EbenLib (Library Management System)

[Group 6]

1. Introduction

The Ebenezer Community Library, located in Ashaiman, serves a wide range of users including students, researchers, and recreational readers. Despite housing thousands of books, the library continues to rely on manual processes for cataloguing and lending, using physical logbooks and handwritten entries. This system has proven to be inefficient, error-prone, and difficult to manage — leading to frequent issues such as lost books, untracked overdue returns, and unreliable inventory visibility.

In response to these challenges, our team was tasked with designing and implementing a **console-based Book Lending and Cataloguing System** that is both **offline-first** and built **entirely from scratch** using core programming constructs. The goal was to develop a solution that modernizes library operations without relying on any external databases, graphical interfaces, or third-party frameworks.

Our system — titled **EbenLib** — was developed entirely in Java using only built-in data structures and file storage mechanisms. It enables librarians to manage books, borrowers, lending activities, and fines, while also supporting advanced features such as report generation, search and sort algorithms, file export/import, and configurable system behavior. It is intended to be used as a lightweight and portable desktop tool that preserves data across sessions and ensures full control over library operations.

2. System Features

The EbenLib system is built as a modular, offline-first console application to support the essential operations of a local library. All features can be accessed via **interactive menus** or by typing **non-interactive commands** directly into the terminal. Key functionalities are described below:

---

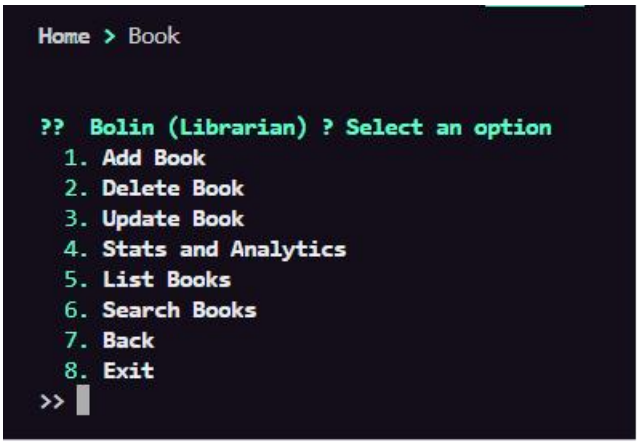
1. Book Catalog Management

The system supports adding, removing, and listing books. Each book record contains:

- **Title**, **Author**, **ISBN**
- **Category**, **Year**, **Publisher**
- **Shelf Location**, **Quantity in stock**

Books are internally stored using a **hash map**, allowing for efficient grouping and filtering. When a book with an existing ISBN is added, its quantity is incremented rather than duplicating the entry.

Non-interactive commands:
book add, book delete, book list, book update, book stats, book search



---

2. Universal Search and Sort Engine

The system includes a **generic search and sort engine** implemented using Java's functional programming capabilities (Comparator<T>, Function<T,R>). These are used across multiple modules, including books, users, and borrowing records.

Search:

- Linear search is used for partial matching (e.g., searching by a keyword).
- Binary search is available for exact lookups in pre-sorted lists.
- Search is case-insensitive and field-agnostic (you can search by title, author, ID, etc., depending on the module).

Sort:

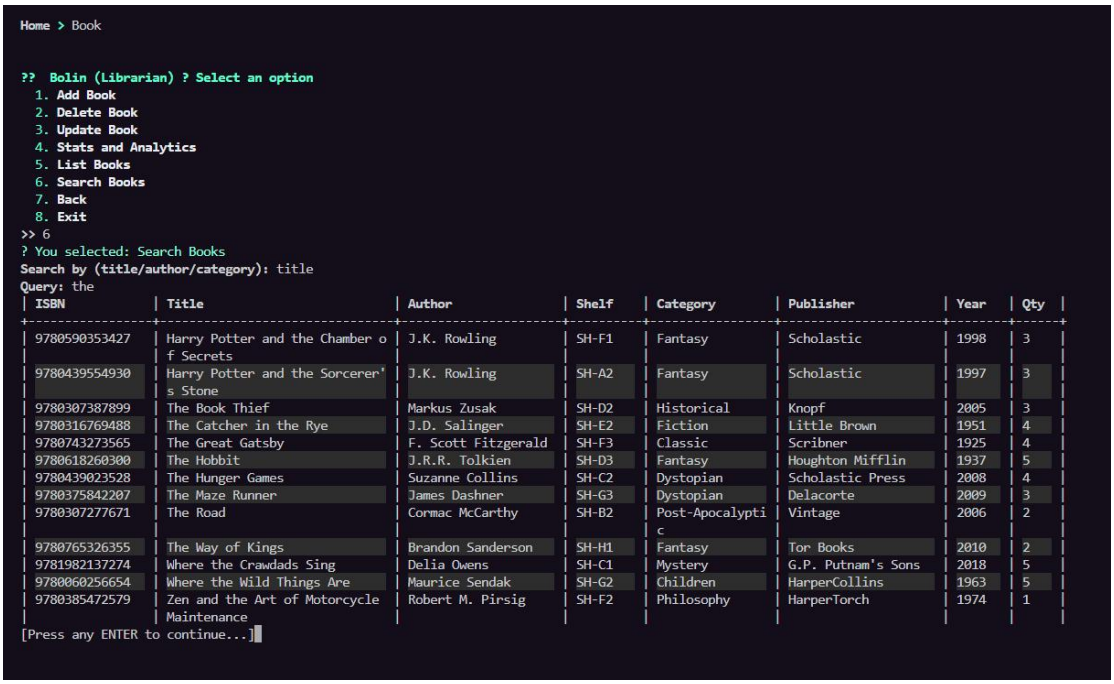
- The system uses a universal **merge sort** implementation to sort lists based on any comparator — such as sorting users by username, books by year, or fines in descending order.
- Sorting is customizable in ascending or descending order.

