

Library Management System

Nhu Ngo

CSE 271: Object-oriented Programming

Meisam Amjad

14 April 2023

Table of Contents

I. Introduction -----	pg. 3
II. Requirement and Design -----	pg. 4
• UML Diagram -----	pg. 4
○ Item class -----	pg. 4
○ Book class -----	pg. 5
○ Dvd class -----	pg. 5
○ Librarian class -----	pg. 6
○ Library class -----	pg. 7
○ Borrowable interface class -----	pg. 9
• Text File -----	pg. 11
• HashMap -----	pg. 12
III. Implementation -----	pg. 13
IV. Testing and Results -----	pg. 13
V. References -----	pg. 18

I. INTRODUCTION

The Library Management System is a piece of software that assists libraries in efficiently managing their business operations and serving their customers. It is intended to streamline daily operations and automate several library tasks, including item borrowing, item returning, item status checking, and member administration. The Library Management System's objectives include streamlining and improving library operations, improving user experience, and raising the general efficacy and efficiency of library services. It offers librarians a user-friendly interface for managing and keeping track of both the items that are on hand in the library and those that have been borrowed by customers.

The Library Management System project normally entails the creation of an intuitive desktop or web application that library staff and users can access via a secure login. The system offers a wide range of features and functionalities to meet the requirements of contemporary libraries, such as:

1. **Item Acquisition and Cataloging:** The system makes it simple for library employees to add new items to the collection, update their details, and categorize them according to different standards such title, author, and genre. Additionally, it makes automatic cataloging of books using established library classification schemes, such as Library of Congress or Dewey Decimal, possible.
2. **Staff Management:** The system enables library employees to maintain member records and handle member information, such as member registration and renewal.
3. **Circulation and Inventory Management:** The circulation process is automated by the system, which also updates and checks item status. The library management system maintains precise records of every book, every user, and every borrowing transaction. This makes it easier for librarians to keep track of the volumes that are available.

The main goals of this project are to:

1. **Library Operations:** The goal of the library management system is to make daily operations more efficient. This is accomplished by automating several the manual book management procedures, including book borrowing, returns, and inventory management. The system lowers errors and saves time by giving librarians a productive way to manage their books and customers.
2. **Customer Experience:** By offering a straightforward and user-friendly interface for borrowing and returning books, the library management system is intended to enhance the patron experience. Customers can quickly search for books, put holds on them, and check the status of their accounts with the use of this technology. The system's main goal is to give library staff and users an intuitive interface that is simple to use and explore. Additionally, it improves the overall library services experience, including online book searching.
3. **Library Efficiency:** The library management system improves library efficiency by automating many of the manual procedures involved in book management. With this system in place, librarians may devote more time to other crucial library jobs like book recommendations and patron outreach and less time to administrative duties.
4. **Accurate Database:** The system attempts to keep accurate and current records of books, users, and transactions, lowering the possibility of mistakes and enhancing the dependability of library operations.
5. **Accessibility:** The system seeks to make library services available online, making it simple for users to look up, reserve, and borrow books whenever and wherever they like.

The library management system is, in general, a crucial instrument for contemporary libraries. It makes managing books and users more effective for librarians, improving the borrowing experience for users. Libraries may boost their overall effectiveness, cut down on errors, and save time by utilizing this method.

The Library Management System project's overall goals include modernizing library operations, enhancing user experience, and optimizing library services through automation and technology. As a result, it is a vital tool for contemporary libraries to manage their resources effectively and offer effective services to their clients.

