import java.io.\*;

import java.util.\*;

import java.util.concurrent.\*;

class Book {

private String title, author, ISBN;

public Book(String title, String author, String ISBN) {

this.title = title;

this.author = author;

this.ISBN = ISBN;

}

public String getTitle() { return title; }

public String getAuthor() { return author; }

public String getISBN() { return ISBN; }

@Override

public String toString() {

return "Title: " + title + ", Author: " + author + ", ISBN: " + ISBN;

}

}

class User {

private String name, userID;

private List<Book> borrowedBooks = new ArrayList<>();

public User(String name, String userID) {

this.name = name;

this.userID = userID;

}

public String getName() { return name; }

public String getUserID() { return userID; }

public List<Book> getBorrowedBooks() { return borrowedBooks; }

public void borrowBook(Book book) { borrowedBooks.add(book); }

public void returnBook(Book book) { borrowedBooks.remove(book); }

@Override

public String toString() {

return "User: " + name + " (ID: " + userID + ")";

}

}

interface ILibrary {

void borrowBook(String ISBN, String userID) throws Exception;

void returnBook(String ISBN, String userID) throws Exception;

void reserveBook(String ISBN, String userID) throws Exception;

Book searchBook(String title);

}

abstract class LibrarySystem implements ILibrary {

protected List<Book> books = new ArrayList<>();

protected List<User> users = new ArrayList<>();

protected ConcurrentHashMap<String, Boolean> reservations = new ConcurrentHashMap<>();

public abstract void addBook(Book book);

public abstract void addUser(User user);

}

class LibraryManager extends LibrarySystem {

private final int MAX\_BOOKS\_ALLOWED = 3;

private final Object lock = new Object();

public void addBook(Book book) { books.add(book); }

public void addUser(User user) { users.add(user); }

public Book searchBook(String title) {

return books.stream().filter(b -> b.getTitle().equalsIgnoreCase(title)).findFirst().orElse(null);

}

public void borrowBook(String ISBN, String userID) throws Exception {

synchronized (lock) {

User user = users.stream().filter(u -> u.getUserID().equals(userID)).findFirst().orElseThrow(() -> new Exception("User not found"));

if (user.getBorrowedBooks().size() >= MAX\_BOOKS\_ALLOWED) throw new Exception("Max books borrowed");

Book book = books.stream().filter(b -> b.getISBN().equals(ISBN)).findFirst().orElseThrow(() -> new Exception("Book not found"));

user.borrowBook(book);

System.out.println(user.getName() + " borrowed " + book.getTitle());

}

}

public void returnBook(String ISBN, String userID) throws Exception {

synchronized (lock) {

User user = users.stream().filter(u -> u.getUserID().equals(userID)).findFirst().orElseThrow(() -> new Exception("User not found"));

Book book = user.getBorrowedBooks().stream().filter(b -> b.getISBN().equals(ISBN)).findFirst().orElseThrow(() -> new Exception("Book not borrowed by user"));

user.returnBook(book);

System.out.println(user.getName() + " returned " + book.getTitle());

}

}

public void reserveBook(String ISBN, String userID) throws Exception {

synchronized (lock) {

if (reservations.putIfAbsent(ISBN, true) == null) {

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

} else {

throw new Exception("Book already reserved");

}

}

}

}

public class LibraryManagement {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

LibraryManager library = new LibraryManager();

while (true) {

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

System.out.println("2. Add User");

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

System.out.println("4. Return Book");

System.out.println("5. Reserve Book");

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

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

int choice = scanner.nextInt();

scanner.nextLine();

try {

switch (choice) {

case 1:

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

String title = scanner.nextLine();

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

String author = scanner.nextLine();

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

String ISBN = scanner.nextLine();

library.addBook(new Book(title, author, ISBN));

break;

case 2:

System.out.print("Enter User Name: ");

String name = scanner.nextLine();

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

String userID = scanner.nextLine();

library.addUser(new User(name, userID));

break;

case 3:

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

String borrowUser = scanner.nextLine();

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

String borrowISBN = scanner.nextLine();

library.borrowBook(borrowISBN, borrowUser);

break;

case 4:

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

String returnUser = scanner.nextLine();

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

String returnISBN = scanner.nextLine();

library.returnBook(returnISBN, returnUser);

break;

case 5:

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

String reserveUser = scanner.nextLine();

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

String reserveISBN = scanner.nextLine();

library.reserveBook(reserveISBN, reserveUser);

break;

case 6:

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

scanner.close();

return;

default:

System.out.println("Invalid choice. Try again.");

}

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

}

}