Search and sort utilities are found in the Searcher and Sorter classes and are reused across:

- BookService

- UserStore
- BorrowStore
- ReportHandler

Used in modules and sections:
Book search, top-borrowed books, top borrowers, fine rankings, report generation, etc.



ISBN	Title	Author	Shelf	Category	Publisher	Year	Qty
9780590353427	Harry Potter and the Chamber of Secrets	J.K. Rowling	SH-F1	Fantasy	Scholastic	1998	3
9780439554930	Harry Potter and the Sorcerer's Stone	J.K. Rowling	SH-A2	Fantasy	Scholastic	1997	3
9780307387899	The Book Thief	Markus Zusak	SH-D2	Historical	Knopf	2005	3
9780316769488	The Catcher in the Rye	J.D. Salinger	SH-E2	Fiction	Little Brown	1951	4
9780743273565	The Great Gatsby	F. Scott Fitzgerald	SH-F3	Classic	Scribner	1925	4
9780618260300	The Hobbit	J.R.R. Tolkien	SH-D3	Fantasy	Houghton Mifflin	1937	5
9780439023528	The Hunger Games	Suzanne Collins	SH-C2	Dystopian	Scholastic Press	2008	4
9780375842207	The Maze Runner	James Dashner	SH-G3	Dystopian	Delacorte	2009	3
9780307277671	The Road	Cormac McCarthy	SH-B2	Post-Apocalyptic	Vintage	2006	2
9780765326355	The Way of Kings	Brandon Sanderson	SH-H1	Fantasy	Tor Books	2010	2
9781982137274	Where the Crawdads Sing	Delia Owens	SH-C1	Mystery	G.P. Putnam's Sons	2018	5
9780060256654	Where the Wild Things Are	Maurice Sendak	SH-G2	Children	HarperCollins	1963	5
9780385472579	Zen and the Art of Motorcycle Maintenance	Robert M. Pirsig	SH-F2	Philosophy	HarperTorch	1974	1

---

3. Borrowing and Lending System

This feature tracks borrowing transactions from request to return. Each borrow record includes:

- **Book ISBN**
- **User ID**
- **Borrow Date**, **Return Date**
- **Status** (PENDING, APPROVED, RETURNED)

Borrow requests are approved only if:

- The book is in stock
- The user is active and has not exceeded the fine threshold

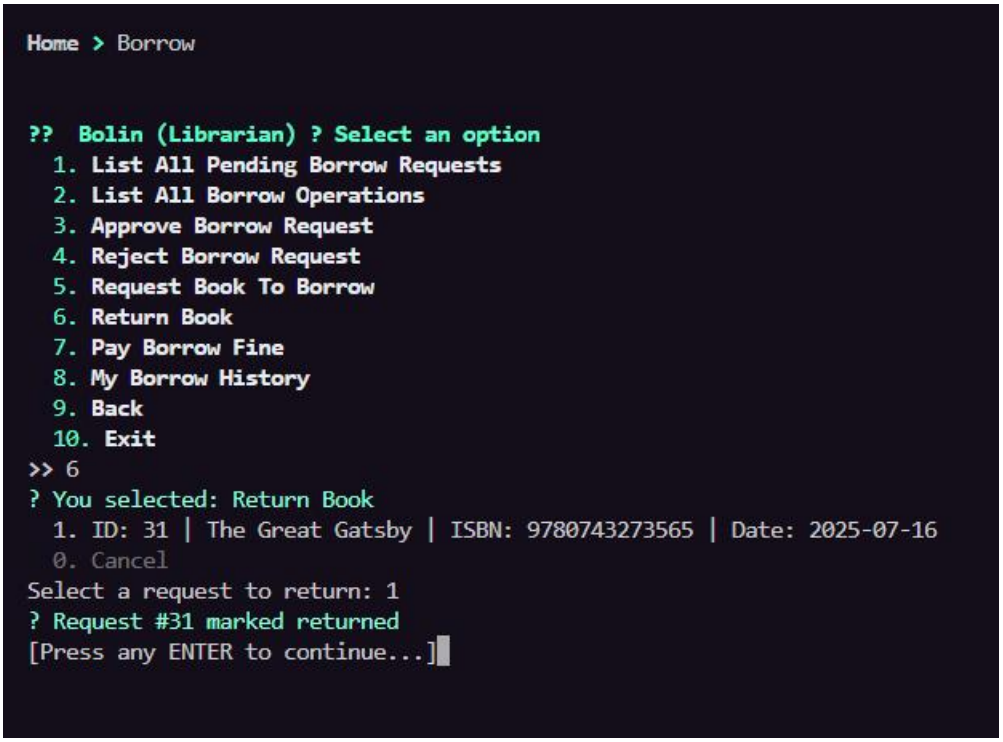
Stock is automatically deducted on approval and replenished upon return.

