------------------------------
Ran 3 tests in 0.015s

OK
>>>
```