Ejercicio de Programación en Java: Gestión de una Biblioteca con Persistencia en Ficheros

Se desea desarrollar una aplicación en Java que permita gestionar una colección de libros mediante una interfaz de consola. El programa debe ofrecer al usuario un conjunto de opciones para consultar, añadir y visualizar libros, así como garantizar la persistencia automática de los datos mediante el uso de ficheros de texto.

Requisitos funcionales:

El programa debe permitir al usuario realizar las siguientes operaciones:

1. Añadir un nuevo libro introduciendo: título, autor, ISBN y año de publicación. No se deben permitir libros con ISBN duplicados.
2. Buscar libros por ISBN.
3. Buscar libros por autor (mostrando todos los resultados que contengan el texto introducido).
4. Buscar libros por título (mostrando todos los resultados que contengan el texto introducido).
5. Mostrar todos los libros registrados.
6. Salir del programa.

Requisitos de persistencia:

- Cada vez que se **añada un libro**, los datos deben **guardarse automáticamente** en un fichero de texto **sin necesidad de que el usuario lo solicite**.
- **Antes de ejecutar cualquier opción del menú**, incluidos los listados o búsquedas, el programa debe **recargar los datos desde el fichero** para garantizar que siempre trabaja con la última versión disponible.
- El fichero será de texto plano, y cada libro ocupará una línea, con los campos separados por punto y coma (;) en el siguiente orden:

  título;autor;ISBN;año

  Ejemplo de línea:

  Cien años de soledad;Gabriel García Márquez;9788497592208;1967

Requisitos técnicos:

- El programa debe ejecutarse en bucle hasta que el usuario elija salir.
- Todas las entradas deben ser validadas: el ISBN debe ser único, el año debe ser numérico, y los campos no pueden estar vacíos.
- Se debe manejar correctamente cualquier error que ocurra durante la lectura o escritura del fichero (por ejemplo, archivo no encontrado, problemas de permisos, formato inválido, etc.).
- La estructura del programa debe seguir los principios de la programación orientada a objetos, con separación clara entre la lógica de gestión y la lógica de entrada/salida.

1. Escribir una línea en un fichero de texto (añadir al final)

```
try (BufferedWriter writer = new BufferedWriter(new FileWriter("libros.txt", true))) {
   writer.write("Cien años de soledad;Gabriel García Márquez;9788497592208;1967");
   writer.newLine(); // Para saltar de línea
} catch (IOException e) {
   System.err.println("Error al escribir en el fichero: " + e.getMessage());
}
```

FileWriter(..., true) activa el modo *append*, es decir, no sobreescribe el fichero.

---

2. Leer un fichero línea a línea

```
try (BufferedReader reader = new BufferedReader(new FileReader("libros.txt"))) {
   String linea;
   while ((linea = reader.readLine()) != null) {
      System.out.println("Línea leída: " + linea);
      // Aquí podrías hacer un split(";") para separar los campos y generar objetos…
   }
} catch (IOException e) {
   System.err.println("Error al leer el fichero: " + e.getMessage());
}
```

---

3. Dividir una línea por punto y coma

```
String linea = "Cien años de soledad;Gabriel García Márquez;9788497592208;1967";
String[] campos = linea.split(";");
String titulo = campos[0];
String autor = campos[1];
String isbn = campos[2];
int year = Integer.parseInt(campos[3]);
```

---

4. Sobrescribir el fichero completo (por ejemplo, al guardar toda la lista de nuevo)

```
try (BufferedWriter writer = new BufferedWriter(new FileWriter("libros.txt"))) {
   for (Libro libro : listaLibros) {
      writer.write(libro.getTitulo() + ";" + libro.getAutor() + ";" + libro.getIsbn() + ";" + libro.getYear());
      writer.newLine();
   }
} catch (IOException e) {
   System.err.println("Error al guardar los libros: " + e.getMessage());
}
```