library_system.py

```python
from dataclasses import dataclass
from typing import List, Tuple, Optional

@dataclass
class Library:
    id: int
    name: str

@dataclass
class Book:
    id: int
    title: str
    author: str
    pages: int
    library_id: int

@dataclass
class LibraryBook:
    library_id: int
    book_id: int

class LibrarySystem:
    def __init__(self, libraries: List[Library], books: List[Book],
library_books: List[LibraryBook]):
        self.libraries = libraries
        self.books = books
        self.library_books = library_books

    def get_all_libraries(self) -> List[Library]:
        """Получить список всех библиотек"""
        return self.libraries

    def get_all_books(self) -> List[Book]:
        """Получить список всех книг"""
        return self.books

    def get_library_books_relations(self) -> List[LibraryBook]:
        """Получить все связи библиотек с книгами"""
        return self.library_books

    def get_books_ending_with_a(self) -> List[Book]:
        """Найти книги, названия которых заканчиваются на 'А'"""
        return [book for book in self.books if book.title.endswith('а')]

    def get_library_avg_pages(self) -> List[Tuple[str, float, int]]:
        """Рассчитать среднее количество страниц в книгах по библиотекам"""
        result = []
        for library in self.libraries:
            library_books_list = self._get_books_by_library(library.id)
```

```
            if library_books_list:
                total_pages = sum(book.pages for book in library_books_list)
                avg_pages = total_pages / len(library_books_list)
                result.append((library.name, avg_pages, len(library_books_list)))

        # Сортировка по среднему количеству страниц
        return sorted(result, key=lambda x: x[1])

    def get_libraries_starting_with_a_with_books(self) -> List[Tuple[Library,
List[Book]]]:
        """Найти библиотеки с названием на 'A' и их книги"""
        libraries_a = [lib for lib in self.libraries if lib.name.startswith('A')]
        result = []

        for library in libraries_a:
            library_books = self._get_books_by_library(library.id)
            result.append((library, library_books))

        return result

    def _get_books_by_library(self, library_id: int) -> List[Book]:
        """Вспомогательный метод: получить книги по ID библиотеки"""
        library_book_ids = [lb.book_id for lb in self.library_books if
lb.library_id == library_id]
        return [book for book in self.books if book.id in library_book_ids]

    def get_library_by_id(self, library_id: int) -> Optional[Library]:
        """Получить библиотеку по ID"""
        for lib in self.libraries:
            if lib.id == library_id:
                return lib
        return None

    def get_book_by_id(self, book_id: int) -> Optional[Book]:
        """Получить книгу по ID"""
        for book in self.books:
            if book.id == book_id:
                return book
        return None

    def print_all_data(self):
        """Вывести все данные"""
        print("Библиотеки:")
        for lib in self.libraries:
            print(f"  {lib.id}. {lib.name}")

        print("\nКниги:")
        for book in self.books:
            print(f"  {book.id}. '{book.title}' - {book.author} ({book.pages}
стр.)")
```

```
        print("\nСвязи книг с библиотеками:")
        for lb in self.library_books:
            lib = self.get_library_by_id(lb.library_id)
            book = self.get_book_by_id(lb.book_id)
            lib_name = lib.name if lib else "Неизвестно"
            book_title = book.title if book else "Неизвестно"
            print(f"  Библиотека '{lib_name}' -> Книга '{book_title}'")

# Инициализация данных
def create_sample_data() -> LibrarySystem:
    libraries = [
        Library(1, "Академическая библиотека"),
        Library(2, "Абонемент художественной литературы"),
        Library(3, "Городская центральная библиотека"),
        Library(4, "Абонемент научной литературы")
    ]

    books = [
        Book(1, "Война и мир", "Л. Толстой", 1225, 1),
        Book(2, "Анна Каренина", "Л. Толстой", 864, 2),
        Book(3, "Евгения Онегина", "А. Пушкин", 320, 1),
        Book(4, "Мастер и Маргарита", "М. Булгаков", 480, 3),
        Book(5, "Преступление и наказание", "Ф. Достоевский", 672, 2),
        Book(6, "Физика для начинающих", "А. Эйнштейн", 350, 4),
        Book(7, "Химия органических соединений", "Д. Менделеев", 540, 4)
    ]

    library_books = [
        LibraryBook(1, 1),
        LibraryBook(1, 3),
        LibraryBook(2, 2),
        LibraryBook(2, 5),
        LibraryBook(3, 4),
        LibraryBook(4, 6),
        LibraryBook(4, 7),
        LibraryBook(1, 5),
        LibraryBook(2, 1),
    ]

    return LibrarySystem(libraries, books, library_books)

def main():
    """Основная функция для запуска программы"""
    system = create_sample_data()

    system.print_all_data()

    print("\n" + "="*50)
    print("ЗАПРОС 1: Книги, названия которых заканчиваются на 'А'")
    print("="*50)
```

```
    books_ending_with_a = system.get_books_ending_with_a()
    for book in books_ending_with_a:
        library = system.get_library_by_id(book.library_id)
        library_name = library.name if library else "Неизвестная библиотека"
        print(f"  '{book.title}' - {book.author} ({book.pages} стр.)")
        print(f"    Находится в: {library_name}")

    print("\n" + "="*50)
    print("ЗАПРОС 2: Библиотеки со средним количеством страниц в книгах")
    print("="*50)

    library_avg_pages = system.get_library_avg_pages()
    for lib_name, avg_pages, book_count in library_avg_pages:
        print(f"  {lib_name}:")
        print(f"    Среднее количество страниц: {avg_pages:.1f}")
        print(f"    Количество книг: {book_count}")

    print("\n" + "="*50)
    print("ЗАПРОС 3: Библиотеки с названием на 'А' и их книги")
    print("="*50)

    libraries_with_books = system.get_libraries_starting_with_a_with_books()
    for library, books_list in libraries_with_books:
        print(f"\n  Библиотека: {library.name}")
        if books_list:
            for book in books_list:
                print(f"    - '{book.title}' - {book.author} ({book.pages}
стр.)")
        else:
            print("    В этой библиотеке пока нет книг")

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

test_library_system.py