II. REQUIREMENT AND DESIGN:

UML Diagram

1. Item class

- **Purpose:** The "Item" class is a simple class that represents items such as books and DVDs. It has methods to handle borrowing and returning of items, check their availability, and change their type and title. It also implements the "Borrowable" interface, which defines methods related to borrowing and returning items.
- **Instance Variables:**
 - Integer id: Represents the unique identifier for the item.
 - String type: Represents the type of the item (e.g., book, DVD).
 - String title: Represents the title of the item.
 - Boolean borrowed: Represents the status of the item (true if borrowed, false if not).
- **Constructors:**
 - public Item (): A default constructor that initializes the instance variables to default values.
 - public Item (String type, int id, String title, Boolean borrowed): A parameterized constructor that takes in values for type, id, title, and borrowed, and initializes the respective instance variables.
- **Methods:**
 - public int getId(): Returns the id of the item.
 - public String getType(): Returns the type of the item.
 - public String getTitle(): Returns the title of the item.
 - public void changeType(String newType): Changes the type of the item to the given new type.
 - public void changeTitle(String newTitle): Changes the title of the item to the given new title.
 - public boolean getBorrowed(): Returns the status of the item (true if borrowed, false if not).
 - public void checkItem(Object obj): Checks the availability of the item by comparing its borrowed with another boolean flag.
 - public void compareBorrow(Object obj): Checks the availability of the item, if it is available to be borrowed, then updates the borrowing status accordingly.
 - public void compareReturn(Object obj): Checks the availability of the item, if it is unavailable to be borrowed, then updates the borrowing status accordingly.
 - public Item clone(): Creates a deep copy of the item object.

- public String toString(): Overrides the toString() method to return a string representation of the item object.
- public void borrow(): Implements the borrow() method from the Borrowable interface, which prints a message indicating that the item has been borrowed.
- public void returnItem(): Implements the returnItem() method from the Borrowable interface, which prints a message indicating that the item has been returned.
- public void isAvailable(): Implements the isAvailable() method from the Borrowable interface, which prints a message indicating that the item is available to borrow.
- public void isUnavailable(): Implements the isUnavailable() method from the Borrowable interface, which prints a message indicating that the item is unavailable to borrow.
- public boolean equals(Object obj): Overrides the equals() method to compare the id of the item object with another object.
- public void printInfo(): A custom method that prints the summary information of the item.
- The "Item" class implements the "Borrowable" interface, which defines the borrowing and returning methods.
- The "Item" class has a custom "printInfo()" method that prints the title of the item, which is not part of the "Borrowable" interface.
- Access Modifiers: All the methods in the "Item" class are declared as public, which means they can be accessed from anywhere in the program.
- 2. Book class:
 - Purpose: This class represents a Book object, which is a specific type of item. It has instance variables for storing information such as the author and version of the book, and it extends the Item class.
 - Instance Variables:
 - String author: Represents the author of the book.
 - Integer version: Represents the version of the book.
 - Constructors:
 - public Book(String type, int id, String title, boolean borrowed, String author, int version): a parameterized constructor that initializes a Book object with the given values for type, id, title, borrowed, author, and version.
 - public Book(Book book): a constructor that takes an object of the Book class as a parameter and creates a null book object.
 - Methods:
 - public getAuthor(): a method that returns the author of the book as a String.
 - public getVersion(): a method that returns the version of the book as an int.
 - public changeAuthor(String newName): a method that changes the author name of the book to the given newName.
 - public changeVersion(int newVersion): a method that changes the version of the book to the given newVersion.
 - public clone(): a method that creates a deep copy of the Book object.
 - public getType(): a method that returns the type of the item as a String.
 - public toString(): a method that overrides the toString() method of the Object class to print the information of the book.
 - public printInfo(): a method that prints the information of the book.
- 3. DVD class:

- Purpose: This class represents a DVD object, which is a specific type of item. It has instance variables for storing information such as the publisher and duration of the DVD, and it extends the Item class.
 - Instance variables:
 - String publisher: Represents the publisher of the DVD.
 - Integer duration: Represents the duration of the DVD.
 - Constructors:
 - public Dvd(String type, int id, String title, boolean borrowed, String publisher, int duration): This is a parameterized constructor that takes in values for type, id, title, borrowed, publisher, and duration, and initializes the corresponding instance variables.
 - public Dvd(Dvd dvd): This is a constructor that takes in a DVD object.
 - Methods:
 - public String getpublisher(): Returns the publisher of the DVD.
 - public int getDuration(): Returns the duration of the DVD.
 - public void changepublisher(String newName): Changes the publisher of the DVD to a new name.
 - public void changeVersion(int newDuration): Changes the duration of the DVD to a new value.
 - public String getType(): Overrides the getType() method from the Item class and returns the type of the item, which is "dvd" in this case.
 - public Dvd clone(): Overrides the clone() method from the Item class and creates a deep copy of the DVD object.
 - public String toString(): Overrides the toString() method from the Object class and returns a string representation of the DVD object, including the values of its instance variables.
 - public void printInfo(): Overrides the printInfo() method from the Item class and prints the title, publisher, and duration of the DVD.
- Librarian class:

