Drivers

En estos ejemplos se utiliza el SGBD **h2 database engine**, aunque se puede utilizar el mismo mecanismo para cualquier otro sgbd, siempre que se disponga del driver.

http://www.h2database.com

Interfaces y clases

Connection (Interface)	Representa a una conexión desde la app hasta una base de datos. La conexión se obtiene solicitándola a la clase DriverManager
PreparedStatement (Interface)	Un objeto que acepta una sentencia SQL escrita en una cadena y la precompila para ser ejecutada de manera eficiente múltiples veces, cambiando sus parámetros. Permite tener *placeholders* (con la marca **?**) para parametrizar la ejecución de la sentencia. También es muy útil para librarse de la mayor parte de intentos de ataque por inyección de código SQL. Se obtiene solicitándolo a una conexión.
~~*Statement*~~ (Interface)	**No recomendado por ser más susceptible a inyección de código que*PreparedStatement*** Es un objeto que permite la ejecución de una sentencia SQL.
ResultSet	Un objeto que representa a los datos devueltos por una sentencia SELECT o similar. En realidad, es un cursor a los resultados. Sólo puede haber un ResultSet activo por cada Statement.
DriverManager	Gestiona los drivers jdbc cargados en la app. Permite obtener conexiones a una BD.
SQLException	Las excepciones lanzadas por los métodos de los objetos de jdbc.

Todas las interfaces y clases mencionadas están en el paquete java.sql.*

Cadenas de conexión en h2

jdbc:h2:tcp://nombre_maquina-o-ip/~/nombre_bd	Conexión en modo servidor a una bd en la carpeta del usuario
jdbc:h2:tcp://nombre_maquina-o-ip/./nombre_bd	Conexión en modo servidor a una bd en la carpeta del servidor
jdbc:h2:tcp://nombre_maquina-o-ip/d:\programacion\nombre_bd	Conexión en modo servidor a una bd en una ruta absoluta en la máquina del servidor

Cadenas de conexión en h2 (cont)

jdbc:h2:/~/nombre_bd	Conexión en modo embebido a una bd en la carpeta del usuario
jdbc:h2:/./nombre_bd	Conexión en modo embebido a una bd en la carpeta del proyecto
jdbc:h2:d:\programacion\nombre_bd	Conexión en modo embebido a una bd en una ruta absoluta

Cada SGBD tiene sus drivers. La sintaxis de las cadenas de conexión es dependiente del driver. Es decir, las cadenas de conexión a una BD en Oracle no tienen la misma estructura que en h2

Secuencia simple para enviar una select

```
/ EJEMPLO de conexión y envío de una sentencia SELECT /
        // Paso 0: Cargar el driver. Sólo hay que hacerlo una vez
        // Ojo: puede lanzar ClassNotFoundException
        Class.forName("org.h2.Driver");

        // Declaramos referencias a las interfaces que usaremos.
        Connection conn = null; // Representa a la conexión
        PreparedStatement stmt = null; // Representa a la sentencia
        ResultSet rs = null; // Solo para Select
        try {
            // PASO 1: Obtener conexion
            conn = DriverManager.getConnection(
                    "jdbc:h2:tcp://localhost/~/test", // Cadena de conexión
                     "sa", "" // Usuario y contraseña
            );

            // Paso 2: Preparamos la sentencia. Podemos poner placeholders
            // para ejecutar la sentencia parametrizada.
            // Supongamos que queremos que la ciudad contenga una determinada
            // cadena y la nota sea superior a un cierto número

            String buscaCiudad = "la"; // Ej: Que la ciudad contenga 'la'
            double notaMinima = 6.0; // y la nota sea mayor o igual a 6

            String sql = "SELECT nombre, apellido, nota"
                    + " from alumno where ciudad like ? and nota >= ?";
            stmt = conn.prepareStatement(sql);

            // Paso 2a: Sustituir los placeholders por valores
            stmt.setString(1, "%" + buscaCiudad + "%");
            stmt.setDouble(2, notaMinima);
```

Secuencia simple para enviar una select (cont)

```
            // Paso 3: Ejecución de la query y obtención de los datos
            rs = stmt.executeQuery();

            // Paso 4 (Solo para select). Recorrer Los datos obtenidos.
            while (rs.next()) {
                String nombre = rs.getString("nombre"); // Se puede obtener el dato de
                String ape = rs.getString("apellido"); // una fila por nombre de columna
                Double nota = rs.getDouble(3); // o por posición
                System.out.format(Locale.ROOT, "%-15s %-15s %.2f",
                        nombre, ape, nota);
            }
        } catch (SQLException ex) {
            // Tomar las acciones necesarias para que el caso de uso continúe
            ex.printStackTrace();
        } finally {
            try {rs.close();} catch (Exception ex) {};
            try {stmt.close();} catch (Exception ex) {};
            try {conn.close();} catch (Exception ex) {};
        }
```

