Щетинин Михаил Ярославович

ИУ5-34Б

Отчет по РК2 по дисциплине

“Парадигмы и конструкции языков программирования”

Вариант Г. Вариант предметной области 15 (Файл - Каталог файлов)

Рефракторинг программы:

```
from operator import itemgetter

class File:
    """Файл size - размер вайла в МБ"""
    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 файлов"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class Files_in_Catalog:
    """Файлы в каталоге' 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, 'Аспирантские закупки'),
    Catalog(2, 'Закупки для университета'),
    Catalog(3, 'Закупки для офиса'),
    Catalog(4, 'Закупки для партнеров')
]

# Файлы
files = [
    File(1, 'Сумма закупки', 500, 1),
```

```
    File(2, 'Руководитель закупки', 320, 1),
    File(3, 'Номер контрактов', 490, 2),
    File(4, 'Поставщики', 550, 3),
    File(5, 'Юр. данные', 120, 4)
]

files_in_catalog = [
    Files_in_Catalog(1, 1),
    Files_in_Catalog(1, 2),
    Files_in_Catalog(2, 3),
    Files_in_Catalog(3, 4),
    Files_in_Catalog(4, 5)
]

def get_one_to_many(catalogs, files):
    """Соединение данных один-ко-многим"""
    return [(f.name, f.size, u.name)
            for u in catalogs
            for f in files
            if f.catalog_id == u.id]

def get_many_to_many(catalogs, files, files_in_catalog):
    """Соединение данных многие-ко-многим"""
    many_to_many_temp = [(u.name, fu.catalog_id, fu.file_id)
                         for u in catalogs
                         for fu in files_in_catalog
                         if u.id == fu.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 task_G1(one_to_many):
    """Задание Г1: найти все файлы, каталоги которых начинаются с 'А'"""
    return [item for item in one_to_many if item[2].startswith('A')]

def task_G2(catalogs, one_to_many):
    """Задание Г2: найти максимальный размер файла в каждом каталоге"""
    res_unsorted = []
    for u in catalogs:
        u_files = list(filter(lambda i: i[2] == u.name, one_to_many))
        if len(u_files) > 0:
            u_sizes = [size for _, size, _ in u_files]
```

```
            u_max_size = max(u_sizes)
            res_unsorted.append((u.name, u_max_size))
    return sorted(res_unsorted, key=itemgetter(1), reverse=True)

def task_G3(catalogs, many_to_many):
    """Задание Г3: список всех файлов в каждом каталоге"""
    res = {}
    for u in catalogs:
        u_files = list(filter(lambda i: i[2] == u.name, many_to_many))
        u_file_names = [name for name, _, _ in u_files]
        res[u.name] = u_file_names
    return res

def main():
    """Основная функция для вызова задач"""
    one_to_many = get_one_to_many(catalogs, files)
    many_to_many = get_many_to_many(catalogs, files, files_in_catalog)

    print('Задание Г1:', task_G1(one_to_many))
    print('Задание Г2:', task_G2(catalogs, one_to_many))
    print('Задание Г3:', task_G3(catalogs, many_to_many))

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

Модульные тесты

```
import unittest
from main import get_one_to_many, get_many_to_many, task_G1, task_G2,
task_G3, catalogs, files, files_in_catalog

class TestFileCatalog(unittest.TestCase):

    def setUp(self):
        """Set up the one-to-many and many-to-many relationships for testing."""
        self.one_to_many = get_one_to_many(catalogs, files)
        self.many_to_many = get_many_to_many(catalogs, files, files_in_catalog)

    def test_task_G1(self):
        """Test task_G1 for finding files in catalogs starting with 'A'."""
        expected_result = [('Сумма закупки', 500, 'Аспирантские закупки'),
```

```
                    ('Руководитель закупки', 320, 'Аспирантские закупки')]
        self.assertEqual(task_G1(self.one_to_many), expected_result)

    def test_task_G2(self):
        """Test task_G2 for finding the largest file in each catalog."""
        expected_result = [('Закупки для офиса', 550),
                    ('Аспирантские закупки', 500),
                    ('Закупки для университета', 490),
                    ('Закупки для партнеров', 120)]
        self.assertEqual(task_G2(catalogs, self.one_to_many), expected_result)

    def test_task_G3(self):
        """Test task_G3 for listing all files by catalog."""
        expected_result = {
            'Аспирантские закупки': ['Сумма закупки', 'Руководитель закупки'],
            'Закупки для университета': ['Номер контрактов'],
            'Закупки для офиса': ['Поставщики'],
            'Закупки для партнеров': ['Юр. данные']
        }
        self.assertEqual(task_G3(catalogs, self.many_to_many), expected_result)

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