МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет "Информатика и системы управления"
Кафедра "Системы обработки информации и управления"

Дисциплина "Парадигмы и конструкции языков программирования"

Отчет по рубежному контролю №2

Выполнил:
Студент группы ИУ5-35Б
Шаньгин Н.А.
Преподаватель:
Гапанюк Ю.Е.

Москва 2025

Задание:

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).

rk_1/main.py

```
"""
Вариант 25Г: Класс 1 - "Раздел",
             Класс 2 - "Документ"
Запросы: 1) Раздел начинается с буквы "l" и список документов в нём (1:M)
         2) Список разделов с максимальным размером файлов в каждом разделе,
         отсортированный по максимальному размеру (1:M)
         3) Вывести список разделов с их файлами отсортированный по разделам
(M:M)

"""

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


class Document:
    def __init__(self, id, name, size, directory_id):
        self.id = id
        self.name = name
        self.size = size
        self.directory_id = directory_id


class DirectoryDocument:
    def __init__(self, directory_id, document_id):
        self.directory_id = directory_id
        self.document_id = document_id


directories = [
    Directory(1, "university docx"),
    Directory(2, "pcpl"),
    Directory(3, "lab0"),
    Directory(4, "lab1"),
    Directory(5, "rk1")
]


documents = [
    Document(1, "lab0.py", 1, 3),
    Document(2, "lab1.py", 3, 4),
    Document(3, "lab1OOP.py", 4, 4),
    Document(4, "rk1.py", 2, 5),
    Document(5, "StudySchedule.docx", 350, 1)
]


directories_documents = [
```

```
    DirectoryDocument(1, 5),
    DirectoryDocument(2, 1),
    DirectoryDocument(2, 2),
    DirectoryDocument(2, 3),
    DirectoryDocument(2, 4),
    DirectoryDocument(3, 1),
    DirectoryDocument(4, 2),
    DirectoryDocument(4, 3),
    DirectoryDocument(5, 4)
]


def get_directories_by_starting_letter(directories, documents, letter='l'):
    result = {}

    filtered_directories = [directory for directory in directories
                            if directory.name.lower().startswith(letter.lower())]

    for directory in filtered_directories:
        directory_documents = [doc for doc in documents if doc.directory_id ==
directory.id]
        result[directory.name] = directory_documents

    return result


def first_task(output = True):
    if output:
        print("Первое задание:\n")
    result = get_directories_by_starting_letter(directories, documents, 'l')

    i = 1
    for directory_name, docs in result.items():
        if output: print(f"{i}. {directory_name}:")
        if docs:
            for doc in docs:
                if output: print(f"\t- {doc.name}")
        else:
            if output: print("\tДокументов нет")
        if output: print()
        i += 1
    if output:
        print("Конец первого задания\n")
    return result


def filter_directories_by_max_size_of_documents(directories, documents):
    directory_and_max_size = {}

    for directory in directories:
```

```
        directory_documents = [doc for doc in documents if doc.directory_id ==
directory.id]
        if directory_documents:
            max_document = max(directory_documents, key=lambda doc: doc.size)
            directory_and_max_size[directory.name] = max_document.size
        else:
            directory_and_max_size[directory.name] = 0

    result = sorted(directory_and_max_size.items(), key=lambda item: item[1],
reverse=True)

    return result


def second_task(output = True):
    if output:
        print("Второе задание:\n")

    result = filter_directories_by_max_size_of_documents(directories, documents)

    i = 1
    for directory_name, max_size in result:
        if max_size > 0:
            if output: print(f"{i}. {directory_name}: {max_size} KB")
        else:
            if output: print(f"{i}. {directory_name}: нет документов")
        i += 1

    if output:
        print("Конец второго задания\n")
    return result


def get_directories_with_documents(directories, documents,
directories_documents):
    result = {}

    for directory in directories:
        document_ids = [conn.document_id for conn in directories_documents
                        if conn.directory_id == directory.id]

        directory_documents = [doc for doc in documents if doc.id in
document_ids]

        result[directory.name] = directory_documents

    return result


def third_task(output = True):
    if output:
```

```
        print("Третье задание:\n")

    result = get_directories_with_documents(directories, documents,
directories_documents)

    i = 1
    for directory_name, docs in result.items():
        if output: print(f"{i}. {directory_name}:")
        if docs:
            for doc in docs:
                if output: print(f"\t- {doc.name}")
        else:
            if output: print("\tДокументов нет")
        if output: print()
        i += 1

    if output:
        print("Конец третьего задания\n")
    return result


def main():
    first_task()
    second_task()
    third_task()


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

rk_2/test.py

```python
import unittest
import os, sys
sys.path.append(os.path.join(os.path.dirname(__file__), "..", "rk_1"))
from main import first_task, second_task, third_task, Directory, Document

class TestFileCatalogue(unittest.TestCase):
    def test_first_task(self):
        result = first_task(output=False)
        self.assertEqual(len(result["lab1"]), 2)
        self.assertEqual(result["lab0"][0].id, 1)

    def test_second_task(self):
        result = second_task(output=False)
        self.assertEqual(result[2][1], 2)
        self.assertEqual(result[3][1], 1)
        self.assertEqual(result[4][1], 0)

    def test_third_task(self):
        result = third_task(output=False)
        self.assertEqual(result["pcpl"][2].name, "lab1OOP.py")
        self.assertEqual(result["university docx"][0].size, 350)
        self.assertEqual(result["lab0"][0].id, 1)
        self.assertEqual(result["lab1"][0].size, 3)

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

Результат работы

```
PS D:\Учёба\3 семестр\ПиКЯП\Shangin-PCPL-Labs-2025\rk_2> py test.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
PS D:\Учёба\3 семестр\ПиКЯП\Shangin-PCPL-Labs-2025\rk_2>
```