Otro ejemplo: Un insert

```
/ EJEMPLO de conexión y envío de una sentencia INSERT /
        // Paso 0: Cargar el driver. Sólo hay que hacerlo una vez
        // Ojo: puede lanzar ClassNotFoundException
        Class.forName("org.h2.Driver");


        // Declaramos referencias a las interfaces que usaremos.
        Connection conn = null; // Representa a la conexión
        PreparedStatement stmt = null; // Representa a la sentencia
        try {
            // PASO 1: Obtener conexion
            conn = DriverManager.getConnection(
                    "jdbc:h2:tcp://localhost/~/test", // Cadena de conexión
                     "sa", "" // Usuario y contraseña
            );

            // Paso 2: Preparamos la sentencia. Ponemos placeholders para
            // los valores.


            String nombre = "Pepe";
```

Otro ejemplo: Un insert (cont)

```
            String apellido = "Pérez";
            double nota = 8.0;

            String sql = "INSERT INTO alumno(nombre, apellido, nota)"
                    + " VALUES(?, ?, ?)";
            stmt = conn.prepareStatement(sql);

            // Paso 2a: Sustituir los placeholders por valores
            stmt.setString(1, nombre);
            stmt.setString(2, apellido);
            stmt.setDouble(3, nota);
            // Paso 3: Ejecución de la sentencia.
            int filas = stmt.executeUpdate();
            // devuelve el numero de filas afectadas
        } catch (SQLException ex) {
            // Tomar las acciones necesarias para que el caso de uso continúe
            ex.printStackTrace();
        } finally {
            try {stmt.close();} catch (Exception ex) {};
            try {conn.close();} catch (Exception ex) {};
        }
    }
```

Consideraciones

La conexión puede dejarse abierta y lanzar varias sentencias en la misma conexión.

La sentencia preparada puede reutilizarse, cambiando el valor de los placeholders con los setters y reejecutándola.

Una sentencia sólo puede tener un resultset abierto. Es decir, si una sentencia se reejecuta, cualquier resultset previamente abierto y asociado a ella queda en un estado no definido.

Al cerrar una sentencia se cierran los resultsets asociados, pero se recomienda cerrar los resultsets explícitamente con close() siempre que se pueda para liberar los recursos de la manera lo más eficiente posible.

Igualmente, al cerrar una conexión se cierran las sentencias asociadas, pero se recomienda cerrarlas explícitamente.

La apertura y cierre de la conexión suele ser una operación costosa. En algunos contextos no se recomienda abrir y cerrar las conexiones en cada operación.

Para optimizar el uso de conexiones permanentemente abiertas, por ejemplo en el contexto de las aplicaciones web, se suelen utilizar pools de conexiones

SGBD-R		
h2	Un sgbd pequeño, ligero y muy fiable y portable completamente construido en Java.	http://www.h2database.com
Apache Derby	Otro sgbd-r de la fundación apache. Escrito en Java.	https://db.apache.org/derby/
Oracle	Grande entre los grandes	https://www.oracle.com/database/index.html
MySQL	Popular durante mucho tiempo por ser bastante eficiente y económica. Muy utilizada en la web. Hoy es un producto de Oracle. Va cayendo en decadencia en favor de MariaDB	https://www.mysql.com/
MariaBD	SGBD de funcionamiento similar a MySQL	https://mariadb.org/
SQL Server	De Microsoft. Otro grande entre los grandes. Se integra facilmente con los productos de desarrollo de Microsoft (Visual Studio IDE, C#, Visual Basic.NET, etc)	https://www.microsoft.com/es-es/sql-server/
Access	De Microsoft. Prácticamente el único SGBD orientado a la ofimática con éxito comercial.	https://products.office.com/es-es/access
IBM DB2	La solución SGBD-R de IBM	https://www.ibm.com/analytics/us/en/technology/db2/
PostgreSQL	Solución de código abierto. Muy popular en la web. Tiene fama de avanzado y fiable.	https://www.postgresql.org/
SQLite	El más popular de los peques. Pensado para ser embebido. Está escrito en C, por lo que va compilado a código nativo.	https://www.postgresql.org/