Лабораторная работа №4

Тема: Библиотека с пользовательскими коллекциями и симуляцией
Выполнил: Попков Максим Александрович, М8О-105БВ-25

Цель работы

- Реализовать модель библиотеки с несколькими типами книг.
- Освоить пользовательские коллекции (списковая и словарная).
- Научиться поддерживать согласованность данных между структурами.
- Получить практику работы с логированием, рандомизацией и модульной архитектурой.
- Написать набор автотестов на PyTest для проверки корректности всех модулей.

Ход выполнения

1. Модели книг

В модуле **models.py** реализованы классы:

- **Book** — базовая модель с полями название, автор, год, жанр, ISBN;
- **FictionBook** — художественная книга с возрастным ограничением;
- **ScienceBook** — научная книга с полем области науки.

Все классы выполняют проверку входных данных и имеют осмысленный __repr__, что облегчает анализ логов.

2. Пользовательские коллекции

В **collections.py** реализованы две коллекции:

BookCollection (списковая):

- добавление, удаление по ISBN, проверка наличия;
- поддержка индексации, срезов и итератора;
- автоматическая замена книги при совпадении ISBN.

IndexDict (словарная):

- индексы по isbn, author и year;
- быстрый поиск и синхронное удаление из всех индексов;
- доступ index[isbn], перебор элементов, проверка на вхождение.

Коллекции работают совместно и корректно обрабатывают ошибки.

3. Класс Library

Модуль **library.py** объединяет коллекции и даёт общий интерфейс:

- централизованное хранение книг;
- добавление/удаление книги по объекту или по ISBN;
- поиск по автору, году, жанру и ISBN;

- итерация и получение размера библиотеки.

4. Симуляция работы библиотеки

В модуле **simulation.py** реализована функция
run_simulation(steps, seed), выполняющая псевдослучайную имитацию:

- создаются случайные книги разных типов;
- выполняются действия ADD / REMOVE / FIND_*;
- генерируются как корректные, так и заведомо неверные запросы (например, неправильный ISBN);
- каждое действие записывается в лог через log.py.

Используются списки констант из **constants.py**. Логирование ведётся в shell.log.

5. Точка входа

Файл **main.py** — простой интерфейс: ввод количества шагов ≥ 10 и запуск симуляции.
При ошибочном вводе программа корректно завершается.

6. Тестирование

Для проверки программы написан полноценный набор автотестов (папка **tests/**) с использованием PyTest:

- **test_models.py** — проверка корректной инициализации, валидации и __repr__ классов Book/FictionBook/ScienceBook.
- **test_collections.py** — тестирование логики BookCollection и IndexDict: добавление, удаление, поиск, обработка ошибок, корректность индексации и срезов.
- **test_library.py** — создание библиотеки, операции добавления/удаления и поиск по всем полям.
- **test_simulation.py** — проверка входных параметров симуляции, поведения с seed и факта генерации лог-файла.

Все тесты успешно пройдены.

Результаты

- Реализована полная структура библиотеки с двумя пользовательскими коллекциями.
- Выполнены все требования: согласованность данных, индексация, поиск, удаление, симуляция, логирование.
- Проект модульный, структурированный и расширяемый.
- Добавлен комплект PyTest-тестов, покрывающий ключевые элементы функциональности.
- Симуляция корректно сохраняет ход работы в shell.log.

Вывод

В ходе работы была разработана библиотека с поддержкой различных типов книг, пользовательских коллекций и индексов. Реализована полноценная симуляция с логированием. Проект дополнен набором автотестов, что повысило устойчивость и надёжность решения. Все

цели лабораторной работы выполнены, навыки проектирования модульного кода, обработки ошибок и тестирования закреплены.