Non-interactive commands:
borrow request, borrow approve, borrow return, borrow list

```
Home > Borrow


??  Bolin (Librarian) ? Select an option
  1. List All Pending Borrow Requests
  2. List All Borrow Operations
  3. Approve Borrow Request
  4. Reject Borrow Request
  5. Request Book To Borrow
  6. Return Book
  7. Pay Borrow Fine
  8. My Borrow History
  9. Back
  10. Exit
>> 5
? You selected: Request Book To Borrow

Search and request a book
Search book by title (leave blank to list all):  the
  1. Harry Potter and the Chamber of Secrets (ISBN: 9780590353427, Qty: 3)
  2. Harry Potter and the Sorcerer's Stone (ISBN: 9780439554930, Qty: 3)
  3. The Book Thief (ISBN: 9780307387899, Qty: 3)
  4. The Catcher in the Rye (ISBN: 9780316769488, Qty: 4)
  5. The Great Gatsby (ISBN: 9780743273565, Qty: 4)
  6. The Hobbit (ISBN: 9780618260300, Qty: 5)
  7. The Hunger Games (ISBN: 9780439023528, Qty: 4)
  8. The Maze Runner (ISBN: 9780375842207, Qty: 3)
  9. The Road (ISBN: 9780307277671, Qty: 2)
  10. The Way of Kings (ISBN: 9780765326355, Qty: 2)
  11. Where the Crawdads Sing (ISBN: 9781982137274, Qty: 5)
  12. Where the Wild Things Are (ISBN: 9780060256654, Qty: 5)
  13. Zen and the Art of Motorcycle Maintenance (ISBN: 9780385472579, Qty: 1)
  0. Cancel
Select a book to request: 5
? Request submitted (ID=31)
[Press any ENTER to continue...]
```



---

4. Overdue Detection and Fine System

The system uses **date-based logic** to determine overdue books. Once a book surpasses the **loan period**, the system calculates a fine using the configured **fine-per-day** value.

- Overdue fines are stored per user and updated on each run.
- Borrowers exceeding the **block threshold** are prevented from making new requests.
- Admins can reduce fines or reset approve dates to help borrowers.

---

5. User Management

Admins can manage user accounts with features to:

- deactivate/activate accounts
- Promote to librarian or demote to reader

All operations are persisted to file and include validation checks to avoid duplicates.

> **Non-interactive commands**:
> user list, user delete, user deactivate, user promote, user demote, user activate

Interactive path:
Home > User

---

6. Reports and Analytics

The system supports rich reporting features powered by in-memory statistics and sorted data structures. It includes:

1. **Top Borrowed Books** (ranked by frequency)
2. **Top Borrowers** (users with the most requests)
3. **Users with Highest Fines Paid**
4. **Inventory Breakdown** (by category, total stock, low stock)
5. **Borrowing Activity** (approved, pending, returned, overdue)

Reports are generated via static methods in the ReportHandler class, and sorting is handled with the universal sort logic.

Non-interactive commands:
report view, report books, report users, report borrows

Interactive path:
Home > Report - (Full Summary, Users, Books, Borrows)

---

7. System Configuration

The borrowing behavior is configurable via a settings file (settings.txt). The admin can update:

loanPeriod — how many days a book may be borrowed

finePerDay — fine charged per day overdue

blockThreshold — max fine before suspension

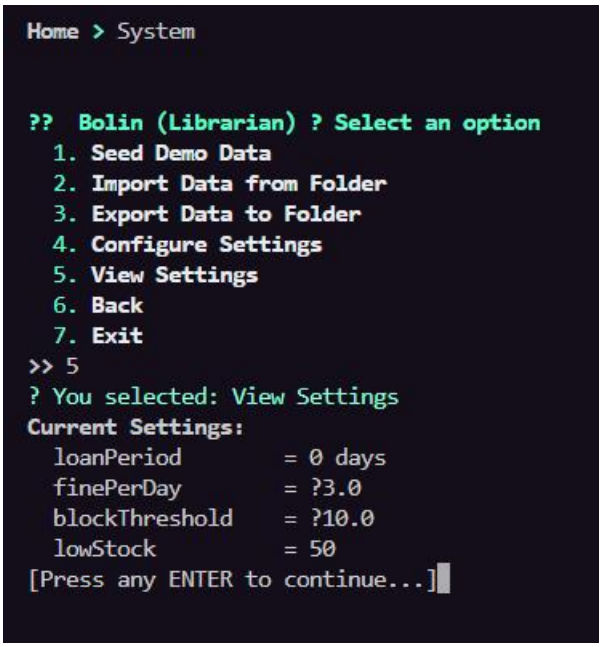
lowStock — stock level below which books are flagged

The config is loaded on startup and persisted on update.

Non-interactive commands:

system config set <loanPeriod | finePerDay | blockThreshold | lowStock> <value>

Interactive path:
Home > System - (Configure Settings)



---

8. Data Persistence, Import, Export, and Seeding

All data is stored in plain text format within a resources/ folder. The app supports:

Automatic saving on every update

Data reload on startup

Export: Dump current state to a folder (excluding session)

