```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
package model;

/**
* Classe Livro representa um livro no sistema de inventário.
*/
public class Livro {
   private int id;
   private String titulo;
   private String autor;
   private String isbn;
   private int anoPublicacao;
   private int quantidade;

   // Construtores, getters e setters...
}
```

ConexaoBD.java

java

```
package model;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

```
/**
* Classe ConexaoBD é responsável por gerenciar a conexão com o banco de dados.
*/
public class ConexaoBD {
    private static final String URL = "jdbc:mysql://localhost:3306/inventario_livros";
    private static final String USER = "root";
    private static final String PASSWORD = "password";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

LivroController.java

java

```
package controller;

import model.Livro;
import model.ConexaoBD;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
* Classe LivroController gerencia as operações relacionadas aos livros.
*/
public class LivroController {
```

```
    // Método para adicionar um livro
    public void adicionarLivro(Livro livro) throws SQLException {
        String sql = "INSERT INTO livros (titulo, autor, isbn, anoPublicacao, quantidade) VALUES (?, ?, ?, ?, ?)";
        try (Connection conn = ConexaoBD.getConnection(); PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, livro.getTitulo());
            stmt.setString(2, livro.getAutor());
            stmt.setString(3, livro.getIsbn());
            stmt.setInt(4, livro.getAnoPublicacao());
            stmt.setInt(5, livro.getQuantidade());
            stmt.executeUpdate();
        }
    }

    // Métodos para buscar, atualizar e remover livros...

    // Método para ordenar os livros por título
    public List<Livro> ordenarPorTitulo() throws SQLException {
        String sql = "SELECT * FROM livros ORDER BY titulo";
        List<Livro> livros = new ArrayList<>();
        try (Connection conn = ConexaoBD.getConnection(); Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                Livro livro = new Livro();
                livro.setId(rs.getInt("id"));
                livro.setTitulo(rs.getString("titulo"));
                livro.setAutor(rs.getString("autor"));
```

```
            livro.setIsbn(rs.getString("isbn"));
            livro.setAnoPublicacao(rs.getInt("anoPublicacao"));
            livro.setQuantidade(rs.getInt("quantidade"));
            livros.add(livro);
        }
      }
      return livros;
    }

    // Outros métodos de ordenação e busca...
}
```

Main.java

java

```
package view;

import controller.LivroController;
import model.Livro;

import java.sql.SQLException;
import java.util.Scanner;

/**
* Classe Main é o ponto de entrada da aplicação.
*/
public class Main {
    public static void main(String[] args) {
        LivroController controller = new LivroController();
        Scanner scanner = new Scanner(System.in);
```

```
        // Exemplo de interação com o usuário
        System.out.println("Bem-vindo ao Sistema de Gestão de Inventário de Livros!");
        System.out.println("1. Adicionar Livro");
        System.out.println("2. Ordenar por Título");
        System.out.println("Escolha uma opção:");

        int opcao = scanner.nextInt();

        switch (opcao) {
            case 1:
                // Código para adicionar um livro
                break;
            case 2:
                try {
                    // Código para ordenar livros por título e exibir os resultados
                    System.out.println("Livros ordenados por título:");
                    controller.ordenarPorTitulo().forEach(livro ->
System.out.println(livro.getTitulo()));
                } catch (SQLException e) {
                    System.err.println("Erro ao ordenar livros: " + e.getMessage());
                }
                break;
            default:
                System.out.println("Opção inválida.");
        }
    }
}
```

Arquivo SQL: schema.sql

sql

```
CREATE DATABASE IF NOT EXISTS inventario_livros;
USE inventario_livros;

CREATE TABLE IF NOT EXISTS livros (
  id INT AUTO_INCREMENT PRIMARY KEY,
  titulo VARCHAR(255) NOT NULL,
  autor VARCHAR(255) NOT NULL,
  isbn VARCHAR(20) NOT NULL,
  anoPublicacao INT NOT NULL,
  quantidade INT NOT NULL



public class LivroController {


  public void adicionarLivro(Livro livro) throws SQLException {
    String sql = "INSERT INTO livros (titulo, autor, isbn, anoPublicacao, quantidade)
VALUES (?, ?, ?, ?, ?)";
    try (Connection conn = ConexaoBD.getConnection(); PreparedStatement stmt =
conn.prepareStatement(sql)) {
      stmt.setString(1, livro.getTitulo());
      stmt.setString(2, livro.getAutor());
      stmt.setString(3, livro.getIsbn());
      stmt.setInt(4, livro.getAnoPublicacao());
      stmt.setInt(5, livro.getQuantidade());
      stmt.executeUpdate();
    }
```

```
    }

    public List<Livro> ordenarPorTitulo() throws SQLException {
        List<Livro> livros = new ArrayList<>();
        String sql = "SELECT * FROM livros ORDER BY titulo";
        try (Connection conn = ConexaoBD.getConnection(); PreparedStatement stmt =
conn.prepareStatement(sql); ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                Livro livro = new Livro();
                livro.setId(rs.getInt("id"));
                livro.setTitulo(rs.getString("titulo"));
                livro.setAutor(rs.getString("autor"));
                livro.setIsbn(rs.getString("isbn"));
                livro.setAnoPublicacao(rs.getInt("anoPublicacao"));
                livro.setQuantidade(rs.getInt("quantidade"));
                livros.add(livro);
            }
        }
        return livros;
    }

    public void atualizarLivro(Livro livro) throws SQLException {
        String sql = "UPDATE livros SET titulo = ?, autor = ?, isbn = ?, anoPublicacao = ?,
quantidade = ? WHERE id = ?";
        try (Connection conn = ConexaoBD.getConnection(); PreparedStatement stmt =
conn.prepareStatement(sql)) {
            stmt.setString(1, livro.getTitulo());
            stmt.setString(2, livro.getAutor());
            stmt.setString(3, livro.getIsbn());
```

```
            stmt.setInt(4, livro.getAnoPublicacao());
            stmt.setInt(5, livro.getQuantidade());
            stmt.setInt(6, livro.getId());
            stmt.executeUpdate();
        }
    }

    public void removerLivro(int id) throws SQLException {
        String sql = "DELETE FROM livros WHERE id = ?";
        try (Connection conn = ConexaoBD.getConnection(); PreparedStatement stmt =
conn.prepareStatement(sql)) {
            stmt.setInt(1, id);
            stmt.executeUpdate();
        }
    }

    // Método para buscar livro por ID
    public Livro buscarLivroPorId(int id) throws SQLException {
        String sql = "SELECT * FROM livros WHERE id = ?";
        Livro livro = null;
        try (Connection conn = ConexaoBD.getConnection(); PreparedStatement stmt =
conn.prepareStatement(sql)) {
            stmt.setInt(1, id);
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    livro = new Livro();
                    livro.setId(rs.getInt("id"));
                    livro.setTitulo(rs.getString("titulo"));
```

```
            livro.setAutor(rs.getString("autor"));
            livro.setIsbn(rs.getString("isbn"));
            livro.setAnoPublicacao(rs.getInt("anoPublicacao"));
            livro.setQuantidade(rs.getInt("quantidade"));
          }
        }
      }
      return livro;
    }
}
);
```