Московский государственный технический университет им. Н.Э. Баумана

Факультет “Радиотехнический”
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю №2

Вариант предметной области из РК1 №15: Файл, Каталог файлов
Вариант запросов: E

Выполнил:
студент группы РТ5-31Б:
Сахарова О.П

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2025 г.

Описание задания

Условия рубежного контроля №2 по курсу ПиК ЯП

Рубежный контроль представляет собой разработку тестов на языке Python.

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

```
import unittest


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


class File:
    def __init__(self, id, title, size, catalog_id=None):
        self.id = id
        self.title = title
        self.size = size
        self.catalog_id = catalog_id


class Link:
    def __init__(self, file_id, catalog_id):
        self.file_id = file_id
        self.catalog_id = catalog_id


def create_data():
    catalogs = [
        Catalog(1, "Рабочий каталог файлов"),
        Catalog(2, "Учёба"),
        Catalog(3, "Файлы отпуска 2024"),
        Catalog(4, "Каталог детских фото"),
        Catalog(5, "Каталог файлов со школы")
    ]
    files = [
        File(1, "Анализ рынка труда.txt", 150, 1),
```

```
            File(2, "Сводная таблица 2024.xlsx", 320.50, 1),
            File(3, "ДЗ_социология.docx", 400, 2),
            File(4, "Маша фото 22.08.2024.png", 750, 4),
            File(5, "Алиса и Маша фото.png", 322, 4),
            File(6, "5А класс.png", 300, 5),
            File(7, "Альпы фото.png", 750, 3)
        ]
        links = [
            Link(1, 1),
            Link(2, 1),
            Link(3, 2),
            Link(4, 3),
            Link(5, 3),
            Link(5, 4),
            Link(7, 3)
        ]
        return catalogs, files, links


    def get_one_to_many(catalogs, files):
        return [(f.title, f.size, c.name) for c in catalogs for f in files if f.catalog_id == c.id]


    def get_many_to_many(catalogs, files, links):
        res = [(c.name, l.catalog_id, l.file_id) for c in catalogs for l in links if c.id ==
l.catalog_id]
        return [(f.title, f.size, catalog_name) for catalog_name, catalog_id, file_id in res for f in
files if f.id == file_id]


    def query_1(catalogs, one_to_many):
        #Запрос 1. Каталоги со словом 'каталог' в названии
        result = {}
        catalogs_filtered = [c for c in catalogs if 'каталог' in c.name.lower()]
        for catalog in catalogs_filtered:
            catalog_files = [title for title, size, c_name in one_to_many
                             if c_name == catalog.name]
            result[catalog.name] = catalog_files
        return result


    def query_2(catalogs, one_to_many):
```

```
    #Запрос 2. Каталоги по среднему размеру файлов (от меньшего к большему)
    result = []
    for catalog in catalogs:
        catalog_files = [(title, size) for title, size, c_name in one_to_many
                         if c_name == catalog.name]
        if catalog_files:
            sizes = [size for _, size in catalog_files]
            avg_size = round(sum(sizes) / len(sizes), 2)
            result.append((catalog.name, avg_size))
    return sorted(result, key=lambda x: x[1])


def query_3(many_to_many):
    #Запрос 3: Файлы, начинающиеся на 'A'
    result = {}
    files_starting_with_a = set([title for title, size, catalog_name in many_to_many
                                 if title and title[0].upper() == 'A'])
    for file_title in files_starting_with_a:
        file_catalogs = [catalog_name for title, size, catalog_name in many_to_many
                         if title == file_title]
        unique_catalogs = list(set(file_catalogs))
        result[file_title] = unique_catalogs
    return result


class TestCatalogSystem(unittest.TestCase):
    def setUp(self):
        self.catalogs, self.files, self.links = create_data()
        self.one_to_many = get_one_to_many(self.catalogs, self.files)
        self.many_to_many = get_many_to_many(self.catalogs, self.files, self.links)

    def test_query_1(self):
        result = query_1(self.catalogs, self.one_to_many)
        self.assertIsInstance(result, dict)
        self.assertEqual(len(result), 3)
        self.assertIn("Рабочий каталог файлов", result)
        self.assertIn("Каталог детских фото", result)
        self.assertIn("Каталог файлов со школы", result)
        self.assertEqual(result["Рабочий каталог файлов"], ["Анализ рынка труда.txt",
"Сводная таблица 2024.xlsx"])

    def test_query_2(self):
```

```
        result = query_2(self.catalogs, self.one_to_many)
        self.assertIsInstance(result, list)
        self.assertEqual(len(result), 5)
        sizes = [avg_size for _, avg_size in result]
        self.assertEqual(sizes, sorted(sizes))
        for name, avg_size in result:
            if name == "Рабочий каталог файлов":
                self.assertAlmostEqual(avg_size, 235.25, places=2)

    def test_query_3(self):
        result = query_3(self.many_to_many)
        self.assertIsInstance(result, dict)
        self.assertEqual(len(result), 3)
        self.assertIn("Анализ рынка труда.txt", result)
        self.assertIn("Алиса и Маша фото.png", result)
        self.assertIn("Альпы фото.png", result)
        self.assertNotIn("Маша фото 22.08.2024.png", result)
        self.assertEqual(len(result["Алиса и Маша фото.png"]), 2)
        self.assertIn("Файлы отпуска 2024", result["Алиса и Маша фото.png"])
        self.assertIn("Каталог детских фото", result["Алиса и Маша фото.png"])


def main():
    catalogs, files, links = create_data()
    one_to_many = get_one_to_many(catalogs, files)
    many_to_many = get_many_to_many(catalogs, files, links)

    print("Запрос 1. Каталоги со словом 'каталог' в названии")
    result1 = query_1(catalogs, one_to_many)
    for catalog_name, file_list in result1.items():
        print(f"{catalog_name}: {', '.join(file_list)}")

    print("\nЗапрос 2. Каталоги по среднему размеру файлов (от меньшего к
большему)")
    result2 = query_2(catalogs, one_to_many)
    for catalog_name, avg_size in result2:
        print(f"{catalog_name}: {avg_size}")

    print("\nЗапрос 3: Файлы, начинающиеся на 'А'")
    result3 = query_3(many_to_many)
    for file_name, catalog_list in result3.items():
        print(f"Файл '{file_name}' находится в каталогах: {', '.join(catalog_list)}")
```

