PARTE 2 PARCIAL- ESTRUCTURA DE DATOS

<https://drive.google.com/file/d/1I8ML-T2l_PlYA33CtlS7YD9PwrZbxCar/view?usp=drive_link>  
  
  
 CODIGO  
  
import java.util.Scanner;  
  
public class BibliotecaU1 {  
  
 // =======================  
 // CLASES INTERNAS  
 // =======================  
  
 // Representa un libro en el catálogo  
 static class Libro {  
 int codigo;  
 String titulo;  
 String autor;  
 int stock;  
 boolean activo; // true = disponible, false = dado de baja  
  
 public Libro(int codigo, String titulo, String autor, int stock) {  
 codigo = codigo;  
 titulo = titulo;  
 autor = autor;  
 stock = stock;  
 activo = true;  
 }  
 }  
  
 // Nodo para la lista enlazada simple de préstamos  
 static class Prestamo {  
 int codigoLibro;  
 String usuario;  
 String fecha;  
 boolean devuelto;  
 Prestamo siguiente;  
  
 public Prestamo(int codigoLibro, String usuario, String fecha) {  
 codigoLibro = codigoLibro;  
 usuario = usuario;  
 fecha = fecha;  
 devuelto = false;  
 siguiente = null;  
 }  
 }  
  
 // Nodo para la lista doble del historial  
 static class Historial {  
 String tipoOperacion; // ALTA, BAJA, PRESTAMO, DEVOLUCION  
 String fecha;  
 Historial siguiente;  
 Historial anterior;  
  
 public Historial(String tipoOperacion, String fecha) {  
 tipoOperacion = tipoOperacion;  
 fecha = fecha;  
 }  
 }  
  
 // =======================  
 // VARIABLES GLOBALES  
 // =======================  
 static final int *MAX\_LIBROS* = 50;  
 static Libro[] *catalogo* = new Libro[*MAX\_LIBROS*];  
 static int *cantidadLibros* = 0;  
  
 // Matriz de disponibilidad por sucursal (ejemplo: 3 sucursales)  
 static int[][] *disponibilidad* = new int[*MAX\_LIBROS*][3];  
  
 // Lista enlazada simple de préstamos  
 static Prestamo *prestamos* = null;  
  
 // Lista doble de historial  
 static Historial *historial* = null;  
 static Historial *ultimoHistorial* = null;  
  
 // Scanner único para entrada de datos  
 static Scanner *sc* = new Scanner(System.*in*);  
  
 // =======================  
 // MÉTODOS DEL SISTEMA  
 // =======================  
  
 // Agregar libro  
 static void cargarLibro() {  
 if (*cantidadLibros* >= *MAX\_LIBROS*) {  
 System.*out*.println("No hay espacio para más libros.");  
 return;  
 }  
 System.*out*.print("Código: ");  
 int codigo = *sc*.nextInt();  
 *sc*.nextLine(); // limpiar buffer  
  
 // Verificar duplicado  
 for (int i = 0; i < *cantidadLibros*; i++) {  
 if (*catalogo*[i].codigo == codigo && *catalogo*[i].activo) {  
 System.*out*.println("Error: ya existe un libro con ese código.");  
 return;  
 }  
 }  
 System.*out*.print("Título: ");  
 String titulo = *sc*.nextLine();  
 System.*out*.print("Autor: ");  
 String autor = *sc*.nextLine();  
 System.*out*.print("Stock inicial: ");  
 int stock = *sc*.nextInt();  
  
 *catalogo*[*cantidadLibros*] = new Libro(codigo, titulo, autor, stock);  
 *disponibilidad*[*cantidadLibros*][0] = stock; // en sucursal 1  
 *cantidadLibros*++;  
  
 *agregarHistorial*("ALTA");  
 System.*out*.println("Libro agregado con éxito.");  
 }  
  
 // Eliminar libro (baja lógica)  
 static void eliminarLibro() {  
 System.*out*.print("Código del libro a eliminar: ");  
 int codigo = *sc*.nextInt();  
 for (int i = 0; i < *cantidadLibros*; i++) {  
 if (*catalogo*[i].codigo == codigo && *catalogo*[i].activo) {  
 *catalogo*[i].activo = false;  
 *agregarHistorial*("BAJA");  
 System.*out*.println("Libro eliminado (baja lógica).");  
 return;  
 }  
 }  
 System.*out*.println("No se encontró el libro.");  
 }  
  