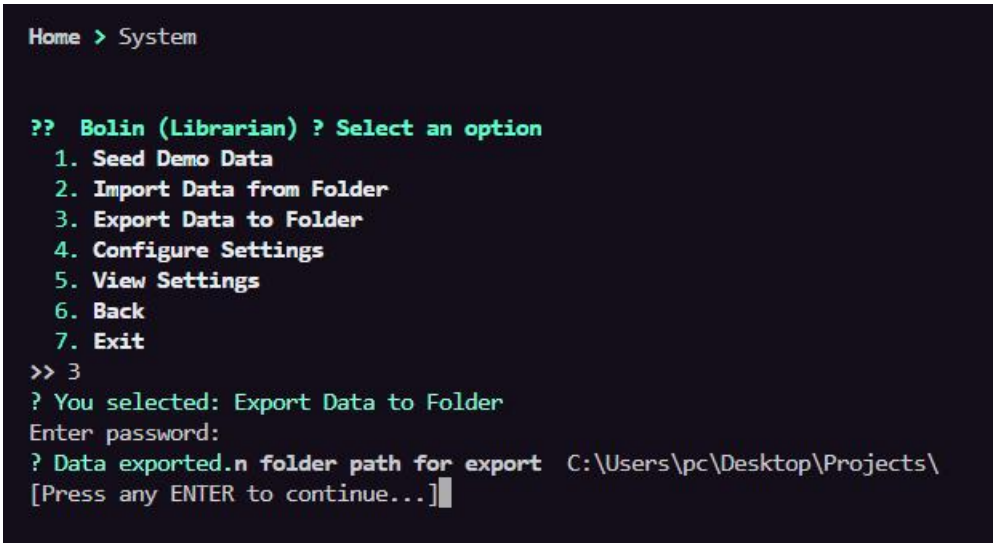
Import: Load from folder (overwrites current data)

Seed: Developer-only feature that populates demo data (only enabled in dev mode)

Session files are excluded from import/export to avoid credential leakage.

Non-interactive commands:
system export --file <path>, system import --file <path>, system seed

Interactive path:
Home > System - (Export Data to Folder, Import Data from Folder, Seed Demo Data)



3. Data Structures Used

This system was intentionally built without external databases or GUI libraries. All data structures were implemented using core Java features and custom implementations under org.ebenlib.ds, with emphasis on correct application of standard DSA techniques in a self-contained library.

3.1 EbenLibList (Custom Growable Array List)

Description: A dynamic, index-based list similar to java.util.ArrayList, built from scratch.

public class EbenLibList<T> implements Iterable<T> { ... }

Where Used:

Primary sequence storage for books, users, borrow records, and buckets in hash-based structures.

Characteristics:

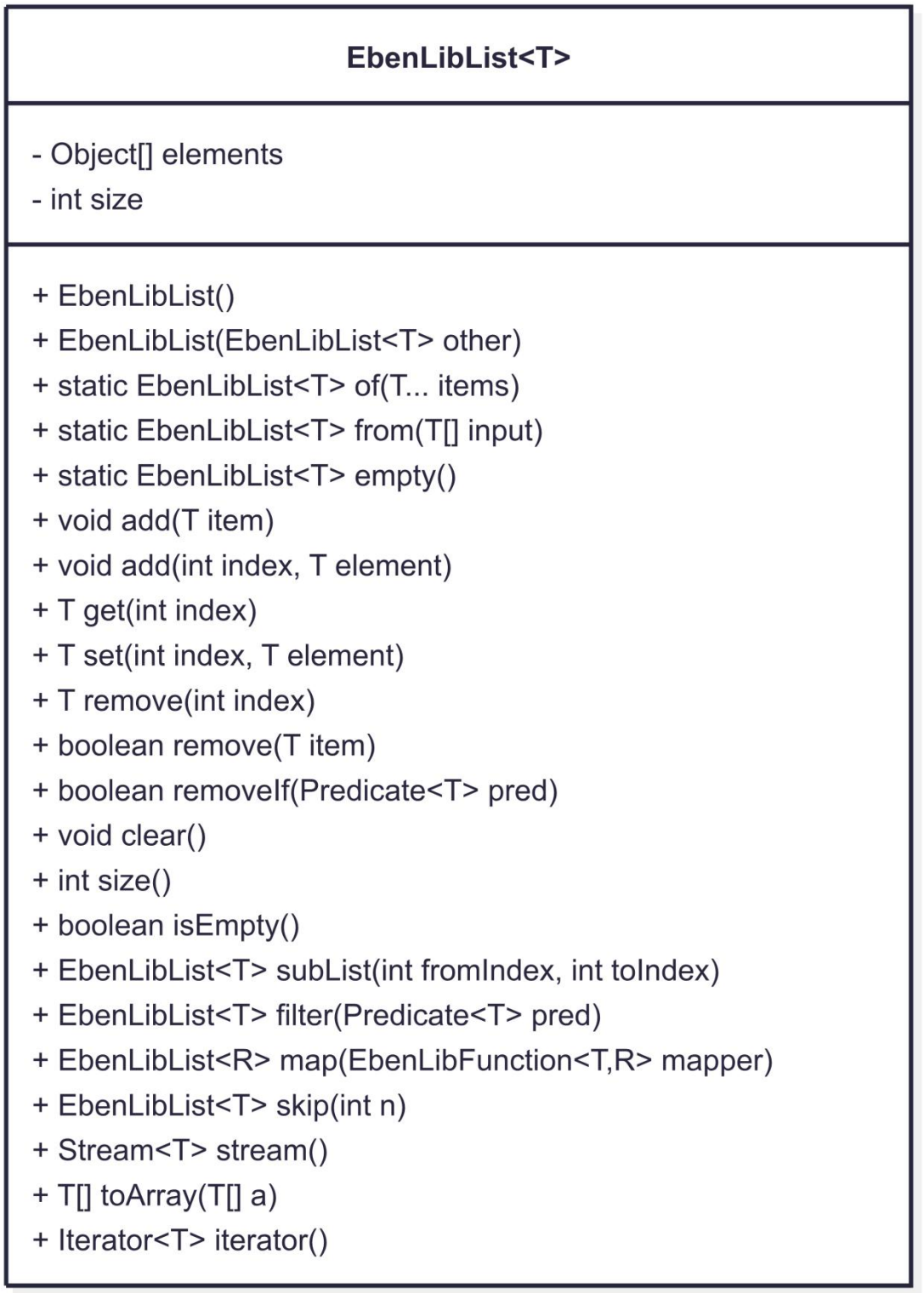
- Backed by an Object[] with initial capacity 10, doubling when full.
- O(1) get/set, O(1) amortized add, O(n) insert/remove due to shifting.
- Utility methods: filter, map, removeIf, skip, stream, subList, toArray, factory methods (of, from, empty).

