МГТУ им. Н. Э. Баумана

Отчет по рубежному контролю №2
по курсу «Базовые компоненты и интернет-технологии»
Вариант 15.

Руководитель:
Гапанюк Ю. Е.
16.12.2022

Выполнил:
Студент группы ИУ5-34Б
Новиков Б. В.
16.12.2022

2022 г.

Полученное задание:

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.
2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

Запросы:

1. «Каталог» и «Файл» связаны соотношением один-ко-многим. Выведите список всех каталогов, у которых название начинается с буквы «А», и список хранящихся в них файлов.
2. «Каталог» и «Файл» связаны соотношением один-ко-многим. Выведите список каталогов с максимальным размером файла в каждом каталоге, отсортированный по максимальному размеру.
3. «Каталог» и «Файл» связаны соотношением многие-ко-многим. Выведите список всех связанных файлов и каталогов, отсортированный по каталогам, сортировка по файлам произвольная.

Текст программы:

main.py:

```
from operator import itemgetter

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

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

class FileCatalog:
    """
    'Файлы каталога' для реализации
    связи многие-ко-многим
    """
    def __init__(self, catalog_id, file_id):
        self.catalog_id = catalog_id
        self.file_id = file_id

# Каталоги
Catalogs = [
    Catalog(1, 'Downloads'),
    Catalog(2, 'Pictures'),
    Catalog(3, 'Applications'),
```

```
]

# Файлы
Files = [
    File(1, 'word-download.txt', 3, 1),
    File(2, 'alps.png', 7, 2),
    File(3, 'mount.jpg', 8, 2),
    File(4, 'tree.jpg', 6, 2),
    File(5, 'mario.exe', 5000, 3),
]

# Каталоги файлов
File_Catalogs = [
    FileCatalog(1,1),
    FileCatalog(2,2),
    FileCatalog(2,3),
    FileCatalog(2,4),
    FileCatalog(3,5),
]


    # Соединение данных один-ко-многим
one_to_many = [(f.fname, f.size, c.name)
    for c in Catalogs
    for f in Files
    if f.catalog_id==c.id]

    # Соединение данных многие-ко-многим
many_to_many_temp = [(c.name, fc.catalog_id, fc.file_id)
    for c in Catalogs
    for fc in File_Catalogs
    if c.id==fc.catalog_id]

many_to_many = [(f.fname, 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 task1(one_to_many):
    res_11 = {}
    # Перебираем все каталоги
    for c in Catalogs:
        if c.name[0] == 'A':
            # Список файлов каталога
            c_Files = list(filter(lambda i: i[2]==c.name, one_to_many))
            # Только имя файла
            c_Files_names = [x for x,_,_ in c_Files]
            # Добавляем результат в словарь
            # ключ - каталог, значение - список файлов
            res_11[c.name] = c_Files_names
    return res_11

def task2(one_to_many):
```

```
    res_12_unsorted = []
    # Перебираем все каталоги
    for c in Catalogs:
        # Список файлов каталога
        c_Files = list(filter(lambda i: i[2]==c.name, one_to_many))
        # Если каталог не пустой
        if len(c_Files) > 0:
            # Зарплаты сотрудников отдела
            c_size = [size for _,size,_ in c_Files]
            # Суммарная зарплата сотрудников отдела
            c_size_max = max(c_size)
            res_12_unsorted.append((c.name, c_size_max))
    # Сортировка по максимальному размеру файла
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return res_12

def task3(many_to_many):
    res_13 = sorted(many_to_many, key=itemgetter(2))
    return res_13
```

test_main.py:

```
import pytest
from main import *
from main import task1, task2, task3, one_to_many, many_to_many

def test_task1():
    res = task1(one_to_many)
    expected = {'Applications':['mario.exe']}
    assert res == expected

def test_task2():
    res = task2(one_to_many)
    expected = [('Applications', 5000), ('Pictures', 8), ('Downloads', 3)]
    assert res == expected


def test_task3():
    res = task3(one_to_many)
    expected = [('mario.exe', 5000, 'Applications'),
                ('word-download.txt', 3, 'Downloads'),
                ('alps.png', 7, 'Pictures'),
                ('mount.jpg', 8, 'Pictures'),
                ('tree.jpg', 6, 'Pictures')]
    assert res == expected
```

Результат выполнения: