Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра ИУ5.
Курс «Базовые компоненты интернет технологий»

Отчет по рубежному контролю №2

Выполнил:
студентка группы ИУ5-31Б
Покшубина С.С.
Подпись и дата: 19.12.2021

Проверил:
Гапанюк Ю.Е.
Подпись и дата:

г. Москва, 2021 г.

Рубежный контроль №2
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).

Текст программы
main.py

```
# используется для сортировки
from operator import itemgetter


class File:
    """ Файл """

    def __init__(self, id, name, size, cat_id):
        self.id = id
        self.name = name
        self.size = size
        self.cat_id = cat_id


class Catalog:
    """Каталог файлов"""

    def __init__(self, id, cat_name):
        self.id = id
        self.cat_name = cat_name


class CatalogFiles:
    """ Файлы каталога для реализации связи многие-ко-многим """

    def __init__(self, cat_id, file_id):
        self.cat_id = cat_id
        self.file_id = file_id


# Каталоги
catalogs = [
    Catalog(1, 'Папка 1'),
    Catalog(2, 'Папка 2'),
    Catalog(3, 'Папка 3'),
    Catalog(11, 'Папка 1 (другая)'),
    Catalog(22, 'Папка 2 (другая)'),
    Catalog(33, 'Папка 3 (другая)'),
]
# Файлы
files = [
    File(1, 'Курсовая работа', 25, 1),
    File(2, 'Отчет по лабораторной работе', 35, 2),
    File(3, 'Отчет по практике', 45, 3),
    File(4, 'Реферат', 35, 3),
    File(5, 'Аннотация по АСОИУ', 25, 3),
]
catalogs_files = [
    CatalogFiles(1, 1),
```

```
    CatalogFiles(2, 2),
    CatalogFiles(3, 3),
    CatalogFiles(3, 4),
    CatalogFiles(3, 5),
    CatalogFiles(11, 1),
    CatalogFiles(22, 2),
    CatalogFiles(33, 3),
    CatalogFiles(33, 4),
    CatalogFiles(33, 5),
]


def sorting_by_name(table):
    return sorted(table, key=itemgetter(2))


def sorting_by_sum_size(table, catalogs):
    result_unsorted = []
    # Перебираем все каталоги
    for c in catalogs:
        # Список файлов каталога
        c_files = list(filter(lambda i: i[2] == c.cat_name, table))
        # Если каталог не пустой
        if len(c_files) > 0:
            # Размеры файлов каталога
            c_sizes = [size for _, size, _ in c_files]
            # Суммарный размер файлов каталога
            c_sizes_sum = sum(c_sizes)
            result_unsorted.append((c.cat_name, c_sizes_sum))
    # Сортировка по суммарному размеру
    return sorted(result_unsorted, key=itemgetter(1), reverse=True)


def output_files_of_catalogs_with_PAPKA2(table, catalogs):
    result = {}
    # Перебираем все каталоги
    for c in catalogs:
        if 'Папка 2' in c.cat_name:
            # Список файлов каталога
            c_files = list(filter(lambda i: i[2] == c.cat_name, table))
            # Только название файлов
            c_files_names = [x for x, _, _ in c_files]
            # Добавляем результат в словарь
            # ключ - каталог, значение - список названий
            result[c.cat_name] = c_files_names
    return result



def main():
    """Основная функция"""
    # Соединение данных один-ко-многим
    one_to_many = [(f.name, f.size, c.cat_name)
                   for c in catalogs
                   for f in files
                   if f.cat id == c.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(c.cat_name, fc.cat_id, fc.file_id)
                         for c in catalogs
                         for fc in catalogs_files
                         if c.id == fc.cat_id]
    many_to_many = [(f.name, f.size, cat_name)
                    for cat_name, cat_id, file_id in many_to_many_temp
```

```
                    for f in files if f.id == file_id]
    print('Задание А1')
    print(sorting_by_name(one_to_many))

    print('\nЗадание А2')
    print(sorting_by_sum_size(one_to_many, catalogs))

    print('\nЗадание А3')
    print(output_files_of_catalogs_with_PAPKA2(many_to_many, catalogs))


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

tests.py

```
from main import File, Catalog, CatalogFiles, sorting_by_name,
sorting_by_sum_size, output_files_of_catalogs_with_PAPKA2
import unittest


class Tests(unittest.TestCase):
    def setUp(self):
        # Библиотеки
        # Каталоги
        self.catalogs = [
            Catalog(1, 'Папка 1'),
            Catalog(2, 'Папка 2'),
            Catalog(3, 'Папка 3'),
            Catalog(11, 'Папка 1 (другая)'),
            Catalog(22, 'Папка 2 (другая)'),
            Catalog(33, 'Папка 3 (другая)'),
        ]
        # Файлы
        self.files = [
            File(1, 'Курсовая работа', 25, 1),
            File(2, 'Отчет по лабораторной работе', 35, 2),
            File(3, 'Отчет по практике', 45, 3),
            File(4, 'Реферат', 35, 3),
            File(5, 'Аннотация по АСОИУ', 25, 3),
        ]
        self.catalogs_files = [
            CatalogFiles(1, 1),
            CatalogFiles(2, 2),
            CatalogFiles(3, 3),
            CatalogFiles(3, 4),
            CatalogFiles(3, 5),
            CatalogFiles(11, 1),
            CatalogFiles(22, 2),
            CatalogFiles(33, 3),
            CatalogFiles(33, 4),
            CatalogFiles(33, 5),
        ]
        # Соединение данных один-ко-многим
        self.one_to_many = [(f.name, f.size, c.cat_name)
                            for c in self.catalogs
                            for f in self.files
                            if f.cat_id == c.id]

        # Соединение данных многие-ко-многим
        self.many_to_many_temp = [(c.cat_name, fc.cat_id, fc.file_id)
                                  for c in self.catalogs
                                  for fc in self.catalogs_files
                                  if c.id == fc.cat_id]
```

```
        self.many_to_many = [(f.name, f.size, cat_name)
                             for cat_name, cat_id, file_id in
self.many_to_many_temp
                             for f in self.files if f.id == file_id]

    def test_sorting_by_name(self):
        result = sorting_by_name(self.one_to_many)
        desired_result = [('Курсовая работа', 25, 'Папка 1'), ('Отчет по
лабораторной работе', 35, 'Папка 2'),
                          ('Отчет по практике', 45, 'Папка 3'), ('Реферат',
35, 'Папка 3'),
                          ('Аннотация по АСОИУ', 25, 'Папка 3')]
        self.assertEqual(result, desired_result)

    def test_sorting_by_sum(self):
        result = sorting_by_sum_size(self.one_to_many, self.catalogs)
        desired_result = [('Папка 3', 105), ('Папка 2', 35), ('Папка 1', 25)]
        self.assertEqual(result, desired_result)

    def test_output_PAPKA2(self):
        result = output_files_of_catalogs_with_PAPKA2(self.many_to_many,
self.catalogs)
        desired_result = {'Папка 2': ['Отчет по лабораторной работе'], 'Папка
2 (другая)': ['Отчет по лабораторной '

'работе']}
        self.assertEqual(result, desired_result)
```

Экранные формы с примерами выполнения работ

main.py

```
Задание А1
[('Курсовая работа', 25, 'Папка 1'), ('Отчет по лабораторной работе', 35, 'Папка 2'), ('Отчет по практике', 45, 'Папка 3'), ('Реферат', 35, 'Папка 3'), ('Аннотация по АСО

Задание А2
[('Папка 3', 105), ('Папка 2', 35), ('Папка 1', 25)]

Задание А3
{'Папка 2': ['Отчет по лабораторной работе'], 'Папка 2 (другая)': ['Отчет по лабораторной работе']}
```

tests.py

```
Ran 3 tests in 0.005s


OK
```