RETO 3: JDBC - Por Gerard Luque Oliver

OBJETIVO
Dada la base de datos de biblioteca (trabajada en clase). Completa el programa principal de insertar un libro, para ello puedes ayudarte del código realizado en clase.
Tienes que incluir Statement, PrepareStatement y una PrepareCall.
Para el PrepareCall dispones del procedimiento de alta de una editorial que deberás de introducirla en la base de datos Biblioteca

Estructura del código

1. **Configuración inicial y conexión a la base de datos**
   - Se define la URL de la base de datos, el usuario y la contraseña.
   - Se utilizan las siguientes clases principales de JDBC:
     - Connection: para establecer la conexión.
     - Statement y PreparedStatement: para ejecutar consultas SQL.
     - CallableStatement: para ejecutar procedimientos almacenados en la base de datos.
   - Las conexiones se gestionan con un bloque *try-with-resources*, asegurando que los recursos se cierran automáticamente al finalizar.

```
public static void main(String[] args) {
    String url = "jdbc:mysql://localhost:3307/biblioteca";
    String usuario = "root";
    String pass = "";
    String sqlInsertP = "INSERT INTO `libro` " +
            "(`idLibro`, `ISBN`, `Titulo`, `NumeroEjemplares`, `idAutor`, `idEditorial`, `idTema`) " +
            "VALUES (NULL, ?, ?, ?, ?, ?, ?)";
    String prepareCall = "{CALL AltaEditorial(?, ?, ?)}";

    Scanner sc = new Scanner(System.in);

    try (Connection con = DriverManager.getConnection(url, usuario, pass);
         Statement sentencia = con.createStatement();
         PreparedStatement sentenciaPreparada = con.prepareStatement(sqlInsertP);
         CallableStatement cs = con.prepareCall(prepareCall);
```

2. **Flujo principal del programa**
   - Se verifica si un libro existe en la base de datos usando su ISBN.
   - Si no existe, se recopila información adicional (título, ejemplares, autor, editorial y tema) del usuario.
   - Se insertan los datos del libro, autor, editorial y tema en sus respectivas tablas si son nuevos.

Explicación paso a paso

1. Conexión a la base de datos
El programa se conecta a la base de datos utilizando DriverManager:

```
Connection con = DriverManager.getConnection(url, usuario, pass);
```

Si la conexión es exitosa, se imprime un mensaje:

```
System.out.println("Conectado con éxito a la Base de Datos");
```

2. Verificar si el libro existe
El método verificarLibro comprueba si un libro con el ISBN ingresado por el usuario ya está registrado:

```
public static boolean verificarLibro(Connection con, Scanner sc) throws SQLException {
   System.out.println("Introduzca el isbn del libro a consultar");
   String isbn = sc.next();
   String verificarLibro = "SELECT * FROM libro WHERE isbn = ?";
   try (PreparedStatement ps = con.prepareStatement(verificarLibro)) {
      ps.setString(1, isbn);
      ResultSet resultado = ps.executeQuery();
      return resultado.next();
   }
```

- Se utiliza un **PreparedStatement** para evitar inyecciones SQL.
- Si el ISBN existe, se informa al usuario y el programa termina.
- Si no existe, continúa el proceso para registrar el libro.

3. Registrar un libro nuevo
Si el libro no está registrado:

- El usuario ingresa:
  - ISBN, título, número de ejemplares, autor, editorial y tema.
- Se utiliza el método setString y setInt para completar una sentencia preparada:

```
//Sentencia preparada
sentenciaPreparada.setString(1,isbn);
sentenciaPreparada.setString(2,titulo);
sentenciaPreparada.setInt(3,ejemplares);
sentenciaPreparada.setInt(4,autorId);
sentenciaPreparada.setInt(5,editorialId);
sentenciaPreparada.setInt(6,temaId);
sentenciaPreparada.executeUpdate();
```

- Se llama a otros métodos auxiliares para gestionar autores, editoriales y temas.

