**"Management Information System (MIS) in Java utilizing OOPs”**

**Aim**: To create a console-based Management Information System (MIS) in Java utilizing Object-Oriented Programming (OOP) principles.

**Source Code:**

import java.util.*\**;

class Book {

    private int bookId;

    private String title;

    private String author;

    private int copiesAvailable;

    public Book(int bookId, String title, String author, int copiesAvailable) {

*this*.bookId = bookId;

*this*.title = title;

*this*.author = author;

*this*.copiesAvailable = copiesAvailable;

    }

    public int getBookId() {

        return bookId;

    }

    public String getTitle() {

        return title;

    }

    public String getAuthor() {

        return author;

    }

    public int getCopiesAvailable() {

        return copiesAvailable;

    }

    public void borrowBook() {

        if (copiesAvailable > 0) {

            copiesAvailable--;

        }

    }

    public void returnBook() {

        copiesAvailable++;

    }

}

class Student {

    private int studentId;

    private String LCID;

    private List<Book> borrowedBooks;

    public Student(int studentId, String LCID) {

*this*.studentId = studentId;

*this*.LCID = LCID;

*this*.borrowedBooks = new ArrayList<>();

    }

    public int getStudentId() {

        return studentId;

    }

    public String getLCID() {

        return LCID;

    }

    public List<Book> getBorrowedBooks() {

        return borrowedBooks;

    }

    public void borrowBook(Book book) {

        borrowedBooks.add(book);

        book.borrowBook();

    }

    public void returnBook(Book book) {

        borrowedBooks.remove(book);

        book.returnBook();

    }

}

class LibraryDatabase {

    List<Book> books;

    List<Student> students;

    public LibraryDatabase() {

        books = new ArrayList<>();

        students = new ArrayList<>();

    }

    public void addBook(Book book) {

        books.add(book);

    }

    public void addStudent(Student student) {

        students.add(student);

    }

    public Book findBookById(int bookId) {

        return books.stream().filter(b -> b.getBookId() == bookId).findFirst().orElse(null);

    }

    public Student findStudentByLCID(String LCID) {

        return students.stream().filter(s -> LCID.equals(s.getLCID())).findFirst().orElse(null);

    }

}

class ReportGenerator {

    public void generateBookReport(List<Book> books) {

        System.out.println("Book Report:");

        for (Book book : books) {

            System.out.println("Book ID: " + book.getBookId());

            System.out.println("Title: " + book.getTitle());

            System.out.println("Author: " + book.getAuthor());

            System.out.println("Copies Available: " + book.getCopiesAvailable());

            System.out.println("------------------------");

        }

    }

    public void generateStudentReport(List<Student> students, List<Book> books) {

        System.out.println("Student Report:");

        System.out.println("Students who borrowed books:");

        boolean hasStudentsWithBorrowedBooks = false;

        for (Student student : students) {

            if (! student. getBorrowedBooks().isEmpty()) {

                System.out.println("LCID: " + student.getLCID());

                System.out.println("Borrowed Books:");

                for (Book book : student.getBorrowedBooks()) {

                    System.out.println(" - " + book.getTitle());

                }

                System.out.println("------------------------");

                hasStudentsWithBorrowedBooks = true;

            }

        }

        if (!hasStudentsWithBorrowedBooks) {

            System.out.println(" - None");

        }

        System.out.println("Students who haven't borrowed any books:");

        boolean hasStudentsWithoutBorrowedBooks = false;

        for (Student student : students) {

            if (student.getBorrowedBooks().isEmpty()) {

                System.out.println("LCID: " + student.getLCID());

                hasStudentsWithoutBorrowedBooks = true;

            }

        }

        if (!hasStudentsWithoutBorrowedBooks) {

            System.out.println(" - None");

        }

    }

}

public class MISMenu {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int choice;

        LibraryDatabase database = new LibraryDatabase();

        ReportGenerator reportGenerator = new ReportGenerator();

