UA04. GESTIÓN DE LA INFORMACIÓN

4.11. Excepciones y cierres

Debemos tener en cuenta siempre que las **conexiones** con una base de datos **consumen muchos recursos** en el sistema gestor, y por lo tanto en el sistema informático en general. Por ello, conviene cerrarlas con el método **close** siempre que vayan a dejar de ser utilizadas, en lugar de esperar a que el **garbage collector** de Java las elimine.

También conviene cerrar las consultas (**Statement** y **PreparedStatement**) y los resultados (**ResultSet**) para liberar los recursos.

Una **excepción** es una situación que no se puede resolver y que provoca la detención del programa de manera abrupta. Se produce por una condición de error en tiempo de ejecución.

En Java hay muchos **tipos de excepciones**, el paquete **java.lang.Exception** es el que contiene los tipos de excepciones.

Excepciones

Cuando se produce un **error** durante la ejecución de un programa, se genera un objeto asociado a esa **excepción**. Ese objeto es de la clase **Exception** o de alguna de sus subclases. Este objeto se pasa entonces al código que se ha definido para gestionar la excepción.

En una porción de programa donde se trabajara con ficheros, y con bases de datos podríamos tener esta estructura para capturar las posibles excepciones.

```
try {
      // Bloque de instrucciones del try

    }catch (FileNotFoundException fnfe){
      // Bloque para excepción por fichero no encontrado

    }catch(IOException ioe){
      // Bloque para excepción por error de entrada salida

    }catch(SQLException sqle){
      // Bloque para excepción por error con SQL
    }catch(Exception e){

    }finally{
     //Instrucciones finales para, por ejemplo, limpieza
    }
```

El bloque de instrucciones del try es el que se ejecuta, y si en él ocurre un error que dispara una excepción, entonces se mira si es de tipo fichero no encontrado; si es así, se ejecutarían las instrucciones del bloque del fichero no encontrado. Si no era de ese tipo la excepción, se mira si es del siguiente tipo, o sea, de entrada salida, y así sucesivamente.

Las instrucciones que hay en el bloque del **finally**, se ejecutarán siempre, se haya producido una excepción o no, ahí suelen ponerse instrucciones de limpieza, de cierre de conexiones, etc.

Las acciones que se realizan sobre una base de datos pueden lanzar la excepción **SQLException**. Este tipo de excepción proporciona entre otra información:

- Una cadena de caracteres describiendo el error. Se obtiene con el método **getMesage**.
- Un código entero de error que especifica al fabricante de la base de datos.

Cierre de conexiones

Como ya hemos dicho, al trabajar con bases de datos, se consumen muchos recursos por parte del sistema gestor, así como del resto de la aplicación.

Por esta razón, resulta totalmente conveniente **cerrarlas** con el método **close** cuando ya no se utilizan.

Podríamos por tanto tener un ejemplo de cómo hacer esto:

```
Connection con = null ;
try {

    con = DriverManager.getConnection("jdbc:odbc:admdb");
    System.out.println("Conexión realizada con éxito.");
    // Aquí hacemos lo que necesitemos
}
catch( SQLException e ) {

    // Aquí haríamos lo necesario para gestionar la excepción SQL
}

finally {
      // Si hay conexión la cerramos
    if( con != null ) {
        try { con.close(); }
    catch( SQLException e ) {
         System.out.println(e.getMessage());
     }
    }
}
```