Título del Trabajo: "Gestión de Biblioteca con JDBC"

Contexto del Proyecto:

El objetivo del trabajo es desarrollar una pequeña aplicación para gestionar la información de una biblioteca usando JDBC, con operaciones básicas de CRUD (Crear, Leer, Actualizar y Eliminar) sobre varias entidades. Se deberán implementar relaciones entre entidades con la tecnología JDBC, teniendo en cuenta la separación de responsabilidades para un buen diseño orientado a objetos.

Entidades del Sistema:

1. **Libro** (Libro):
    - id: Identificador único.
    - titulo: Nombre del libro.
    - isbn: Código ISBN del libro.
    - **Relación:** Un libro puede tener uno o más autores (Relación *muchos a muchos* con Autor).
2. **Autor** (Autor):
    - id: Identificador único.
    - nombre: Nombre del autor.
    - **Relación:** Un autor puede haber escrito uno o más libros (Relación *muchos a muchos* con Libro).
3. **Usuario** (Usuario):
    - id: Identificador único.
    - nombre: Nombre del usuario.
    - **Relación:** Un usuario puede pedir prestados varios libros (Relación *uno a muchos* con Prestamo).
4. **Préstamo** (Prestamo):
    - id: Identificador único.
    - fechaInicio: Fecha de inicio del préstamo.
    - fechaFin: Fecha de devolución estimada.
    - **Relación:** Un préstamo está asociado a un usuario y a un libro (Relación *muchos a uno* con Usuario y Libro).

Objetivos Específicos:

1. Implementar operaciones **CRUD** para las entidades *Libro*, *Autor*, *Usuario* y *Préstamo* utilizando **JDBC**.
2. Establecer una relación **uno a muchos** entre *Usuario* y *Prestamo*.
3. Implementar una relación **muchos a muchos** entre *Libro* y *Autor*.
4. Separar correctamente las responsabilidades de las clases siguiendo el principio de **Responsabilidad Única (SRP)**.

Requisitos de la aplicación:

1. **CRUD Básico:**
    - Implementar las operaciones de **Crear**, **Leer**, **Actualizar** y **Eliminar** para las entidades *Libro*, *Autor*, *Usuario* y *Préstamo*.
    - Crear las correspondientes tablas en una base de datos relacional usando SQL (pueden usar *MySQL*, *PostgreSQL*, etc.).
    - Usar JDBC para conectarse a la base de datos y realizar estas operaciones.
2. **Relación 1 a muchos (Usuario - Prestamo):**
    - Cada *Usuario* puede tener varios *Préstamos*, pero cada *Préstamo* solo puede pertenecer a un *Usuario*.
    - Crear la tabla de *Prestamo* con una clave foránea que apunte a la tabla de *Usuario*.
    - Al eliminar un *Usuario*, todos sus *Préstamos* deberían eliminarse (ON DELETE CASCADE).
3. **Relación muchos a muchos (Libro - Autor):**
    - Un *Libro* puede tener varios autores, y un *Autor* puede haber escrito varios libros.
    - Implementar esta relación a través de una tabla intermedia Libro_Autor con las claves foráneas idLibro e idAutor.
    - Al actualizar o eliminar un *Libro* o *Autor*, manejar adecuadamente la sincronización en la tabla intermedia.
4. **Separación de Responsabilidades:**
    - **Clases de Entidad**: Crear clases de entidad como Libro, Autor, Usuario, y Prestamo que representen los objetos del modelo.
    - **Clases DAO (Data Access Object)**: Implementar una clase DAO para cada entidad (por ejemplo, LibroDAO, AutorDAO, etc.) que se encargue de interactuar con la base de datos usando JDBC.
    - **Clase de Servicio**: Crear una clase de servicio (BibliotecaService) que contenga la lógica de negocio (por ejemplo, agregar un libro y asignarle autores).
    - **Clase Principal**: Crear una clase principal Main que gestione la interacción del usuario con el sistema (ya sea a través de consola o interfaz gráfica).

Bonus (Opcional):

- **Interfaz Gráfica**: Si lo consideran conveniente, los estudiantes pueden implementar una interfaz gráfica con *Swing* para facilitar la interacción con el sistema. Si no no pasa nada, esto es si os sobra mucho tiempo al hacerlo y os sentís ociosas/os.