4. Librarian class:

- Purpose: The Librarian class represents a librarian with properties such as ID, type, first name, last name, and area of work. It provides methods to access and modify these properties, as well as methods for object comparison, cloning, and generating summary and string representations of the object.
- Instance Variables:
 - Integer id: Represents the ID of the librarian.
 - String type: Represents the type of librarian.
 - String firstName: Represents the first name of the librarian.
 - String lastName: Represents the last name of the librarian.
 - String area: Represents the area of work for the librarian.
- Constructors:
 - public Librarian(): This is a default constructor that initializes a Librarian object with default values for ID, first name, last name, and area.
 - public Librarian(String type, int id, String firstName, String lastName, String area): This is a parameterized constructor that allows the creation of a Librarian object with specified values for ID, type, first name, last name, and area.
- Methods:

- public getId(): Returns the ID of the librarian.
- public getName(): Returns the full name of the librarian.
- public getArea(): Returns the area of work for the librarian.
- public changeName: Changes the first name and last name of the librarian.
(Parameter variables: String newFirstName, String newLastName)
- public changeArea(String newArea): Changes the area of work for the librarian.
(Parameter variable: String newArea)
- public equals(Object obj): Overrides the equals() method to compare Librarian objects for equality based on their IDs.
- public clone(): Overrides the clone() method to create a deep copy of the Librarian object.
- public printSummary(): Returns a summary string of the librarian's information.
- public toString(): Overrides the toString() method to return a string representation of the Librarian object.

