**Exercise 3: Online Bookstore - Handling Path Variables and Query Parameters**

Business Scenario:

Enhance the book management endpoints to handle dynamic URLs and query parameters.

**Instructions**

**1. Path Variables**

**Objective:** Implement an endpoint to fetch a book by its ID using a path variable.

**Implementation Details:**

* **Endpoint:** GET /books/{id}
* **Description:** Retrieves a specific book based on the provided book ID.
* **Path Variable:** {id} - Represents the unique identifier of the book.

**Path Variables**

**Endpoint:** GET /books/{id}

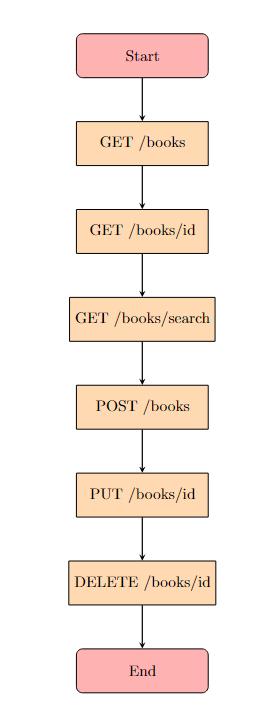
* **Purpose:** Fetch a book by its unique ID.
* **Key Points:**
  + **@GetMapping("/{id}"):** Maps HTTP GET requests with a dynamic path variable to the getBookById method.
  + **@PathVariable Long id:** Extracts the {id} from the URL and passes it to the method. This allows for dynamic fetching of a book based on its ID.

**Query Parameters**

**Endpoint:** GET /books/search

* **Purpose:** Filter books based on query parameters like title and author.
* **Key Points:**
  + **@GetMapping("/search"):** Maps HTTP GET requests to the searchBooks method for searching functionality.
  + **@RequestParam String title, @RequestParam String author:** Binds the title and author query parameters from the URL to the method parameters. This enables filtering of books based on the provided search criteria.

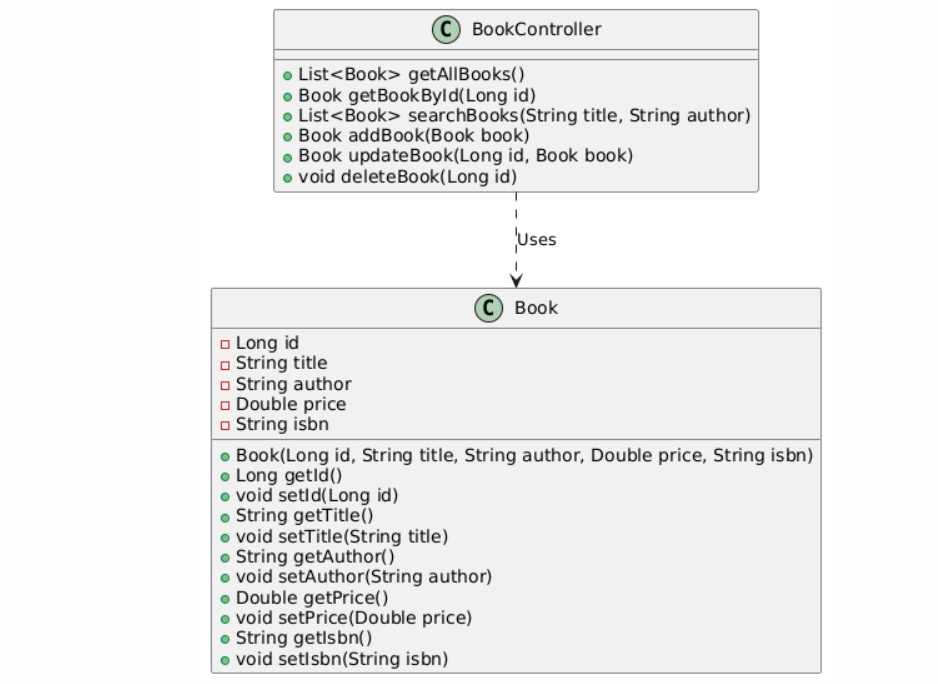
**FLOWCHART :**



**Explanation of Flowchart**

* **Start**: Indicates the beginning of the process.
* **GET /books**: Fetches a list of all books.
* **GET /books/{id}**: Retrieves a specific book by its ID.
* **GET /books/search**: Searches for books based on title and author query parameters.
* **POST /books**: Adds a new book to the collection.
* **PUT /books/{id}**: Updates an existing book with the specified ID.
* **DELETE /books/{id}**: Deletes the book with the specified ID.
* **End**: Indicates the end of the process.

**CLASS DIAGRAM :**



**Explanation :**

**Book**: Represents the book entity with attributes such as id, title, author, price, and isbn. It includes constructors and getters/setters for these attributes.

* **BookController**: Manages the book-related endpoints. It includes methods for:
  + Retrieving all books (getAllBooks)
  + Getting a specific book by ID (getBookById)
  + Searching for books by title and author (searchBooks)
  + Adding a new book (addBook)
  + Updating an existing book (updateBook)
  + Deleting a book (deleteBook)
* **Relationship**: The BookController class uses the Book class to perform operations on book data.