4. Manejar autores
El método existeAutor verifica si un autor ya está registrado en la tabla autor:

- Si el autor no existe, se solicita al usuario el nombre del autor y se registra usando un Statement:

```
private static int existeAutor(String autor, Connection con,Scanner sc, Statement sentencia) throws SQLException {
   String selectAutor = "SELECT idAutor FROM autor WHERE nombreAutor = ?";
   try (PreparedStatement ps = con.prepareStatement(selectAutor)) {
      ps.setString(1, autor);
      try (ResultSet resultado = ps.executeQuery()) {
         if (resultado.next()) {
            return resultado.getInt("idAutor");
         }else {
            System.out.println("Vaya! no tenemos registrado este autor, repita el nombre para su registro");
            String nombreNuevoAutor = sc.nextLine();
            String sentenciaAuto = "INSERT INTO autor VALUES (NULL, '" + nombreNuevoAutor + "')";
            sentencia.executeUpdate(sentenciaAuto);

            return existeAutor(nombreNuevoAutor,con,sc,sentencia);
         }
      }
   }
}
```

5. Manejar editoriales

El método existeEditorial verifica si una editorial está registrada:

- Si no está registrado, se utiliza un **CallableStatement** para ejecutar un procedimiento almacenado que inserta la nueva editorial:

```
private static int existeEditorial(String editorial, Connection connection,Scanner sc, CallableStatement call) throws
SQLException {
    String selectEditorial = "SELECT idEditorial FROM editorial WHERE NombreEditorial = '" + editorial + "'";
    ResultSet salida = connection.createStatement().executeQuery(selectEditorial);
    if (salida.next()) {
        return salida.getInt("idEditorial");
    }else{
        System.out.println("No tenemos registrada esta editorial, repitala para su registro");
        String nuevoEditorial = sc.nextLine();
        System.out.println("Escriba la dirección");
        String nuevaDireccion = sc.nextLine();
        System.out.println("Escriba el teléfono");
        String nuevaTelefono = sc.nextLine();
        call.setString(1,nuevoEditorial);
        call.setString(2,nuevaDireccion);
        call.setString(3,nuevaTelefono);
        call.executeUpdate();

        return existeEditorial(nuevoEditorial,connection,sc,call);
    }
}
```

6. Manejar temas

El método existeTema funciona de manera similar al manejo de autores:

- Comprueba si el tema ya está registrado.
- Si no lo está, lo inserta usando un Statement:

```
private static int existeTema(String tema, Connection connection, Scanner sc, Statement sentencia) throws
SQLException {
    String selectTema = "SELECT idTema FROM tema WHERE NombreTema = '" + tema + "'";
    ResultSet salida = connection.createStatement().executeQuery(selectTema);

    if(salida.next()){
        return salida.getInt("idTema");
    }else{
        System.out.println("No tenemos este tema registrado, repitame el nombre para su registro");
        String nombreTema= sc.nextLine();
        String sentenciaInsert = "INSERT INTO tema VALUES(null,'" + nombreTema + "')";
        sentencia.executeUpdate(sentenciaInsert);

        return existeTema(nombreTema,connection,sc,sentencia);
    }
}
```

7. Manejo de entradas del usuario

Se utiliza un bucle para asegurarse de que el usuario ingrese un valor válido para el número de ejemplares:

```
System.out.println("Dígame el número de ejemplares por favor");
int ejemplares;
while (true) {
    try {
        ejemplares = sc.nextInt();
        sc.nextLine();
```

```
        break;
    } catch (InputMismatchException e) {
        System.out.println("Entrada no válida. Por favor, ingrese un número entero:");
        sc.nextLine();
    }
}
```

Manejo de errores

El programa está preparado para manejar errores comunes:

1. **Errores de conexión**: Se capturan con un bloque catch (SQLException e), mostrando el mensaje de error.
2. **Errores de entrada del usuario**: Se manejan con try-catch alrededor de Scanner.