        String projectTitle = "Library Management System";

        for (int i = 1000; i <= 2500; i++) {

            String LCID = "LC0001700" + i;

            Student student = new Student(i, LCID);

            database.addStudent(student);

        }

        do {

            clearConsole();

            System.out.println("==============================");

            System.out.println("Welcome to " + projectTitle + " MIS");

            System.out.println("==============================");

            System.out.println("1. Add Book");

            System.out.println("2. Borrow Book");

            System.out.println("3. Return Book");

            System.out.println("4. Generate Book Report");

            System.out.println("5. Generate Student Report");

            System.out.println("6. Exit");

            System.out.println("==============================");

            System.out.print("Please enter your choice (1-6): ");

            if (scanner.hasNextInt()) {

                choice = scanner.nextInt();

                scanner.nextLine();

            } else {

                choice = -1;

                scanner.nextLine();

            }

            switch (choice) {

                case 1:

                    addBook(database, scanner);

                    break;

                case 2:

                    borrowBook(database, scanner);

                    break;

                case 3:

                    returnBook(database, scanner);

                    break;

                case 4:

                    reportGenerator.generateBookReport(database.books);

                    break;

                case 5:

                    reportGenerator.generateStudentReport(database.students, database.books);

                    break;

                case 6:

                    System.out.println("Exiting the MIS.");

                    break;

                default:

                    System.out.println("Invalid choice. Please choose 1-6.");

            }

            pauseForUserInput(scanner);

        } while (choice != 6);

        scanner.close();

    }

    private static void clearConsole() {

        System.out.print("\033[H\033[2J");

        System.out.flush();

    }

    private static void pauseForUserInput(Scanner scanner) {

        System.out.println("\nPress Enter to continue...");

        scanner.nextLine();

        scanner.nextLine();

    }

    private static void addBook(LibraryDatabase database, Scanner scanner) {

        System.out.print("Enter Book ID: ");

        int

bookId = scanner.nextInt();

        scanner.nextLine();

        System.out.print("Enter Title: ");

        String title = scanner.nextLine();

        System.out.print("Enter Author: ");

        String author = scanner.nextLine();

        System.out.print("Enter Copies Available: ");

        int copiesAvailable = scanner.nextInt();

        Book book = new Book(bookId, title, author, copiesAvailable);

        database.addBook(book);

        System.out.println("Book added successfully.");

    }

    private static void borrowBook(LibraryDatabase database, Scanner scanner) {

        System.out.print("Enter Student LCID: ");

        String studentLCID = scanner.next();

        Student student = database.findStudentByLCID(studentLCID);

        if (student != null) {

            System.out.println("Available Books:");

            for (Book book : database.books) {

                System.out.println("Book ID: " + book.getBookId() + ", Title: " + book.getTitle());

            }

            System.out.print("Enter Book ID to borrow: ");

            int bookId = scanner.nextInt();

            Book book = database.findBookById(bookId);

            if (book != null) {

                if (book.getCopiesAvailable() > 0) {

                    student.borrowBook(book);

                    System.out.println("Book borrowed successfully.");

                } else {

                    System.out.println("No copies of the book are available.");

                }

            } else {

                System.out.println("Book not found.");

            }

        } else {

            System.out.println("Student not found.");

        }

    }

    private static void returnBook(LibraryDatabase database, Scanner scanner) {

        System.out.print("Enter Student LCID: ");

        String studentLCID = scanner.next();

        Student student = database.findStudentByLCID(studentLCID);

        if (student != null) {

            System.out.print("Enter Book ID to return: ");

            int bookId = scanner.nextInt();

            Book book = database.findBookById(bookId);

            if (book != null) {

                if (student.getBorrowedBooks().contains(book)) {

                    student.returnBook(book);

                    System.out.println("Book returned successfully.");

                } else {

                    System.out.println("You haven't borrowed this book.");

                }

            } else {

                System.out.println("Book not found.");

            }

        } else {

            System.out.println("Student not found.");

        }

    }

}

**Output:**