```
import unittest
from library_system import Library, Book, LibraryBook, LibrarySystem,
create_sample_data

class TestLibrarySystem(unittest.TestCase):
    """Тесты для системы управления библиотекой"""

    def setUp(self):
        """Настройка тестовых данных перед каждым тестом"""
        self.system = create_sample_data()

    def test_get_books_ending_with_a(self):
        """Тест 1: Поиск книг, названия которых заканчиваются на 'А'"""
```

```
        # Arrange
        expected_titles = [«Анна Каренина», «Евгения Онегина», «Мастер и
Маргарита»]

        # Act
        books = self.system.get_books_ending_with_a()

        # Assert
        self.assertEqual(len(books), 3, f”Должно быть найдено 3 книги”)

        actual_titles = [book.title for book in books]
        for expected_title in expected_titles:
            self.assertIn(expected_title, actual_titles,
                          f»Книга '{expected_title}' должна быть в результате»)

        # Проверяем, что все найденные книги действительно заканчиваются на 'а'
        for book in books:
            self.assertTrue(book.title.endswith(‘а’),
                            f»Название книги '{book.title}' должно заканчиваться на
'а'»)

    def test_get_library_avg_pages(self):
        «»»Тест 2: Расчет среднего количества страниц по библиотекам»»»
        # Arrange & Act
        library_avg_pages = self.system.get_library_avg_pages()

        # Assert
        self.assertEqual(len(library_avg_pages), 4, “Должно быть 4 библиотеки”)

        # Проверяем правильность расчета для конкретной библиотеки
        library_names = [lib[0] for lib in library_avg_pages]

        # Находим «Академическую библиотеку» и проверяем расчет
        for lib_name, avg_pages, book_count in library_avg_pages:
            if lib_name == «Академическая библиотека»:
                # В Академической библиотеке должны быть книги с ID 1, 3, 5
                # Книга 1: 1225 стр., книга 3: 320 стр., книга 5: 672 стр.
                expected_avg = (1225 + 320 + 672) / 3
                self.assertAlmostEqual(avg_pages, expected_avg, places=1,
                                       msg=f»Неправильное среднее для {lib_name}»)
                self.assertEqual(book_count, 3, f»Неправильное количество книг
для {lib_name}»)
                break

        # Проверяем сортировку по возрастанию среднего количества страниц
        for I in range(len(library_avg_pages) - 1):
            self.assertLessEqual(library_avg_pages[i][1], library_avg_pages[I +
1][1],
                                 «Библиотеки должны быть отсортированы по
возрастанию среднего количества страниц»)
```

```
    def test_get_libraries_starting_with_a_with_books(self):
        «»»Тест 3: Поиск библиотек с названием на 'A' и их книг»»»
        # Arrange & Act
        libraries_with_books =
self.system.get_libraries_starting_with_a_with_books()

        # Assert
        # Должно быть 3 библиотеки, начинающиеся на 'A'
        self.assertEqual(len(libraries_with_books), 3, “Должно быть 3 библиотеки
на ‘A’”)

        expected_library_names = [“Академическая библиотека”,
                                  «Абонемент художественной литературы»,
                                  «Абонемент научной литературы»]

        actual_library_names = [lib[0].name for lib in libraries_with_books]

        for expected_name in expected_library_names:
            self.assertIn(expected_name, actual_library_names,
                          f»Библиотека '{expected_name}' должна быть в
результате»)

        # Проверяем, что у библиотек есть правильные книги
        for library, books_list in libraries_with_books:
            # Проверяем, что название библиотеки начинается с 'A'
            self.assertTrue(library.name.startswith(‘A’),
                            f»Название библиотеки '{library.name}' должно
начинаться с 'A'»)

            # Для Академической библиотеки проверяем количество книг
            if library.name == «Академическая библиотека»:
                self.assertEqual(len(books_list), 3,
                                 f»В Академической библиотеке должно быть 3 книги»)

                # Проверяем наличие конкретных книг
                book_titles = [book.title for book in books_list]
                expected_books = [«Война и мир», «Евгения Онегина», «Преступление
и наказание»]
                for expected_book in expected_books:
                    self.assertIn(expected_book, book_titles,
                                  f»Книга '{expected_book}' должна быть в
Академической библиотеке»)

    def test_get_library_by_id(self):
        «»»Дополнительный тест: Получение библиотеки по ID»»»
        # Arrange
        library_id = 1

        # Act
        library = self.system.get_library_by_id(library_id)
```

```
        # Assert
        self.assertIsNotNone(library, “Библиотека должна быть найдена”)
        self.assertEqual(library.id, library_id, “ID библиотеки должен
совпадать”)
        self.assertEqual(library.name, “Академическая библиотека”,
                         «Название библиотеки должно быть 'Академическая
библиотека'»)

    def test_get_book_by_id(self):
        «»»Дополнительный тест: Получение книги по ID»»»
        # Arrange
        book_id = 1

        # Act
        book = self.system.get_book_by_id(book_id)

        # Assert
        self.assertIsNotNone(book, “Книга должна быть найдена”)
        self.assertEqual(book.id, book_id, “ID книги должен совпадать”)
        self.assertEqual(book.title, “Война и мир”,
                         «Название книги должно быть 'Война и мир'»)

if __name__ == “__main__”:
    unittest.main(verbosity=2)
```

Результаты работы