 // Buscar por título  
 static void buscarLibroPorTitulo() {  
 *sc*.nextLine();  
 System.*out*.print("Título a buscar: ");  
 String titulo = *sc*.nextLine();  
 for (int i = 0; i < *cantidadLibros*; i++) {  
 if (*catalogo*[i].activo && *catalogo*[i].titulo.equalsIgnoreCase(titulo)) {  
 System.*out*.println("Encontrado: " + *catalogo*[i].titulo + " (Stock: " + *catalogo*[i].stock + ")");  
 return;  
 }  
 }  
 System.*out*.println("Libro no encontrado.");  
 }  
  
 // Prestar libro  
 static void prestarLibro() {  
 System.*out*.print("Código del libro a prestar: ");  
 int codigo = *sc*.nextInt();  
 *sc*.nextLine();  
 System.*out*.print("Usuario: ");  
 String usuario = *sc*.nextLine();  
  
 for (int i = 0; i < *cantidadLibros*; i++) {  
 if (*catalogo*[i].codigo == codigo && *catalogo*[i].activo) {  
 if (*catalogo*[i].stock > 0) {  
 *catalogo*[i].stock--;  
  
 System.*out*.print("Fecha del préstamo (dd/mm/aaaa): ");  
 String fecha = *sc*.nextLine();  
  
 Prestamo nuevo = new Prestamo(codigo, usuario, fecha);  
 if (*prestamos* == null) {  
 *prestamos* = nuevo;  
 } else {  
 Prestamo temp = *prestamos*;  
 while (temp.siguiente != null) temp = temp.siguiente;  
 temp.siguiente = nuevo;  
 }  
 *agregarHistorial*("PRESTAMO");  
 System.*out*.println("Préstamo realizado.");  
 return;  
 } else {  
 System.*out*.println("No hay stock disponible.");  
 return;  
 }  
 }  
 }  
 System.*out*.println("Libro no encontrado.");  
 }  
  
 // Devolver libro  
 static void devolverLibro() {  
 System.*out*.print("Código del libro a devolver: ");  
 int codigo = *sc*.nextInt();  
 *sc*.nextLine();  
 System.*out*.print("Usuario: ");  
 String usuario = *sc*.nextLine();  
  
 Prestamo anterior = null;  
 Prestamo actual = *prestamos*;  
  
 while (actual != null) {  
 if (actual.codigoLibro == codigo && actual.usuario.equals(usuario) && !actual.devuelto) {  
 actual.devuelto = true;  
 // aumentar stock  
 for (int i = 0; i < *cantidadLibros*; i++) {  
 if (*catalogo*[i].codigo == codigo) {  
 *catalogo*[i].stock++;  
 break;  
 }  
 }  
 // eliminar de lista  
 if (anterior == null) {  
 *prestamos* = actual.siguiente;  
 } else {  
 anterior.siguiente = actual.siguiente;  
 }  
  
 System.*out*.print("Fecha de devolución (dd/mm/aaaa): ");  
 String fecha = *sc*.nextLine();  
  
 *agregarHistorial*("DEVOLUCION");  
 System.*out*.println("Libro devuelto.");  
 return;  
 }  
 anterior = actual;  
 actual = actual.siguiente;  
 }  
 System.*out*.println("Préstamo no encontrado.");  
 }  
  
 // Listar préstamos activos  
 static void listarPrestamos() {  
 Prestamo temp = *prestamos*;  
 if (temp == null) {  
 System.*out*.println("No hay préstamos activos.");  
 return;  
 }  
 while (temp != null) {  
 if (!temp.devuelto) {  
 System.*out*.println("Libro " + temp.codigoLibro + " prestado a " + temp.usuario + " el " + temp.fecha);  
 }  
 temp = temp.siguiente;  
 }  
 }  
  
 // Historial  
 static void agregarHistorial(String operacion) {  
 System.*out*.print("Fecha de la operación (dd/mm/aaaa): ");  
 String fecha = *sc*.nextLine();  
  