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

Примеры выполнения программы

```
Testing started at 23:13 ...

Launching unittests with arguments python -m unittest C:\Users\olgas\учёба\ПиКЯП\rk_2\rk_1.py in C:\Users\olgas\учёба\ПиКЯП\rk_2


Ran 3 tests in 0.002s

OK
```

```
PS C:\Users\olgas\учёба\ПиКЯП\rk_2> python
Запрос 1. Каталоги со словом 'каталог' в названии
Рабочий каталог файлов: Анализ рынка труда.txt, Сводная таблица 2024.xlsx
Каталог детских фото: Маша фото 22.08.2024.png, Алиса и Маша фото.png
Каталог файлов со школы: 5А класс.png

Запрос 2. Каталоги по среднему размеру файлов (от меньшего к большему)
Рабочий каталог файлов: 235.25
Каталог файлов со школы: 300.0
Учёба: 400.0
Каталог детских фото: 536.0
Файлы отпуска 2024: 750.0

Запрос 3: Файлы, начинающиеся на 'А'
Файл 'Альпы фото.png' находится в каталогах: Файлы отпуска 2024
Файл 'Алиса и Маша фото.png' находится в каталогах: Файлы отпуска 2024, Каталог детских фото
Файл 'Анализ рынка труда.txt' находится в каталогах: Рабочий каталог файлов
PS C:\Users\olgas\учёба\ПиКЯП\rk_2>
```