Московский государственный технический университет им. Н.Э. Баумана

Факультет информатика и системы управления
Кафедра системы обработки информации и управления

Курс «Парадигмы и конструкции языков программирования»
Отчет по рубежному контролю №2
Вариант А13

Выполнил:
Студент группы ИУ5-31Б:
Попов А.С.
Подпись и дата:

Проверил:
Преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2025 г.

Постановка задачи

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

rk2.py

```
class Book:
    def __init__(self, id, title, author, pages, libraryID):
        self.id = id
        self.title = title
        self.author = author
        self.pages = pages
        self.libraryID = libraryID

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

class BookLibrary:
    def __init__(self, libraryID, bookID):
        self.libraryID = libraryID
        self.bookID = bookID

books = [
    Book(1, "1984", "Оруэлл", 328, 1),
    Book(2, "Дивный новый мир", "Хаксли", 311, 2),
    Book(3, "451° по Фаренгейту", "Брэдбери", 256, 1),
    Book(4, "Мы", "Замятин", 232, 2),
    Book(5, "Солярис", "Лем", 204, 3),
    Book(6, "Трудно быть богом", "Стругацкие", 224, 3),
]

libraries = [
    Library(1, "Районная библиотека №5"),
    Library(2, "Государственная библиотека фантастики"),
    Library(3, "Научная библиотека 'Космос'"),
]

booksLibraries = [
    BookLibrary(1, 1),
    BookLibrary(1, 3),
    BookLibrary(2, 2),
    BookLibrary(2, 4),
    BookLibrary(2, 1),
    BookLibrary(3, 5),
    BookLibrary(3, 6),
    BookLibrary(3, 2),
]

def get_one_to_many(books, libraries):
    return [
```

```
        (book.title, book.author, book.pages, lib.name)
        for book in books
        for lib in libraries
        if book.libraryID == lib.id
    ]

def get_many_to_many(books, libraries, booksLibraries):
    many_to_many_temp = [
        (lib.name, bl.libraryID, bl.bookID)
        for lib in libraries
        for bl in booksLibraries
        if lib.id == bl.libraryID
    ]

    return [
        (book.title, lib_name)
        for lib_name, lib_id, book_id in many_to_many_temp
        for book in books
        if book.id == book_id
    ]


def task_1(books, libraries):
    one_to_many = get_one_to_many(books, libraries)
    return sorted(one_to_many, key=lambda x: x[3])

def task_2(books, libraries):
    one_to_many = get_one_to_many(books, libraries)
    result = []
    for lib in libraries:
        books_in_lib = list(filter(lambda x: x[3] == lib.name, one_to_many))
        if len(books_in_lib) > 0:
            total_pages = sum([pages for _, _, pages, _ in books_in_lib])
            result.append((lib.name, total_pages))

    return sorted(result, key=lambda x: x[1], reverse=True)

def task_3(books, libraries, booksLibraries):
    many_to_many = get_many_to_many(books, libraries, booksLibraries)
    result = {}

    for title, lib_name in many_to_many:
        if 'библиотека' in lib_name.lower():
            if lib_name not in result:
                result[lib_name] = []
            result[lib_name].append(title)

    return result

def main():
    print("--- Запрос A1 ---")
    print("Список всех связанных книг и библиотек (1:M), отсортированный по
библиотекам:")
    result_1 = task_1(books, libraries)
    for i in result_1:
        print(f"  Библиотека: {i[3]}, Книга: {i[0]}, Автор: {i[1]}")

    print("\n--- Запрос A2 ---")
    print('Список библиотек с суммарным количеством страниц книг,
отсортированный по сумме (по убыванию):')
    result_2 = task_2(books, libraries)
    for i in result_2:
        print(f"  Библиотека: {i[0]}, Суммарно страниц: {i[1]}")
```

```
    print("\n--- Запрос А3 ---")
    print("Список библиотек, содержащих слово 'библиотека', и их книг
(М:М):")
    result_3 = task_3(books, libraries, booksLibraries)
    for lib_name, book_titles in result_3.items():
        print(f"  Библиотека: {lib_name}")
        for title in book_titles:
            print(f"    - Книга: {title}")

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

test_rk2.py

```
import unittest
import rk2

class TestRK2(unittest.TestCase):

    def setUp(self):
        """Инициализация данных перед каждым тестом"""
        self.books = rk2.books
        self.libraries = rk2.libraries
        self.booksLibraries = rk2.booksLibraries

    def test_task_1(self):
        """Тестирование Задания А1 (Сортировка по библиотекам)"""
        expected_first_lib_name = "Государственная библиотека фантастики"

        result = rk2.task_1(self.books, self.libraries)

        self.assertTrue(len(result) > 0)

        self.assertEqual(result[0][3], expected_first_lib_name)

    def test_task_2(self):
        """Тестирование Задания А2 (Сумма страниц и сортировка)"""


        result = rk2.task_2(self.books, self.libraries)

        self.assertEqual(result[0][0], "Районная библиотека №5")
        self.assertEqual(result[0][1], 584)

        self.assertTrue(result[0][1] >= result[1][1])

    def test_task_3(self):
        """Тестирование Задания А3 (Поиск 'библиотека' и М:М связи)"""
        result = rk2.task_3(self.books, self.libraries, self.booksLibraries)

        self.assertIsInstance(result, dict)

        lib_cosmos = "Научная библиотека 'Космос'"
        self.assertIn(lib_cosmos, result)
        self.assertIn("Дивный новый мир", result[lib_cosmos])

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

Скриншот работы приложения

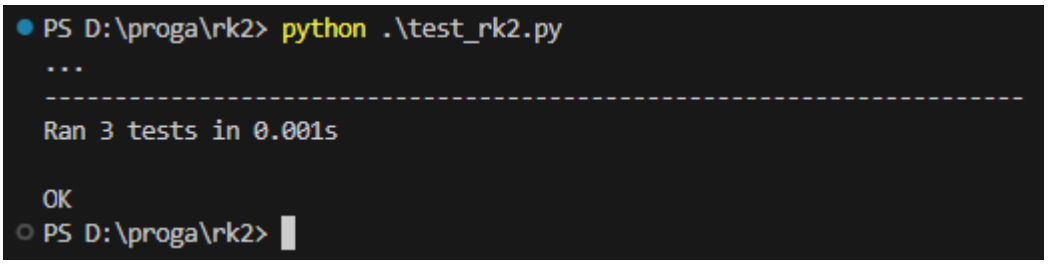

Рисунок 1. Вывод результатов программы