 Historial nuevo = new Historial(operacion, fecha);  
 if (*historial* == null) {  
 *historial* = nuevo;  
 *ultimoHistorial* = nuevo;  
 } else {  
 *ultimoHistorial*.siguiente = nuevo;  
 nuevo.anterior = *ultimoHistorial*;  
 *ultimoHistorial* = nuevo;  
 }  
 }  
  
 static void listarHistorialAdelante() {  
 Historial temp = *historial*;  
 while (temp != null) {  
 System.*out*.println(temp.tipoOperacion + " en " + temp.fecha);  
 temp = temp.siguiente;  
 }  
 }  
  
 static void listarHistorialAtras() {  
 Historial temp = *ultimoHistorial*;  
 while (temp != null) {  
 System.*out*.println(temp.tipoOperacion + " en " + temp.fecha);  
 temp = temp.anterior;  
 }  
 }  
  
 // =======================  
 // MENÚ PRINCIPAL  
 // =======================  
 public static void main(String[] args) {  
 int opcion;  
 do {  
 System.*out*.println("\n--- MENU BIBLIOTECA ---");  
 System.*out*.println("1. Cargar libro");  
 System.*out*.println("2. Eliminar libro");  
 System.*out*.println("3. Buscar libro por título");  
 System.*out*.println("4. Prestar libro");  
 System.*out*.println("5. Devolver libro");  
 System.*out*.println("6. Listar préstamos");  
 System.*out*.println("7. Historial adelante");  
 System.*out*.println("8. Historial atrás");  
 System.*out*.println("9. Salir");  
 System.*out*.print("Opción: ");  
 opcion = *sc*.nextInt();  
 *sc*.nextLine(); // limpiar buffer  
  
 switch (opcion) {  
 case 1 -> *cargarLibro*();  
 case 2 -> *eliminarLibro*();  
 case 3 -> *buscarLibroPorTitulo*();  
 case 4 -> *prestarLibro*();  
 case 5 -> *devolverLibro*();  
 case 6 -> *listarPrestamos*();  
 case 7 -> *listarHistorialAdelante*();  
 case 8 -> *listarHistorialAtras*();  
 case 9 -> System.*out*.println("Saliendo...");  
 default -> System.*out*.println("Opción no válida.");  
 }  
 } while (opcion != 9);  
 }  
}

README  
  
# 📚 BibliotecaU1 – Sistema de Gestión de Biblioteca  
  
## 📌 Descripción  
Este proyecto implementa un sistema de gestión de biblioteca en \*\*Java\*\*, aplicando estructuras de datos fundamentales de la Unidad 1.   
El sistema permite:  
- Registrar libros en un catálogo.  
- Administrar préstamos y devoluciones.  
- Mantener un historial de operaciones que puede recorrerse hacia adelante y hacia atrás.  
  
El programa está diseñado en modo consola e integra \*\*arrays, listas enlazadas simples, listas dobles y programación estructurada\*\*.  
  
---  
  
## 🛠️ Razones de diseño y complejidad  
  
### 1. Catálogo de libros  
- \*\*Estructura:\*\* `Libro[]` (array unidimensional de tamaño fijo).  
- \*\*Razón:\*\* acceso rápido por índice y control de cantidad máxima de libros.  
- \*\*Complejidad:\*\*  
 - Inserción: `O(1)`  
 - Búsqueda lineal por título: `O(n)`  
  
### 2. Disponibilidad por sucursal  
- \*\*Estructura:\*\* `int[][]` (matriz bidimensional).  
- \*\*Razón:\*\* permite manejar existencias de un mismo libro en varias sucursales.  
- \*\*Complejidad:\*\* acceso directo `O(1)`.  
  
### 3. Préstamos  
- \*\*Estructura:\*\* lista enlazada simple (`Prestamo`).  
- \*\*Razón:\*\* los préstamos son dinámicos, se insertan y eliminan con frecuencia.  
- \*\*Complejidad:\*\*  
 - Inserción al final: `O(n)`  
 - Eliminación/búsqueda de préstamo: `O(n)`  
  
### 4. Historial de operaciones  
- \*\*Estructura:\*\* lista doblemente enlazada (`Historial`).  
- \*\*Razón:\*\* permite recorrer las operaciones tanto hacia adelante como hacia atrás.  
- \*\*Complejidad:\*\*  
 - Inserción al final: `O(1)`  
 - Recorrido: `O(n)`  
  
  
---  
  
## ▶️ Cómo compilar y ejecutar  
  
### Requisitos  
- \*\*Java JDK 17\*\* o superior.  
- IntelliJ IDEA (o cualquier IDE que soporte Java).  
  
### Compilar desde terminal  
1. Posicionarse en la carpeta `src` del proyecto.  
2. Ejecutar:  
 ```*bash* javac BibliotecaU1.java