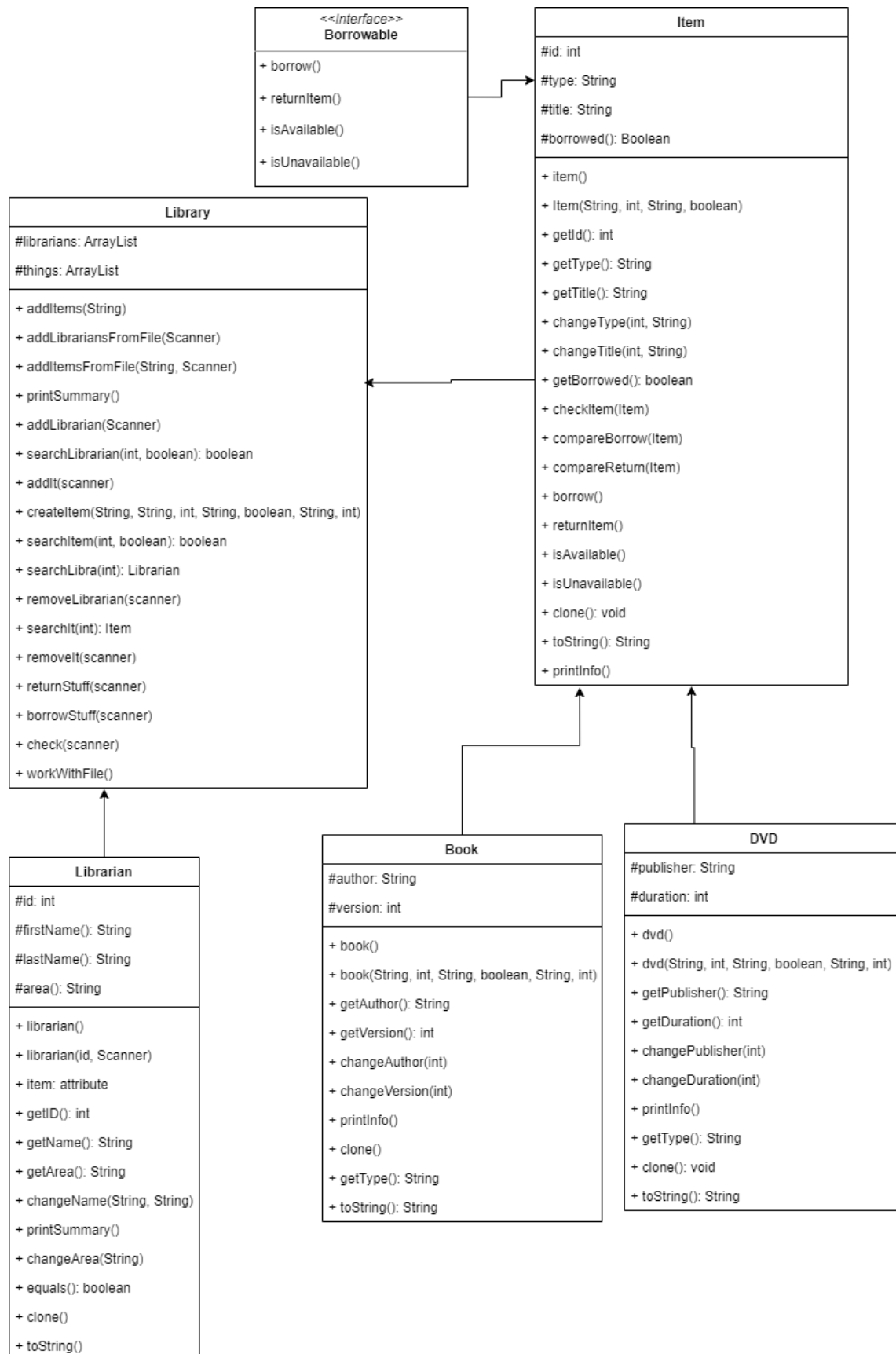
5. Library class

- Purpose: This class represents a library that manages a collection of books and provides various operations on them, such as adding books, removing books, and getting information about books in the library.
- Member Variables:
 - things (ArrayList<Item>): Represents the collection of items in the library as an ArrayList of Item objects. Items are added to this list via the addItem method.
 - librarians (ArrayList<Librarian>): Represents the collection of librarians in the library as an ArrayList of Librarians objects. Librarians are added to this list via the addLibrariansFromFile method.
 - fileName: A String representing the name of the file from which data is loaded to populate the library's items and librarians.
 - inputSuccess: A String used to announce that data has been inputted successfully.
 - removeSuccess: A String used to announce that data has been removed successfully.
- Methods:
 - public addItem(String fileName): This method takes a String fileName as a parameter and is used to load items from a given text file. It uses a Scanner object to read the file, and processes the data using a delimiter ("\t"). It creates Item objects or Librarian objects based on the type of data read, and adds them to the things or librarians ArrayLists respectively.
 - public addItemFromFile(String type, Scanner input): This is a helper method called from the addItem method. It takes a String type and a Scanner input as parameters, and creates and returns an Item object based on the type of data read from the file. The type of item to create is determined by the type parameter.
 - public addLibrariansFromFile(Scanner input): This is a helper method called from the addItem method. It takes a Scanner input as a parameter and creates and returns a Librarian object based on the data read from the file.
 - public printSummary(): This method iterates through the list of librarians and prints a summary of each librarian by calling the printSummary() method in Librarian class on each Librarian object.
 - public addLibrarian(Scanner sc): This method allows the user to input information for a new librarian, checks if the librarian ID already exists in the list, creates a new

Librarian object with the input information, adds it to the list of librarians, and updates the text file.

- public searchLibrarian(int id, Boolean flag) method uses a HashMap to search for a librarian object by ID in the list of librarians. If the flag argument is true, which belongs to the addLibrarian() method, it checks if the ID already exists in the list and returns false if it does (to prompt the user to enter a different ID for adding a new librarian). If the flag argument is false, which belongs to the removeLibrarian(), it returns true if the ID exists in the list (indicating that the librarian can be removed).
 - public addIt(Scanner sc) method allows the user to input information for a new item, checks if the item ID already exists in the list, creates a new Item object with the input information and the specified type, adds it to the list of items, and updates the text file.
 - public createItem(String type, String type2, int id, String title, boolean borrowed, String author, int number): This method creates an object (either a Book or a Dvd object) based on the provided information and adds it to the things array list.
 - public searchItem(int id, boolean flag): This method searches for an item with the given ID in the things array list. The flag parameter is used to determine if the search is for adding or removing a librarian. If the flag is true, which means it belongs to the addIt() method, it indicates that the search is for adding a librarian, and if the flag is false, which belongs to the removeIt() method, it indicates that the search is for removing a librarian. The method returns a boolean value indicating if the item is found in the array list.
 - public searchLibra(int id): This method searches for a librarian with the given ID in the librarians array list and returns the corresponding Librarian object.
 - public removeLibrarian(Scanner sc) throws FileNotFoundException: This method removes a librarian from the librarians array list based on the ID entered by the user. It also updates the test file after removing the librarian.
 - public searchIt(int id): This method searches for an item with the given ID in the things array list and returns the corresponding Item object.
 - public removeIt(Scanner sc) throws FileNotFoundException: This method removes an item from the things array list based on the ID entered by the user. It also updates the test file after removing the item.
 - public workWithFile() throws FileNotFoundException: this method is designed to update information in a text file using a PrintWriter object. It throws a FileNotFoundException if the file specified by fileName is not found.
 - The addItem method uses the fileName parameter to specify the name of the file from which data is loaded to populate the library's items and librarians. It throws a FileNotFoundException if the file is not found. The addItemFromFile and addLibrariansFromFile methods are helper methods that process the data read from the file and create and return corresponding objects (Item or Librarian) based on the data. The things and librarians ArrayLists are used to store the created objects in the library for further use. The inputSuccess and removeSuccess Strings are used to provide feedback to the user about the success of data input or removal.
6. Borrowable interface class:
- Purpose: The "Borrowable" interface defines a set of methods that can be implemented by classes representing borrowable items, such as books or DVDs. The interface provides methods for borrowing, returning, and checking the availability status of an item.

