Práctica 1
Statement, PreparedStatement y Pool de conexiones

Objetivos
Los objetivos perseguidos por esta práctica son:
- Repasar conceptos ya conocidos de JDBC
- Comparativa Statement/PreparedStatement
- Conexiones a diversos SGBD: Oracle, HSQLdb. Comparativa de tiempos.
- Uso de un Pool de conexiones

Preparación
- Crear un directorio RI22-23/lab1/ en tu HOME
- Descargar el material y descomprimirlo aquí. Se proporciona
  - una base de datos HSQLDB
  - un script para crear la misma base de datos en Oracle
  - drivers JDBC para conectarse a ambos SGBD
  - un driver para manejar pools de conexión (c3p0)
- Crear un directorio workspace y arrancar el IDE Eclipse
- Arrancar el servidor HSQLdb (ejecutar data/startup).
- Conectarse al servidor de Oracle. Ejecutar el "Script para la creación de tablas y datos en Oracle".

Datos de conexión
- **Oracle**
  - URL = "jdbc:oracle:thin:@156.35.94.98:1521:desa19";
  - PUERTOS: 1521, 1526, 5504, 5850, 3389
  - USER = UO
  - PASS = password Oracle
- **HSQLDB**
  - URL = "jdbc:hsqldb:hsql://localhost";
  - USER = "sa";
  - PASS = "";

Ejercicios
Crear distintos proyectos para solucionar los siguientes ejercicios. No olvides configurar el build path para enlazar las librerías (drivers y pool de conexiones) que se necesiten.
Si es necesario borrar las tablas de curso pasado:
SELECT 'DROP TABLE '||table_name||' CASCADE CONSTRAINTS;' FROM user_tables

1. Statement vs. PreparedStatement: comparación de tiempos

Crear un programa java que nos permita comparar el rendimiento de **Statement** frente al de **PreparedStatement**.

Para ello, el programa debe conectarse a la base de datos y ejecutar un bucle de **1000** repeticiones con la **misma consulta** utilizando una y otra.
Se medirán tiempos para compararlos. Para que la diferencia sea significativa debe ejecutarse una consulta de cierta complejidad que vaya cambiando en cada iteración del bucle. Como ejemplo, **contar el número de vehículos cuya matrícula contiene el número de iteración**.

En el caso del Statement se construirá la sentencia SQL y se ejecutará (concatenando cadenas, por ejemplo), y en el caso del PreparedStatement se deben utilizar marcadores **"?".**

Para medir tiempos se puede utilizar `System.currentTimeMillis()` que nos devuelve la fecha/hora en número de milisegundos desde el 01/01/1970

Realizar la comparativa conectándose tanto a Oracle como a HSQLDB.

Se puede ver una discusión muy interesante al respecto de lo trabajado en esta práctica en: http://asktom.oracle.com/pls/asktom/f?p=100:11:115361867455944::::P11_QUESTION_ID:1993620575194

2. Comprobación de costes: apertura y cierre de conexiones

Crear un programa que nos permita ver el coste de crear conexiones con la base de datos. Para ello se procederá de dos formas distintas midiendo los tiempos y comparándolos:

1. Se realizará un bucle de 100 repeticiones y dentro del bucle se abrirá una conexión, se realizará una consulta y se cerrará la conexión.
   Repetir 100 veces
   Abrir conexión
   Hacer algo (ej: contar los vehículos de la tabla *tvehicles*)
   Cerrar conexión
2. Similar al anterior, pero la apertura y cierre de la conexión se realizará fuera del bucle.
   Abrir conexión
   Repetir 100 veces
   Hacer algo (lo mismo que en el caso anterior)
   Cerrar conexión

3. Pool de Conexiones

Para esta práctica utilizaremos el pool de conexiones c3p0 cuyos drivers están en la carpeta lib (descargados de http://www.mchange.com/projects/c3p0/).
La práctica consiste en crear un pool de conexiones con las siguientes características:

- Máximo de conexiones (*maxPoolSize*): 30
- Mínimo de conexiones (*minPoolSize*): 3
- Tamaño inicial del pool (*initialPoolSize*): 3

Para comprobar el funcionamiento correcto haremos como en el ejercicio anterior en el que medimos el coste en tiempo de abrir y cerrar conexiones dentro de un bucle. Se debe realizar el mismo bucle pero comparando entre utilizar una conexión del pool o una conexión obtenida sin pool. (Ej hazAlgo: contar el número de vehículos de la tabla tvehicles)

```
for (int i=0; i<1000; i++){
     crearConexion();
     hazAlgo();
     cerrarConexion();
}
```

```
crearPoolConexiones();
for (int i=0; i<1000; i++){
     getConexionFromPool();
     hazAlgo();
     releaseConexion();
}
```

A modo de ejemplo de la utilización de c3p0 se copia aquí un extracto de código disponible en la web de c3p0 (http://www.mchange.com/projects/c3p0/#using_combopooleddatasource). Con el propio driver viene código de ejemplo. El código siguiente es para iniciar un Pooled DataSource. Una vez creado se podrán obtener Connection.

```
DataSource ds_unpooled = DataSources.unpooledDataSource(URL, USER, PASS);

Map overrides = new HashMap();
overrides.put("minPoolSize", 3);
overrides.put("maxPoolSize", 50);
overrides.put("initialPoolSize", 3);
ds_pooled = DataSources.pooledDataSource(ds_unpooled, overrides );
//ya está inicializado
//Para obtener conexión
con = ds_pooled.getConnection();
```

Se debe probar tanto con Oracle como con HSQLDB.
A continuación, intenta implementar hazAlgo con hilos, de forma que se reciban 100 intentos de conexión simultáneos y no secuenciales.

Tomar los tiempos después de crear el pool y al finalizar la ejecución. Compararlos con los tiempos anteriores. Tratar de explicar la diferencia.