Práctica 1: Reserva de Pistas de Tenis

***Nota**:* Se trata de un ejercicio inicial en el que aún no se aborda el tratamiento de excepciones.

Sean las tablas:

1. *Pistas (nro)* cuya PK es *nro* que representa las pistas de un club de tenis
2. y *Reservas (Fecha, Hora, Pista, Socio)* cuya PK es la compuesta por *Fecha*, *Hora* y *Pista*, y que mantiene una FK hacia *pistas* mediante el campo Pista que representa el *nro* de pista. *Fecha* es un campo *date* y *Hora* es un campo *integer* que toma valores de 0 a 23. *Socio* es un VARCHAR(30) con el nombre del socio.

En la carpeta sql tienes un script SQL de carga que borra las tablas, las crea de nuevo, el cual utiliza secuencias para implementar la clave primaria de *Pistas.*

1. Implementar la transacción:

   ```
   public static boolean anularReserva(String socio, Date fecha, int hora,
                                       int pista) throws SQLException {
   ```

   que busca una reserva del socio del primer parámetro de la pista, fecha y hora de los tres últimos parámetros.

   El método devuelve verdadero si la anulación tiene éxito, falso en caso contrario.

   La excepción que lanza sólo tiene que ver con que se produzca cualquier error imprevisto de SQL (e.g., se cae la conexión, el disco se queda sin espacio, la *select* tiene un error sintáctico, la tabla de la *select* no existe etc …)

   ### Posible pseudocódigo

   1. Intentar borrar la fila correspondiente a la reserva
   2. Si borra alguna fila devolver verdadero, sino falso
   3. Si durante la ejecución se produce algún error hacer *rollback,* registrar el mensaje de error en el *logger* con nivel *error* y asegurarse de que (a) también devuelve falso y (b) la excepción SQL se propaga hacia el método que ha hecho la llamada.

2. Implementar la transacción:º

   ```
   public static boolean reservarPista(String socio, Date fecha, int hora)
                   throws SQLException {
   ```

   que busca una pista libre para esa hora y día, si la encuentra inserta la reserva a nombre del socio **socio**.

   El método devuelve verdadero si la reserva tiene éxito, falso en caso contrario.

   La excepción que lanza sólo tiene que ver con que se produzca cualquier error imprevisto de SQL (e.g., se cae la conexión, el disco se queda sin espacio, la *select* tiene un error sintáctico, la tabla de la *select* no existe etc …)

Posible pseudocódigo

1. Hacer una consulta que obtenga las pistas que no estén ocupadas en esa fecha y hora
2. Si la consulta devuelve alguna fila
    1. Leer el número de pista de la primera fila que me encuentre en la consulta
    2. Insertar una reserva para el socio, fecha y hora pasados por parámetro en la pista obtenida con la consulta
3. sino
    1. devolver falso
4. Si durante la ejecución se produce algún error hacer *rollback,* registrar el mensaje de error en el *logger* con nivel *error* y asegurarse de que (a) también devuelve falso y (b) la excepción SQL se propaga hacia el método que ha hecho la llamada.

Importante para ambas transacciones:

Utiliza la clase *PoolDeConexiones.java* que se te provee.

Utiliza sentencias preparadas, aprovechando que esa clase te implementa una *caché* de sentencias.

Parte de la clase vacía *PlantillaPistasDeTenis* que se te da como esqueleto. Nota que en el *main* ya se da un juego de pruebas partiendo de las tablas vacías. Es posible que casualmente el test pase alguna de las pruebas, pero es sólo debido a que aún no se ha hecho ninguna reserva correcta. Para que el ejercicio esté correcto, tendrían que aparecerte los cuatro mensajes “OK”.