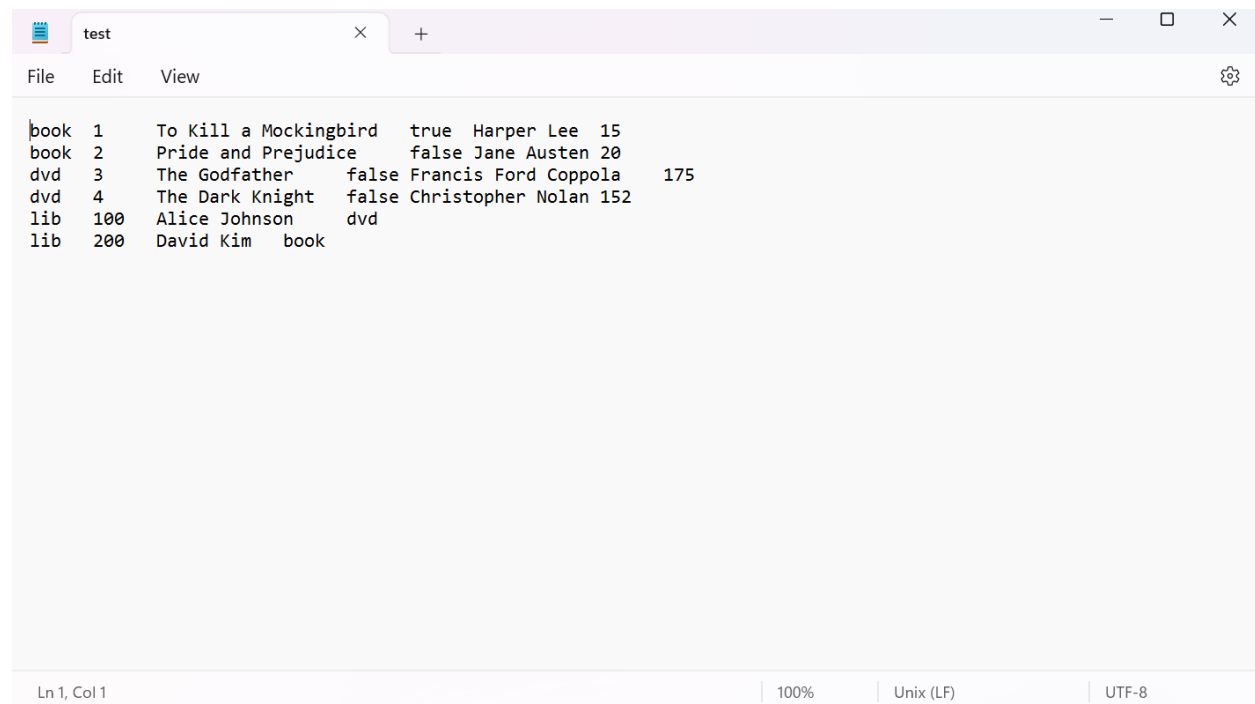
- Methods:
 - `public void borrow()`: Represents the action of borrowing an item. This method should be implemented to handle the borrowing logic of the specific item.
 - `public void returnItem()`: Represents the action of returning an item. This method should be implemented to handle the returning logic of the specific item.
 - `public void isAvailable()`: Represents the action of checking the availability status of an item. This method should be implemented to determine whether the item is currently available for borrowing or not.
 - `public void isUnavailable()`: Represents the action of checking the unavailability status of an item. This method should be implemented to determine whether the item is currently unavailable for borrowing or not.
- Access Modifiers: All the methods in the "Borrowable" interface are declared as public, as interfaces can only have public methods by default.
- The "Borrowable" interface does not define any instance variables or constructors, as interfaces only define method signatures and constants.
- The implementing classes should provide the actual implementation for the methods defined in the "Borrowable" interface according to the specific logic for borrowing, returning, and checking availability of the item.
- The "Borrowable" interface can be implemented by multiple classes, allowing them to have common behavior related to borrowing and returning.



