Library Management System Using JAVA

Q2: Consider a scenario where you are tasked with developing a simple Library Management System (LMS) in Java. We will start with a sample C++ code snippet and proceed in seven stepsto test your Java skills. Below is a simplified C++ code for managing books in a library:

```
 #include <iostream>
 #include <string> using
 namespace std;

 class Book {
 public:
static int nextId; // Static variable to auto-increment the ID
        int   id; string
        title; string
        author;int year;

        Book(string t, string a, int y) {
            id = nextId++; // Assign the next available ID and increment ittitle = t;
            author = a;
            year = y;
        }

        void display() {
            cout << "ID: " << id << "Title: " << title << " by " << author << "(" << year << ")"
 << endl;
        }
 };

 int Book::nextId = 1; // Initialize the static ID counter

 int main() {
        string title, author;
        int year;
        cout << "Enter the title of the book: ";getline(cin, title);
        cout << "Enter the author of the book: ";getline(cin,
        author);
        cout << "Enter the year of publication of the book: ";cin >> year;
        cin.ignore(); // Ignore the newline character

        Book book(title, author, year);book.display();
        return 0;
 }
```

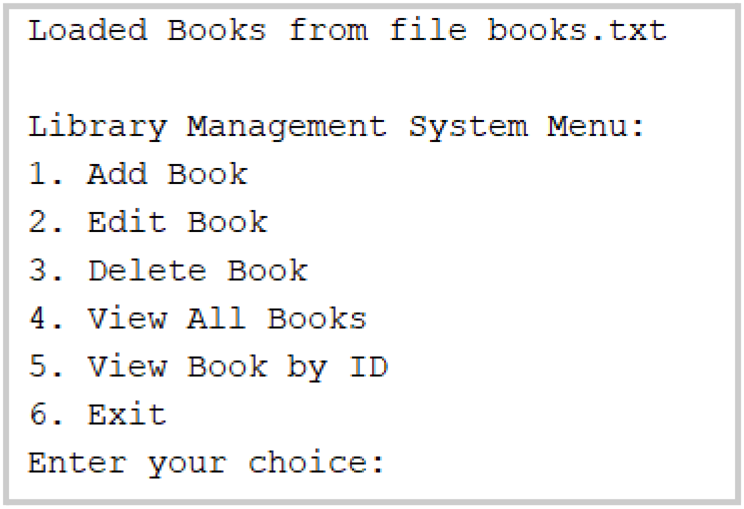
PART-A:

Convert the provided C++ code into Java, creating appropriate classes and methods to maintain the same functionality. Make sure to use proper object-oriented principles. Also ensure that the Java code maintains the ability to input book title, author, and year from the user and display the book's details. The ID of the book is supposed to be automatically incremented one by one. It should not be taken from the user.

PART-B: Add the following features to your Java-based Library Management System:

- Create a new Libraryclass that can store multiple Bookobjects.
- Implement methods to add, edit, delete books in the library and also display all books in the library.
- Make a proper menu for user to be able to add, edit through ID, delete through ID and view all books or a specific book by providing its ID.
- **MUST: Add a method to read data through the provided file and load book details.**

Your menu will look like this after this part:



```
Loaded Books from file books.txt

Library Management System Menu:
1. Add Book
2. Edit Book
3. Delete Book
4. View All Books
5. View Book by ID
6. Exit
Enter your choice:
```

PART-C: Extend your library system by introducing the concept of different types of materials,such as magazines and newspapers. Create an Item superclass that Book, Magazine and Newspaper classes inherit from. Ensure that you can perform the operations implemented inthe previous part on books, magazines and newspapers in the library. Consider the following table for each of the item types:

Book	**Magazine**	**Newspaper**
A book has **an author** & **publishing year** **Type in the file is 1**	A magazine has **multiple list of** **authors** and **a single publisher** **company** **Type in the file is 2**	A newspaper has **a publisher** **company** and **date of its** **publication. DD-MM-YYYY** **Type in the file is 3**

Note: Every item has a title. Also make sure to modify your read from file method to read book, magazine and newspaper as well.

PART-D: Implement a method in your Library class that takes an item (either a book or a magazine or a newspaper) as a parameter and displays its details. Hint: Utilize polymorphism to achieve this.
PART-E: To use more concepts from java, we will create an interface named "Configuration" which will be implemented by Item class. For now, this interface will have a method displayInfo(). Use this method from the interface for each of the child classes of Item.

PART-F: To make your Library Management more fancy, add a new feature to track the popularity of the items in the library based upon borrowing times. For this feature, we will be needing a new class called Borrower. Consider the following requirements for this question:

1. An Item that is currently borrowed from the library cannot be borrowed by any other user.
2. A user cannot borrow the same item twice.

3. Add a popularity count of each item and you must add a new feature called "Hot Picks" in the main menu which will sort the items by their popularity count. Moreover, add a feature to borrow a book from a list of available books.

4. You will also need to modify the item class and keep track of its popularity count in order to implement this feature.