Benefits:

- Rich API for functional-style operations.
- Cache-friendly contiguous layout.

Trade-offs:

- No thread-safety.
- Manual casts and unchecked operations.



3.2 EbenLibSet & EbenLibHashSet (Custom Set)

Description: A EbenLibSet<T> interface defining a basic set, with EbenLibHashSet<T> implementing a hash table via separate chaining.

```
public interface EbenLibSet<T> { ... }
public class EbenLibHashSet<T> implements EbenLibSet<T> { ... }
```

Where Used:

- Tracking unique items (e.g., keyword indexes, tag lists, active sessions).

Characteristics:

- Buckets: array of `EbenLibList<T>`, initial capacity 16, load factor 0.75.
- O(1) amortized `add`, O(n/b) `remove`/`contains`, O(n) iteration.
- Resizes by doubling capacity and rehashing entries.

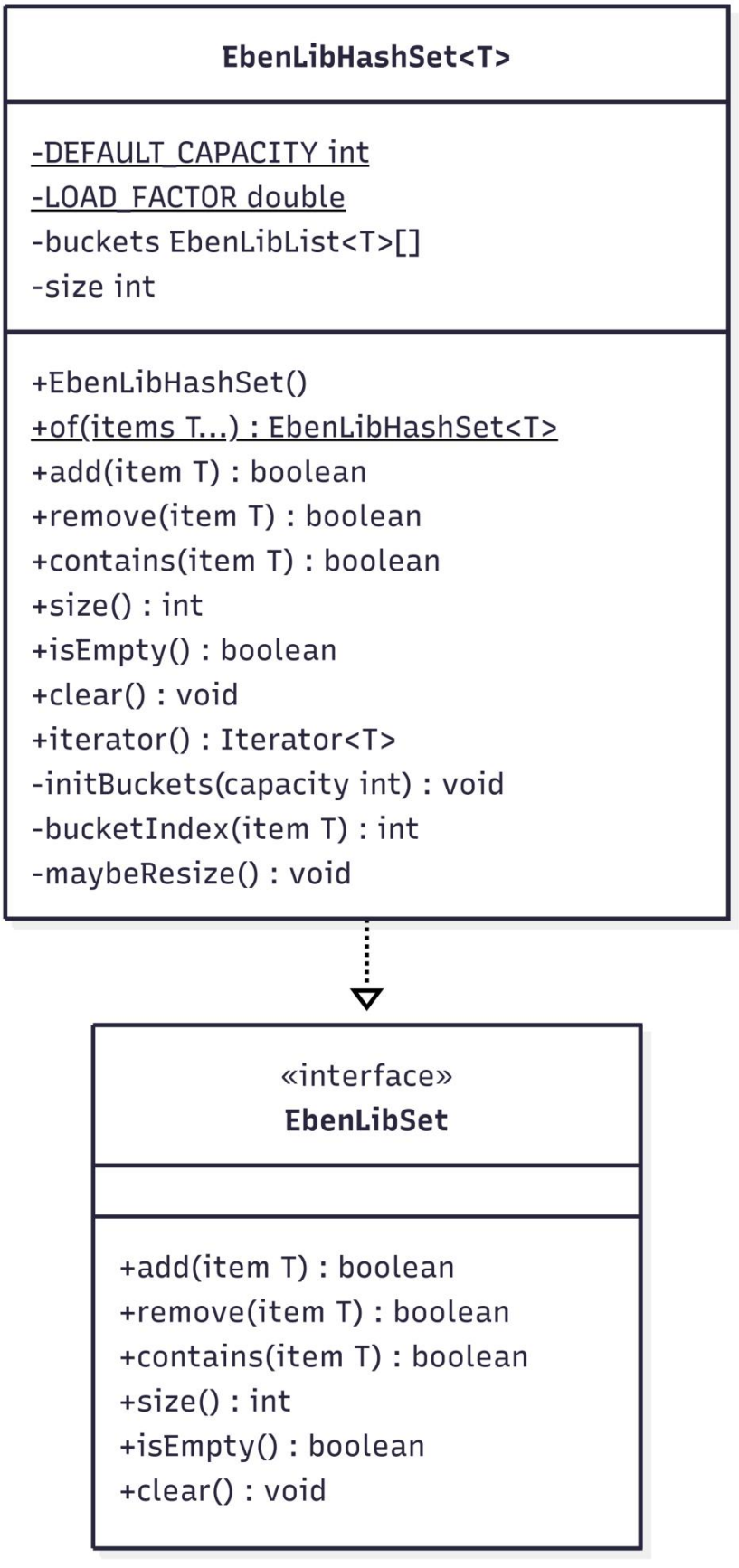
Benefits:

- Custom collision handling and integration with `EbenLibList`.
- Educational clarity on hashing.

Trade-offs:

- Lacks synchronization.
- Dependent on `EbenLibList` performance.

EbenLib Set Implementation



3.3 EbenLibStack (Custom Stack)

Description: A LIFO stack built atop EbenLibList<T>.

public class EbenLibStack<T> { ... }

Where Used:

- Breadcrumb navigation in ConsoleUI (e.g., "Home > System > Settings").