Text File:

This text file contains both the information of items and information of librarians. That means there will be some lines that have 6 data and there will be some lines that only have 5 data. Each data will be separated by “tab.”

- Lines that have 6 data belong to the Items information:
 - The first column is the type of the item (String type).
 - The second column is the ID of the item (int id).
 - The third column is the title of the item (String title).
 - The fourth column is the Boolean flag to show whether this one is available or unavailable. True for unavailable means someone has already borrowed this one, and false is available to be borrowed. (Boolean borrowed).
 - The fifth column is the author’s name or the publisher’s name of the item (String author or String publisher) depends on the object class.
 - The sixth column is the version of the books, or the duration time of the DVD (int version or int duration) depends on the object class.
- Lines that have 4 data belong to the librarian’s information:
 - The first column is the type of the item (String type).
 - The second column is the ID of the item (int id).
 - The third column is the first name of the librarian (String firstName).
 - The fourth column is the last name of the librarian (String lastName).
 - The fifth column is the area that the librarian is responsible for (String area).



```
book 1    To Kill a Mockingbird  true  Harper Lee  15
book 2    Pride and Prejudice   false Jane Austen 20
dvd  3    The Godfather        false Francis Ford Coppola 175
dvd  4    The Dark Knight       false Christopher Nolan 152
lib 100   Alice Johnson    dvd
lib 200   David Kim       book
```

HashMap

In computer programming, a HashMap is a type of data structure that offers a key-value pair mapping as a means of storing and retrieving data. In some other programming languages, it is also known as a dictionary or a hash table. In order to create an index in the underlying array where the appropriate value is kept, the key—which serves as a unique identifier—is hashed.

Key-value pairs can be stored and retrieved quickly and effectively using a data structure called HashMap. Each key's index or bucket is determined by a hash function, and the associated value is then stored in an array.

1. **Hashing:** The hash function generates a hash code for each key when a key-value pair is added to a hash map. A fixed-size integer known as a hash code serves as a numerical representation of the key. The bucket where the value should be kept is determined using the hash code as an index.
2. **Bucket:** Key-value pairs are stored as entries in a bucket, which is a container. As different keys may hash to the same index, each bucket in a hash map can hold numerous items.
3. **Resolution of Collisions:** When two or more keys generate the same hash code or hash to the same bucket, a collision has occurred. HashMap provides built-in collision handling features including independent chaining and open addressing. Each bucket in a distinct chaining has a linked list of elements that hashes to the same index. In open addressing, the entries are kept in the buckets directly, and if there is a collision, the entry is kept in the following available bucket.
4. **Retrieval:** The same hash algorithm is used to recalculate the key's hash code in order to obtain a value from a hash map. The bucket in which the key-value pair might be stored is then identified using the derived hash code. Based on the key, the value is fetched from the bucket.
5. **Insertion and deletion:** When adding a new key-value pair, the entry is placed in the bucket that corresponds to the key's hash code. The entry is resolved using the collision resolution process if a collision takes place. The key is used to generate the hash code and remove the entry from the appropriate bucket when deleting a key-value combination.
6. **Dynamic Resizing:** HashMap automatically resizes themselves to maintain efficient performance when the quantity of key-value pairs rises above a predetermined threshold. Resizing entails rehashing all the key-value pairs into the new array, which has a bigger size.

HashMap is effective for handling huge datasets because their insertion, deletion, and retrieval operations have quick average time complexity, often $O(1)$. However, in cases where hash collisions happen frequently, which results in performance degradation, the worst-case time complexity can be $O(n)$. To guarantee the best possible speed in HashMap operations, care should be made to select a suitable hash function and handle collisions appropriately.

