Creación de una base de datos orientada a objetos utilizando Java:

Suponemos que estamos diseñando un sistema para una biblioteca. Necesitas modelar libros y clientes. Cada libro tiene un título, un autor y un identificador único. Cada cliente tiene un nombre, un número de identificación único y una lista de libros que ha tomado prestados.

Primero, creamos las clases `Libro` y `Cliente`:

```
public class Libro {
	private String titulo;
	private String autor;
	private int id;

	// Constructor
	public Libro(String titulo, String autor, int id) {
	this.titulo = titulo;
	this.autor = autor;
	this.id = id;
	}

	// Getters
	public String getTitulo() {
	return titulo;
	}

	public String getAutor() {
	return autor;
	}

	public int getId() {
	return id;
	}
}

public class Cliente {
	private String nombre;
	private int id;
	private ArrayList<Libro> librosPrestados;

	// Constructor
	public Cliente(String nombre, int id) {
	this.nombre = nombre;
	this.id = id;
	this.librosPrestados = new ArrayList<>();
	}

	// Métodos para prestar y devolver libros
```

```
    public void prestarLibro(Libro libro) {
    librosPrestados.add(libro);
    }

    public void devolverLibro(Libro libro) {
    librosPrestados.remove(libro);
    }

    // Getters
    public String getNombre() {
    return nombre;
    }

    public int getId() {
    return id;
    }

    public ArrayList<Libro> getLibrosPrestados() {
    return librosPrestados;
    }
}
```

Ahora, probramos estas clases en nuestro método `main`:

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
    // Crear algunos libros
    Libro libro1 = new Libro("El señor de los anillos", "J.R.R. Tolkien", 1);
    Libro libro2 = new Libro("Cien años de soledad", "Gabriel García Márquez", 2);

    // Crear algunos clientes
    Cliente cliente1 = new Cliente("Juan", 101);
    Cliente cliente2 = new Cliente("María", 102);

    // Prestar libros
    cliente1.prestarLibro(libro1);
    cliente2.prestarLibro(libro2);

    // Mostrar libros prestados por cada cliente
    System.out.println(cliente1.getNombre() + " tiene los siguientes libros prestados:");
    for (Libro libro : cliente1.getLibrosPrestados()) {
    System.out.println(libro.getTitulo());
    }

    System.out.println(cliente2.getNombre() + " tiene los siguientes libros prestados:");
```

```
        for (Libro libro : cliente2.getLibrosPrestados()) {
        System.out.println(libro.getTitulo());
        }
        }
}
```

En este ejemplo básico de cómo podemos diseñar un sistema de base de datos orientada a objetos en Java para una biblioteca. Además de realizar y replicar el ejemplo que os dejo, hay que agregar las siguientes funcionalidades:

1. **Buscar libros por autor:** Agrega un método en la clase Biblioteca que tome el nombre de un autor como argumento y devuelva una lista de libros escritos por ese autor.
2. **Mostrar todos los libros disponibles:** Agrega un método en la clase Biblioteca que muestre todos los libros disponibles para préstamo.
3. **Verificar disponibilidad de un libro:** Agrega un método en la clase Biblioteca que tome el identificador de un libro como argumento y devuelva true si el libro está disponible para préstamo y false si no lo está.
4. **Mostrar libros prestados por un cliente específico:** Agrega un método en la clase Cliente que muestre todos los libros prestados por ese cliente.
5. **Mostrar todos los clientes con libros prestados:** Agrega un método en la clase Biblioteca que muestre todos los clientes que tienen libros prestados junto con los libros que tienen.
6. **Devolver un libro:** Agrega un método en la clase Cliente que tome un libro prestado como argumento y lo devuelva a la biblioteca.
7. **Gestión de multas:** Implementa un sistema que registre la fecha de devolución esperada de cada libro prestado. Si un cliente no devuelve un libro a tiempo, se le puede aplicar una multa.
8. **Clasificación de libros por género o categoría:** Agrega una propiedad "género" a la clase Libro y permite que los libros se clasifiquen y busquen por género.