Characteristics:

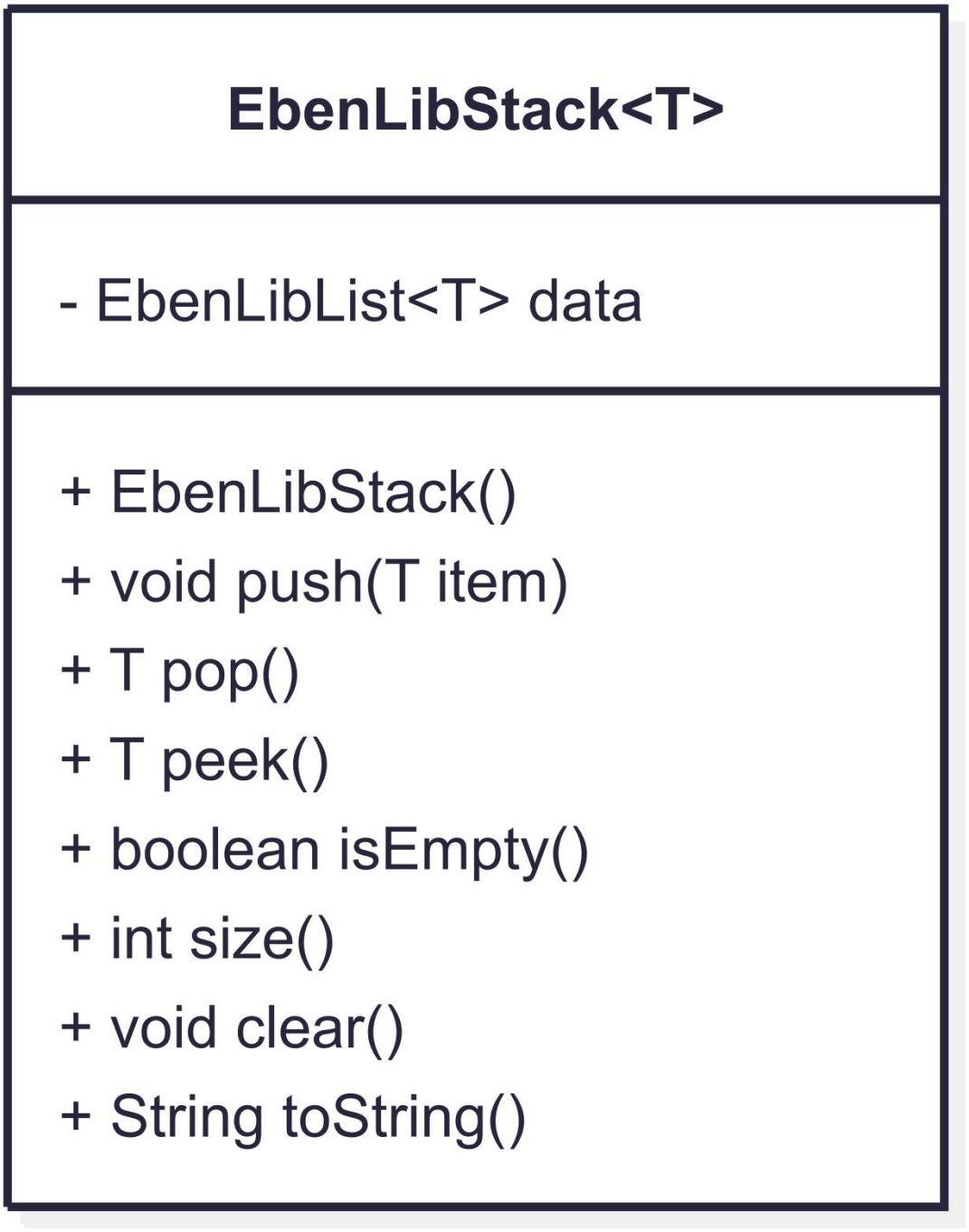
- O(1) push and pop at list end.
- Methods: push, pop, peek, isEmpty, size, clear.

Benefits:

- Simple LIFO abstraction without external dependencies.

Trade-offs:

- No built-in thread-safety.



3.4 EbenLibMap & EbenLibMapEntry (Custom Map)

Description: A hash-table-based map from K to V, with optional insertion-order tracking.

public class EbenLibMap<K, V> implements Iterable<EbenLibMapEntry<K, V>> { ... }
public class EbenLibMapEntry<K, V> { ... }

Where Used:

- Key-value stores such as settings, index maps, and analytic frequency counts.

Characteristics:

- Buckets: array of EbenLibList<EbenLibMapEntry<K, V>>, initial size 16, load factor 0.75.
- Supports put, get, remove, merge, getOrDefault, containsKey, keySet, values, entrySet.
- Optional ordered=true to maintain insertion order via EbenLibList<K>.
- O(1) amortized put/get, O(n/b) remove, O(n) iteration.