HashMap makes my project searching faster and more convenient. A library management system can employ a HashMap to speed up searching by efficiently storing and retrieving books or other resources based on unique identifiers.

1. **Fast Retrieval:** HashMap computes the key's hash code using a hash function, which is then used to identify the bucket in which the value is kept. This procedure is extremely quick and constant in time regardless of the size of the HashMap because it has an average time complexity of $O(1)$. Due to the ability to retrieve the bucket based on the key instead of having to traverse through the complete list of books or resources, retrieving resources from the HashMap is now incredibly efficient.
2. **Unique Key-Value Pairs:** HashMap demands that keys be distinct, ensuring that every book or resource is connected to a distinct key-value pair in the HashMap. This property speeds up the

library management system's search for a particular book or resource since you may access it directly using its unique key without having to look through a list of items or compare values.

3. Flexible Key-Value Pairs: HashMap supports flexible key-value pairs, where the key can be selected based on the attribute that is most useful for searching. This adaptability enables effective searching using various criteria, and you can quickly move between keys without modifying the HashMap's structure.
4. Scalability: HashMap can effectively handle a high number of books or resources in the library management system because they automatically grow themselves when the number of key-value pairs rises above a particular threshold. By doing this, it is made sure that search results are reliable no matter how big the library gets.
5. Implementation: Implementation is simple since HashMap is a standard feature of many programming languages and libraries, which makes them simple to integrate into a library management system. They offer an easy-to-use method of storing and retrieving key-value pairs, making it simpler to arrange and look for books or resources within the system.

I may dramatically increase the speed of searching and general performance of a library management system by storing and managing books or resources in a HashMap. This makes the system more effective and user-friendly for both library patrons and employees.

III. IMPLEMENTATION

1. [Borrowable Interface class](#)
2. [Item class](#)
3. [Book class](#)
4. [Dvd class](#)
5. [Librarian class](#)
6. [Library class](#)
7. [FrontEnd class](#)

IV. TESTING AND RESULTS

To run this program, we are going to put all the classes into the same place as the test file. We will run the FrontEnd class first, because this is the class to run everything. The first step is to input the name of the text file, which is named "test.txt", and from that it will appear a menu for us to choose.

For the librarian menu, it includes the print summary information of the librarians, add or remove the librarians. To go back to the main menu that you saw first, input -1 and it will bring you back to the main menu.

For the item menu, it includes the add or remove the items, check status, borrow or return the items. To go back to the main menu that you saw first, input -1 and it will bring you back to the main menu.

```
Console x Debug Shell
<terminated> FrontEnd [Java Application] C:\Users\Nhu Ngo\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v
Welcome to the library.
Enter file name to add inventory: test.txt
What would you like to do [0 for menu]: 0
    1. Work with librarians.
    2. Work with items.
    0. Show this menu
   -1. Quit

What would you like to do [0 for menu]: 1
    1. Print summary of things.
    2. To add the new librarian.
    3. To remove the librarian.
    0. Show this menu
   -1. Go back to the main menu

What would you like to do [0 for menu]: 1
Librarian David Kim works at the book section.
Librarian Lisa Berk works at the book section.

    1. Print summary of things.
    2. To add the new librarian.
    3. To remove the librarian.
    0. Show this menu
   -1. Go back to the main menu

What would you like to do [0 for menu]: 2
Please input the ID of the librarian: 121
Please input the first name of the librarian: Emily
Please input the last name of the librarian: Dyer
Please input the area of the librarian works in: Dvd
The data is inputted successfully.

    1. Print summary of things.
    2. To add the new librarian.
    3. To remove the librarian.
    0. Show this menu
   -1. Go back to the main menu
```

After we input the name of the text file, it will appear the first menu for us to choose.

