МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

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

Дисциплина "Основы программирования"

Отчет по рубежному контролю №2

Выполнил:
Студент группы ИУ5-35Б
Королев М.О.
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).

https://github.com/koromax/Korolev-PCPL-Labs-2025/tree/main/RK2

Листинг кода:

```
# ../RK1/main.py
"""
Вариант 15Г: Класс 1 -- Файл, Класс 2 -- Каталог файлов
Запросы:  1) ...название начинается с буквы «А», и список...
          2) ...список каталогов с максимальным размеров файлов в каждом каталоге,
отсортированный по максимальному размеру.
          3) ...список всех связанных файлов и каталогов, отсортированный по отделам...
"""

class File:
    def __init__(self, id, name, size, catalogue_id):
        self.id = id
        self.name = name
        self.size = size
        self.catalogue_id = catalogue_id

    def __repr__(self):
        return f"File(id={self.id}, name='{self.name}', size={self.size},
catalogue_id={self.catalogue_id})"

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

    def __repr__(self):
        return f"Catalogue(id={self.id}, name='{self.name}')"

class FileCatalogue:
    def __init__(self, file_id, catalogue_id):
        self.file_id = file_id
        self.catalogue_id = catalogue_id

files = [
    File(1, "report.txt", 1024, 1),
    File(2, "Ivanov_document.pdf", 2048, 2),
    File(3, "presentation.pptx", 3072, 1),
    File(4, "Sidorov_data.txt", 512, 2),
    File(5, "Petrov_notes.doc", 4096, 3),
    File(6, "image.jpg", 1536, 2),
    File(7, "archive.zip", 10240, 3)
]

catalogues = [
    Catalogue(1, "Documents"),
    Catalogue(2, "User Files"),
    Catalogue(3, "Archives")
]

file_catalogue = [
    FileCatalogue(1, 1),
    FileCatalogue(2, 2),
    FileCatalogue(2, 1),
    FileCatalogue(3, 1),
    FileCatalogue(4, 2),
    FileCatalogue(5, 3),
    FileCatalogue(5, 2),
    FileCatalogue(6, 2),
```

```
    FileCatalogue(7, 3)
]

def first_task(output=True):
    ret = {}
    if output: print("--- TASK ONE ---")

    for catalogue in [c for c in catalogues if c.name.startswith('A')]:
        if output: print(f"Catalogue: {catalogue.name}")
        files_in_catalogue = [f for f in files if f.catalogue_id == catalogue.id]
        ret[catalogue.name] = files_in_catalogue
        if files_in_catalogue:
            for f in files_in_catalogue:
                if output: print("   ", f)
        else:
            if output: print("    is empty")

    if output: print("--- TASK END ---")
    return ret

def second_task(output=True):
    ret = []
    if output: print("--- TASK TWO ---")

    cats = sorted(catalogues, key=lambda cat: max(
        (f.size for f in files if f.catalogue_id == cat.id),
        default = 0
    ), reverse=True)

    for c in cats:
        size = max((f.size for f in files if f.catalogue_id == c.id), default = 0)
        if size > 0:
            if output: print(f"{str(c).ljust(34, ' ')} with max file size of {str(size).rjust(5, ' ')}
bytes")
            ret.append((c, size))

    if output: print("--- TASK END ---")
    return ret

def third_task(output=True):
    if output: print("-- THIRD TASK --")
    cells_interlinked = sorted([
        (catalogue, file)
        for relation in file_catalogue
        for catalogue in catalogues
        if catalogue.id == relation.catalogue_id
        for file in files
        if file.id == relation.file_id
    ],
    key=lambda x: (x[0].name, -x[1].size))

    for x in cells_interlinked:
        if output: print(f"{str(x[0]).ljust(34, ' ')} {x[1]}")

    if output: print("--- TASK END ---")
    return cells_interlinked


if __name__ == "__main__":
```

```
    print()
    first_task()
    print()
    second_task()
    print()
    third_task()
```

```
# test.py
import unittest
import sys
import os
sys.path.append(os.path.join(os.path.dirname(__file__), "..", "RK1"))
from main import *

class TestFileCatalogue(unittest.TestCase):
    def test_first_task(self):
        result = first_task(output=False)
        self.assertEqual(len(result["Archives"]), 2)
        self.assertEqual(result["Archives"][0].catalogue_id, 3)

    def test_second_task(self):
        result = second_task(output=False)
        self.assertEqual(result[0][1], 10240)
        self.assertEqual(result[1][1], 3072)
        self.assertEqual(result[2][1], 2048)

    def test_third_task(self):
        result = third_task(output=False)
        self.assertEqual(result[0][0].name, "Archives")
        self.assertEqual(result[0][1].size, 10240)
        self.assertEqual(result[-1][0].name, "User Files")
        self.assertEqual(result[-1][1].size, 512)

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

Результат работы программы:

```
[запуск программы]
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
[программа завершила выполнение]
```