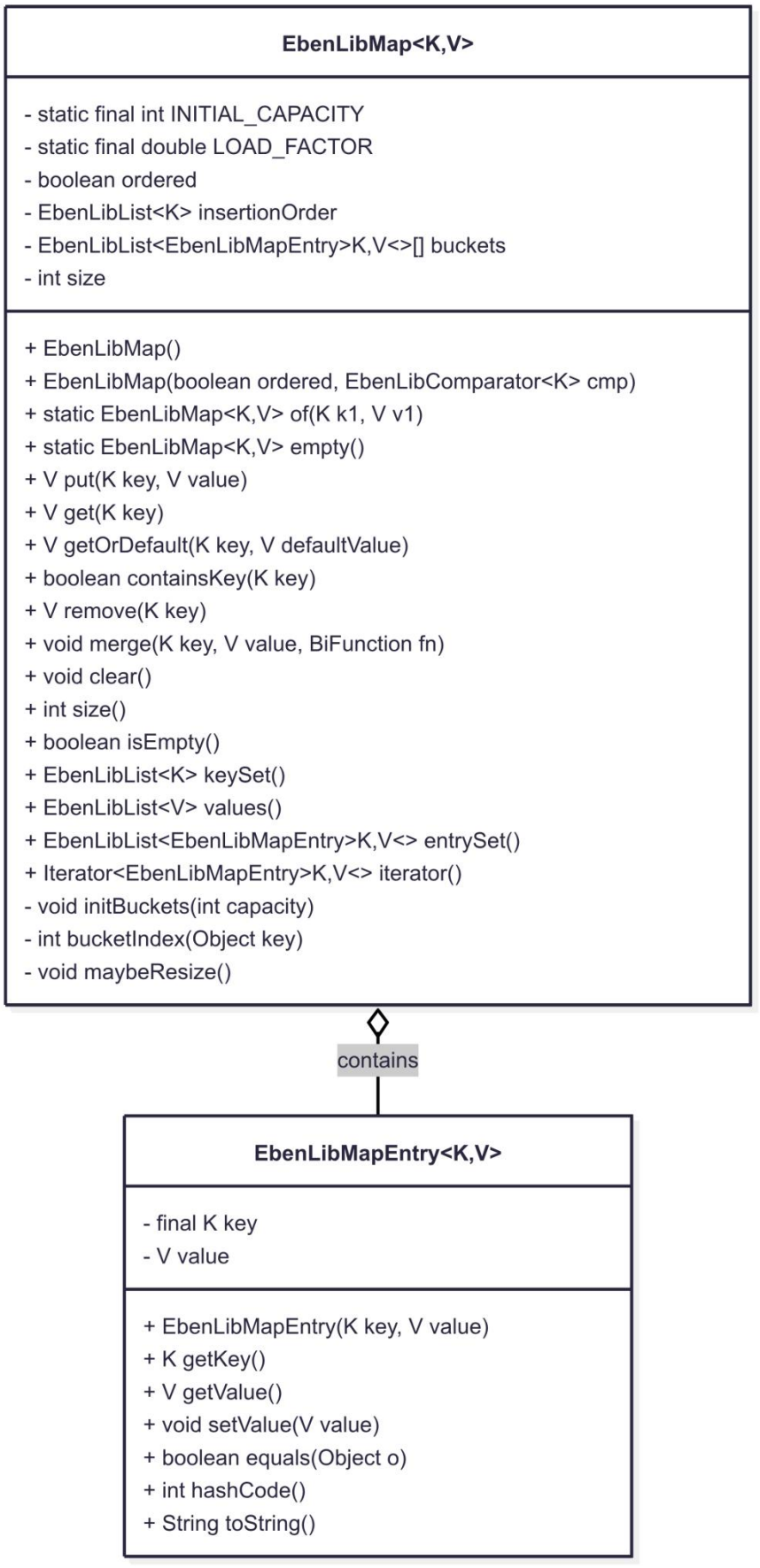
Benefits:

- Flexible map with optional ordering.
- Educational implementation of hashing and collision chains.

Trade-offs:

- No built-in thread-safety.
- Iteration order nondeterministic unless ordered enabled.

EbenLib Map Implementation



3.5 EbenLibPriorityQueue (Custom Min-Heap)

Description: A priority queue implemented as a binary heap over EbenLibList<T>, parameterized by a comparator.

public class EbenLibPriorityQueue<T> { ... }

Where Used:

- Overdue monitoring (min-heap on borrow dates).
- Fine ranking (max-heap via custom comparator).

Characteristics:

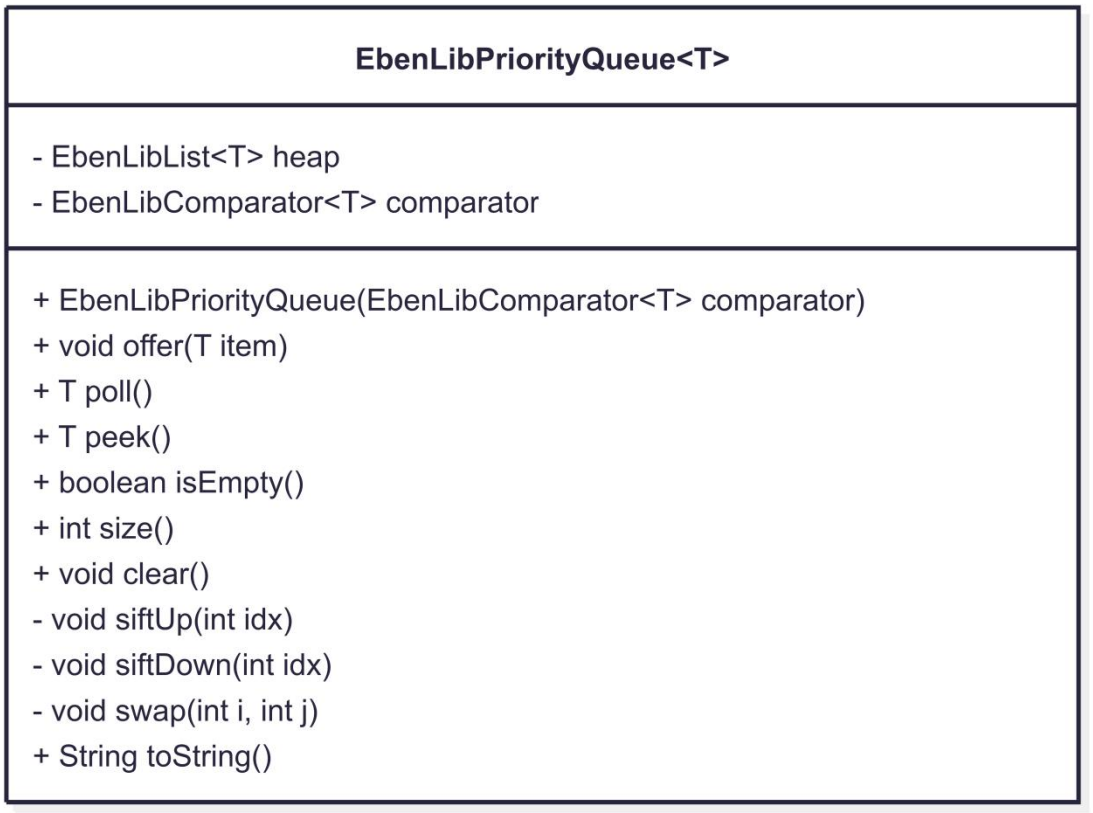
- Backing heap in a list, offer O(log n), poll O(log n), peek O(1).
- Sift-up and sift-down operations maintain heap invariants.