The librarian menu allows us to input to add the new librarian or remove the librarian that exists in both the ArrayList and the text file.

```
Console × Debug Shell
<terminated> FrontEnd [Java Application] C:\Users\Nhu Ngo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20230718

What would you like to do [0 for menu]: 1
Librarian David Kim works at the book section.
Librarian Lisa Berk works at the book section.
Librarian Emily Dyer works at the Dvd section.

1. Print summary of things.
2. To add the new librarian.
3. To remove the librarian.
0. Show this menu
-1. Go back to the main menu

What would you like to do [0 for menu]: 3
Please input the ID of the librarian: 121
The data is removed successfully.

1. Print summary of things.
2. To add the new librarian.
3. To remove the librarian.
0. Show this menu
-1. Go back to the main menu

What would you like to do [0 for menu]: 1
Librarian David Kim works at the book section.
Librarian Lisa Berk works at the book section.

1. Print summary of things.
2. To add the new librarian.
3. To remove the librarian.
0. Show this menu
-1. Go back to the main menu

What would you like to do [0 for menu]: -1
What would you like to do [0 for menu]: 0
1. Work with librarians.
2. Work with items.
0. Show this menu
```

Like the librarian menu, the item menu also allows us to do the same things. It also allows us to see the status of the items and return or borrow the items.

```
Console × Debug Shell
<terminated> FrontEnd [Java Application] C:\Users\Nhu Ngo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v2

What would you like to do [0 for menu]: 2
1. To add items
2. To remove items
3. To borrow an item
4. To return an item
5. To check the availability of an item
0. Show this menu
-1. Go back to the main menu

What would you like to do [0 for menu]: 5
Please input the ID of the item: 1
This item is unavailable to borrow.
1. To add items
2. To remove items
3. To borrow an item
4. To return an item
5. To check the availability of an item
0. Show this menu
-1. Go back to the main menu

What would you like to do [0 for menu]: 4
Please input the ID of the item: 2
This item is available to borrow.
1. To add items
2. To remove items
3. To borrow an item
4. To return an item
5. To check the availability of an item
0. Show this menu
-1. Go back to the main menu

What would you like to do [0 for menu]: 3
Please input the ID of the item: 2
Thank you for borrowing this item.
1. To add items
2. To remove items
```


Console × Debug Shell

<terminated> FrontEnd [Java Application] C:\Users\Nhu Ngo\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.!

What would you like to do [0 for menu]: 4

Please input the ID of the item: 2

This item is available to borrow.

1. To add items
2. To remove items
3. To borrow an item
4. To return an item
5. To check the availability of an item
0. Show this menu
- 1. Go back to the main menu

What would you like to do [0 for menu]: 3

Please input the ID of the item: 2

Thank you for borrowing this item.

1. To add items
2. To remove items
3. To borrow an item
4. To return an item
5. To check the availability of an item
0. Show this menu
- 1. Go back to the main menu

What would you like to do [0 for menu]: 1

Please input the type of the item: Dvd

Please input the ID of the item: 8

Please input the title of the item: The Lord of the Rings: The Fellowship of the Ring

Please input the author/publisher of the item: New Line Home Entertainment

Please input the version/duration of the item: 178

The data is inputted successfully.

1. To add items
2. To remove items
3. To borrow an item
4. To return an item
5. To check the availability of an item
0. Show this menu
- 1. Go back to the main menu

```
Console × Debug Shell
<terminated> FrontEnd [Java Application] C:\Users\Nhu Ngo\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_17.0.5
what would you like to do [0 for menu]: 1
Please input the type of the item: Dvd
Please input the ID of the item: 8
Please input the title of the item: The Lord of the Rings: The Fellowship of the Ring
Please input the author/publisher of the item: New Line Home Entertainment
Please input the version/duration of the item: 178
The data is inputted successfully.
    1. To add items
    2. To remove items
    3. To borrow an item
    4. To return an item
    5. To check the availability of an item
    0. Show this menu
    -1. Go back to the main menu

What would you like to do [0 for menu]: 2
Please input the ID of the item: 2
The data is removed successfully.
    1. To add items
    2. To remove items
    3. To borrow an item
    4. To return an item
    5. To check the availability of an item
    0. Show this menu
    -1. Go back to the main menu

What would you like to do [0 for menu]: -1
What would you like to do [0 for menu]: 0
    1. Work with librarians.
    2. Work with items.
    0. Show this menu
    -1. Quit

What would you like to do [0 for menu]: -1
Thank you for using this system!
```

V. REFERENCES

Class HashMap<K,V>. (n.d.). Retrieved 04 14, 2023, from docs.oracle.com:

<https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html>

geeksforgeeks. (2023, 03 06). *HashMap in Java with Examples*. Retrieved 04 16, 2023, from

geeksforgeeks: <https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/>

w3 schools. (n.d.). *Java HashMap*. Retrieved 04 14, 2023, from w3 schools:

https://www.w3schools.com/java/java_hashmap.asp