**EXERCISE 6**

**Library Management System**

**1. Understand Search Algorithms:**

**Linear Search**:

* **Algorithm**: Checks each item in a list sequentially until the desired item is found or the list is exhausted.
* **Time Complexity**:
  + **Best Case**: O(1) — The target item is the first item in the list.
  + **Worst Case**: O(n) — The target item is the last item or not in the list.
* **Use Case**: Suitable for small or unsorted lists. Simple and straightforward but less efficient for large data sets.

**Binary Search**:

* **Algorithm**: Efficiently finds the position of a target value by repeatedly dividing the search interval in half. It requires the list to be sorted.
* **Time Complexity**:
  + **Best Case**: O(1) — The target item is at the middle of the list.
  + **Worst Case**: O(log n) — The target item is not in the list but the entire search space has been divided.
* **Use Case**: Ideal for large, sorted lists. More complex than linear search but significantly faster for large data sets.

**2. Setup:**

**Book Class**:

* Represents a book with attributes such as bookId, title, and author. This class provides getters for these attributes and a toString method for easy representation.

**3. Implementation:**

**Linear Search Implementation**:

* Searches through an unsorted list of books sequentially.

**Binary Search Implementation**:

* Searches through a sorted list of books. The list must be sorted by title for binary search to work correctly.

**Main Class**:

* Tests both linear and binary search methods.

**4. Analysis:**

**Time Complexity Comparison**:

* **Linear Search**: O(n) — Scales linearly with the number of books. Simple and effective for small or unsorted datasets but becomes inefficient as the dataset grows.
* **Binary Search**: O(log n) — Scales logarithmically with the number of books, making it much faster for large, sorted datasets. However, it requires the list to be sorted, which can add overhead if the list is not already sorted.

**Use of Each Algorithm**:

* **Linear Search**: Use when dealing with small or unsorted datasets, or when simplicity is preferred over efficiency.
* **Binary Search**: Use for large, sorted datasets where the additional complexity of maintaining a sorted list is justified by the performance gain