Benefits:

- Efficient top-k queries for time-critical tasks.

Trade-offs:

- Underlying EbenLibList shifting on removal.
- No bulk heapify constructor.



4. Algorithms Used

This system features a few carefully selected algorithms, implemented manually as part of the course requirement. The main categories are:

Binary Search

Where Used:

- Searching for a book or borrow record by ID or ISBN in a pre-sorted list.

Why: Reduces search time from O(n) to O(log n) in sorted lists.

Implementation:

A generic `Searcher.binarySearch(List<T>, T target, Comparator<T>)` was created to allow reuse across different object types.

Trade-offs:

- Requires pre-sorted data.
- Less flexible than linear search in unsorted lists.

Merge Sort

Where Used:

- Sorting books by title, year, or other fields.
- Sorting user or borrow statistics for reporting (e.g., most borrowed books, top borrowers).

Why: Stable, efficient O(n log n) sorting algorithm well-suited to large datasets.

Implementation:

- A generic `Sorter.mergeSort(List<T>, Comparator<T>)` was written to allow sorting of any type.

Trade-offs:

- Uses extra space for intermediate sublists (recursive strategy).
- Not optimal for small or nearly-sorted lists compared to insertion sort.

5. Conclusion

The **Ebenezer Library Book Lending System** meets the core project objectives:

- It's fully offline and file-based.
- It replaces the library's outdated manual tracking with an efficient, user-friendly console system.
- It includes core features like book cataloging, borrow tracking, fine handling, and statistical reporting — all built with relevant data structures and algorithms as per the course requirements.

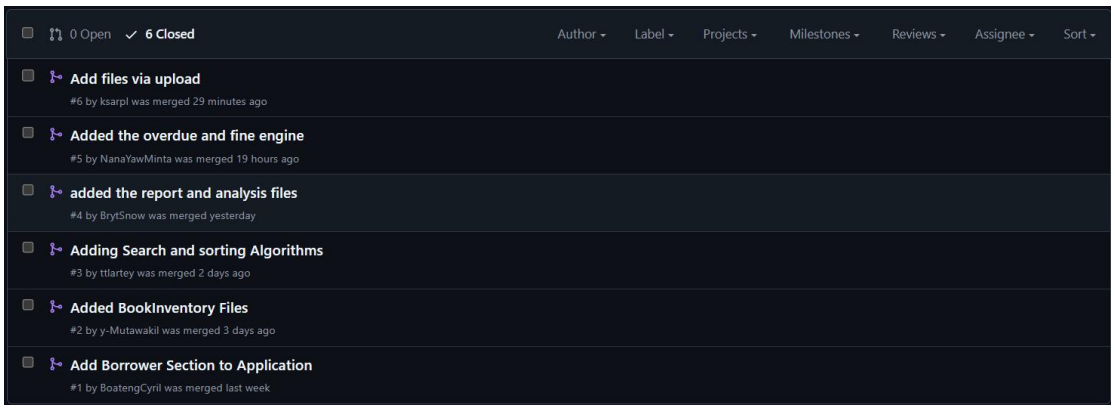
We adopted a modular design with reusable utilities (e.g., search/sort), and designed the system to be extendable for future GUI or network features if needed.

This project solidified our understanding of data structures like hash maps, arrays, and priority queues, as well as the practical implementation of algorithms like binary search and merge sort in real-world scenarios.

---

6. Group Member Contributions

Name	ID	Activities Done	Contribution %	Attendance (X/7)
Mensah Lartey Isaiah Nii Larte	11222100	Core backend, file storage, settings, borrow logic, CLI	16%	7/7
Anti Bright Gyeadu	11340436	Report & Analysis Generator	14%	7/7
Adjei-Gyebi Minta Okatakyie	11040586	Borrow Over Due and Fine logic	14%	6/7
Kofi Sarpei Lartey	11333137	Lending & Return System	13%	6/7
Boateng Cyril Konadu	11124483	Borrower Registry	14%	7/7
Yussif Mutawakil	11252675	Book Inventory & Catalog	15%	7/7
Thomas Tetteh Lartey	11223578	Search and Sort Logic	14%	7/7



Flow Diagram