Diseño y Desarrollo de Sistemas de Información

Acceso a Bases de Datos con Java (JDBC)

- ***JDBC →Java DataBase Connectivity***
- Es la librería (API) estándar de acceso a base de datos desde Java
- Está incluida en Java SE (*Standard Edition*)
- En Java SE 6 se incluye JDBC 4.0. En la actualidad, la mayoría de las bases de datos soportan JDBC 3.0

- Para conectarse a una base de datos concreta, es necesario su **driver** JDBC
- El driver es un fichero JAR que se añade a la aplicación como cualquier otra librería
- El driver de Oracle se llama ***ojdbc6.jar***

- El funcionamiento de un programa JDBC sigue estos pasos:

1. Importar las clases necesarias
2. Cargar el driver JDBC
3. Identificar el origen de los datos (la base de datos)
4. Establecer una conexión con la base de datos
5. Enviar consultas SQL y procesar el resultado
6. Liberar los recursos al terminar

- Los programas deben declarar el uso del paquete ***java.sql***

```
import java.sql.*
```

Carga del driver

- La primera función que debe hacer la aplicación es cargar el driver JDBC
- Sólo hay que hacerlo una única vez al comienzo de la aplicación

```
Class.forName("oracle.jdbc.OracleDriver");
```

- El método forName() puede lanzar la excepción **ClassNotFoundException** si no puede encontrar la clase del driver (el fichero .jar)

Establecer una conexión

- Nuestra base de datos se encuentra en un servidor y las aplicaciones se comunicarán como clientes a través de la red
- El objeto ***Connection*** representa una conexión física entre el cliente y el servidor
- Para crear una conexión se usa la clase ***DriverManager*** y el método ***getConnection()***
- Se especifica la dirección IP o URL del servidor, el nombre y la contraseña de un usuario de la BD

```
Connection conn = DriverManager.getConnection(
 "jdbc:oracle:thin:@172.17.20.75:1521:rabida",
 "DDSI_001","DDSI_001");
```

Cerrar una conexión

- Cuando se termina de usar una conexión (Connection) es necesario liberar los recursos que han necesitado

- Se utiliza el método ***close()***

```
conn.close()
```

Gestión de los errores

- Es necesario mostrar información detallada sobre los posibles errores al interactuar con el origen de datos

- Utilizaremos bloques try-catch
  try → sentencias JDBC
  catch → capturar las excepciones

```
try {
    instrucciones
}
catch (tipo-excepción variable) {
    instrucciones
}
```

Gestión de los errores

- **SQLException**
  - Todos los métodos de las clases e interfaces JDBC pueden **lanzar** una SQLException
  - El elemento básico de cualquier excepción es una cadena que describe dicha excepción y que es el valor que devuelve el método **getMessage**()
- El **SQLState** también muestra información sobre la excepción que ha tenido lugar
  - Se obtiene mediante una llamada al método **getSQLState()**
  - El standard X/OpenSQL define que el SQL State es una cadena formada por 5 caracteres; los dos primeros definen la clasificación del estado. Los 3 siguientes definen la subclasificación
- El código de error del fabricante se obtiene al llamar al método **getErrorCode()** y su significado depende por completo del fabricante

Gestión de los errores

- Cuando se lanza una excepción SQL es posible que haya más de un objeto ***Exception*** asociado con el error

- El objeto **SQLException** es, básicamente, un nodo de una lista enlazada. Mediante el método ***getNextException()*** podemos consultar la lista completa

```
catch (SQLException sqle) {
     do // recorre todas las excepciones
     {
        // muestra mensajes para cada excepcion
        System.err.println("Exception occurred:\nMessage: " +
          sqle.getMessage());
        System.err.println("SQL state: " + sqle.getSQLState());
        System.err.println("Vendor code: " + sqle.getErrorCode() + "\n");
     } while((sqle = sqle.getNextException()) != null);
}
```

Gestión de los errores

- Ejemplo de captura de excepción SQLException

```
catch(SQLException se) {
   String mensaje = ("codigo: " + se.getErrorCode()+
                     " SQL: " + se.getSQLState()+
                     " Texto :" + se.getMessage());
   System.out.println("Atención, se ha producido un
                       error con " + mensaje);
}
```

Gestión de los errores

- En lugar de capturar las excepciones en el propio método, también se pueden propagar al método de nivel superior
- Para ello, en el método donde se vaya a lanzar la excepción, se indica el/los tipos de excepciones que pueden ser lanzados, usando la instrucción **throws**
- Y también podría lanzarse de forma explícita una excepción mediante la instrucción **throw**

```
public conexionOracle() throws Exception {
        …
}
```