Práctica 2: Alta de Alumnos

Dada la tabla de asignaturas:

- *Asignaturas (nombre(PK), nCreditos)*
  Donde **nombre** es la clave y representa el nombre de la asignatura.

la tabla de grupos:

- *Grupos (idGrupo, asignatura (FK ---> Asignaturas), plazasLibres)*
  PK compuesta = idGrupo + asignatura
  Que representa los grupos de cada asignatura, y donde **plazasLibres** indica el número de plazas libres que quedan en ese grupo para esa asignatura.

y la tabla de matrículas:

- *Matriculas(idMatricula(PK), alumno, asignatura, grupo)*
  FK Compuesta grupo+asignatura--> Grupos
  Donde **IdMatricula** se obtiene a partir de la secuencia **seq_matricula**

En la carpeta sql se haya un script SQL de carga que borra las tablas y las crea de nuevo, y el cual es llamado de manera automática al ejecutar el método main de la clase Main.

A su vez, la practica necesita de la librería *user_library* para Eclipse, la cual se encuentra incluida en el código repartido junto al tema de JDBC.

También se recomienda experimentar con la configuración de registros de la aplicación (logs), la cual se encuentra en el archivo log4j.properties. En ese archivo se pueden configurar los límites para la impresión de registros tanto en fichero como en la consola.

Tarea 1 - Implementar la siguiente transacción:

```
public void matricular(String alumno, String asig, int grupo) throws SQLException
```

Dado el alumno, asignatura y grupos que toma como argumentos, intenta insertar la fila correspondiente en la tabla de matrículas, y además decrementa el número de plazas libres de ese grupo de asa asignatura en uno.

El método se encuentra en la clase ServicioImpl, la cual extiende de la interfaz Servicio.

También se ha de completar la codificación de la clase AltaAlumnoException.java.

Posible pseudocódigo para la implementación (se puede hacer de otras formas):

1. Insertar una fila en la tabla de matrículas para el alumno p_alumno, la asignatura p_asig y el grupo p_grupo.

2. Actualizar la tabla de grupos decrementando el campo **plazasLibres** en una unidad para el grupo p_grupo de la asigntura p_asig.

El método tomará una conexión del pool al principio, y cometerá la transacción. Al finalizar, deberán de cerrarse todos los recursos utilizados.

Tarea 2 – Implementar excepciones:

El método lanza sólo excepciones del tipo AltaAlumnoException, con dos posibles códigos de errores.

1. La primera registrará el problema de que no exista el grupo o la asignatura pasadas por parámetro. Su código será el "1" y el mensaje de error **"Probablemente la asignatura o el grupo no exista."**.
   Para detectar este problema la clase OracleSGBDErrorUtil.java tiene codificado el error 2291 de Oracle como FK_VIOLATED, que precisamente es el error que ocurre (entre otras ocasiones) cuando se viola una clave ajena por inserción de una fila hija sin la correspondiente fila padre. Se valora que tu solución no dependa de que este error sea codificado como 2291, usando para ello el método checkExceptionToCode y la clase OracleSGBDErrorUtil.java.
2. La segunda registrará el problema de intentar matricular a un alumno en un grupo sin plazas vacantes. Su código será el "2" y el mensaje de error **"No hay plazas en ese grupo."**.

El resto de las excepciones serán tratadas dentro del propio método.

Toda excepción sea del tipo que sea retrocederá la transacción y se propagará al método que hace la llamada a la transacción.

Las excepciones que no sean del tipo AltaAlumnoException dejará el mensaje del error con logger.error(...) en el logger.

Pruebas automáticas:

Las pruebas automáticas se invocan desde el *main*, a través de la clase *Tests*. Estas pruebas hacen 4 cosas:

1. Matricular a un alumno en un grupo lleno (el grupo 1 de OFIM). La prueba comprobará que salta la excepción AltaAlumnoException, indicando que no hay plazas en ese grupo (por eso, de momento, al ejecutarlo, te sale NO se da cuenta de que el grupo está lleno MAL).
2. Matricular a un alumno en un grupo que no existe (el grupo 3 de OFIM). La prueba comprobará que salta la excepción AltaAlumnoException, indicando que no existe la asignatura o no existe el grupo (por eso, de momento, al ejecutarlo, te sale NO se da cuenta de que el grupo no existe MAL).
3. Matricular a un alumno en una asignatura que no existe (ALGEBRA). La prueba comprobará que salta la excepción AltaAlumnoException, indicando que no o no existe la asignatura o no existe el grupo (por eso, de momento, al ejecutarlo, te sale NO se da cuenta de que la asignatura no existe MAL).

4. Finalmente hacemos una matrícula sin problemas, (grupo1 de FPROG). En ese caso no tiene que saltar ninguna excepción y ese grupo quedará con 3 plazas libres, y además la fila de esa matrícula se habrá insertado. La prueba genera un String con el contenido de la base de datos y comprueba si tiene todos estos cambios. De momento te saldrá Matricula MAL a la espera que programes correctamente el método.