5. Also add an option to display the names of borrowers and which book they are borrowing currently.

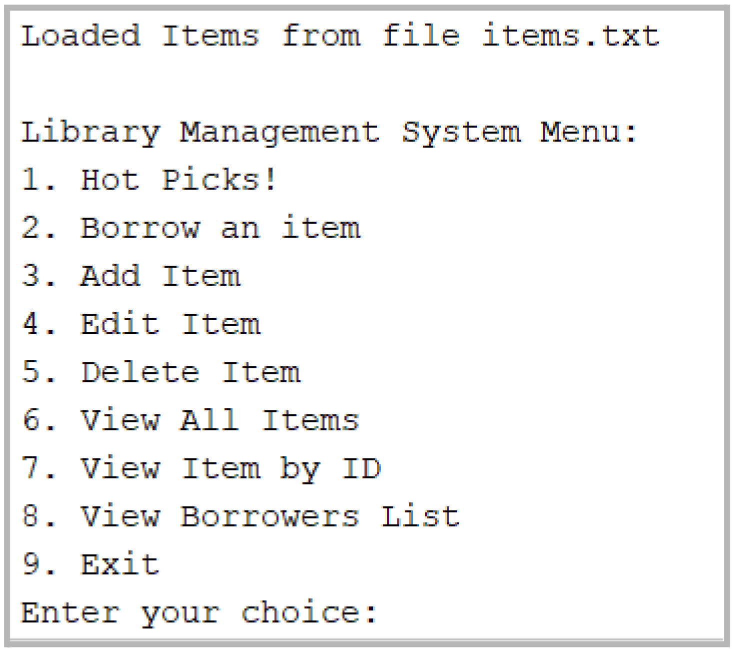
PART-G: This part is very similar to the PART-F, you have to add a new method in the interface which will be used to calculate the cost of borrowing an item from the library, add this functionality in the borrow an item you implemented in the previous part. Use the following table for cost calculation:

Book	**Newspaper**	**Magazine**
Book Cost + 20% *(Cost of Book) + 200 R.s GST	10 R.s + 5 R.s Publisher Charges	Magazine Cost * (Popularity Count)

Note: It is very important on how you store and implement these formulas. Hint: Use finalkeyword, @Override

Show the total bill after borrowing the item with its details.

Your Final Menu should look like this:



```
Loaded Items from file items.txt

Library Management System Menu:
1. Hot Picks!
2. Borrow an item
3. Add Item
4. Edit Item
5. Delete Item
6. View All Items
7. View Item by ID
8. View Borrowers List
9. Exit
Enter your choice:
```

You can use the following UML diagram as an assistance for this assignment

You can modify this UML according to your needs and assumptions.

Use the following file format, name your file as “data.txt”:

item_type, item_title, item_authors/company, item_year if book or newspaper, item_popularity_count, item_price

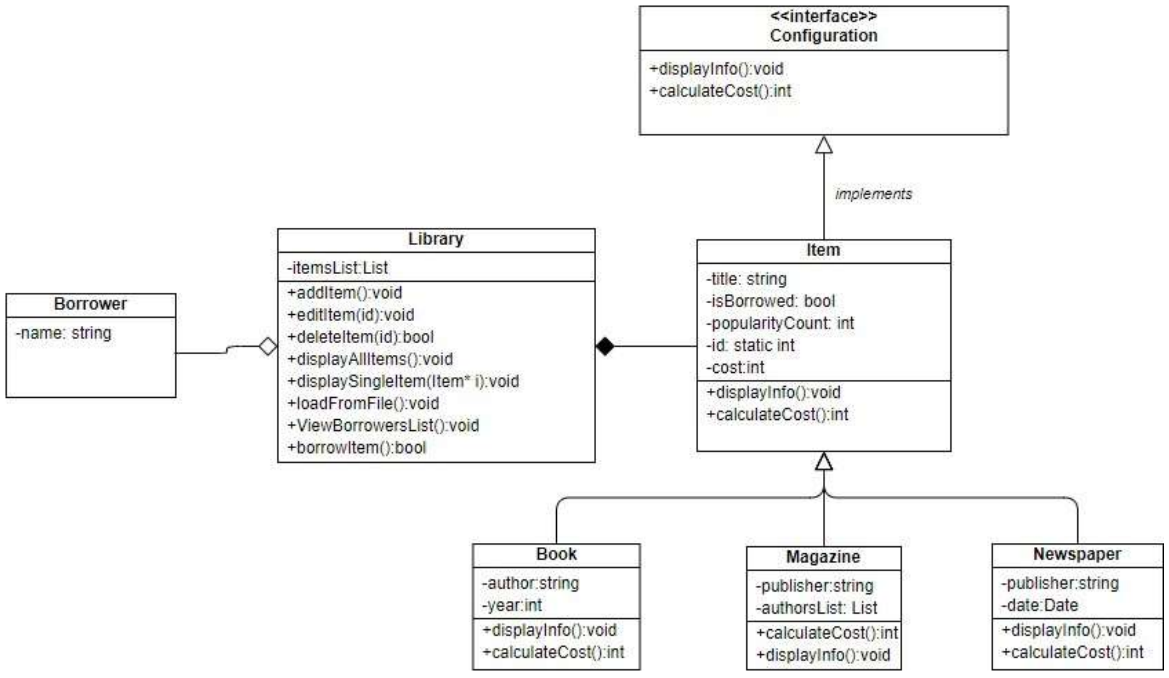
Example:

1, book_title, author_name, 1950, 3, 200

2, magazine_title, author1_name, author2_name, author3_name., publisher_company, 10, 500

3, newspaper_title, publisher_company, 1, 20-09-2023

Note: There is a "." after the last name of author for identifying that all authors are loaded in case of magazines



BEST OF LUCK