Práctica JDBC: Gestion Médicos

Sean las tablas:

1. *cliente (NIF (PK), nombre, ape1, ape2, direccion)*
2. *medico (id_medico (PK), NIF, nombre, ape1, ape2, especialidad, consultas*)
3. *consulta (id_consulta (PK), fecha_consulta, NIF (FK → cliente), id_medico (FK → medico)).*
4. *anulacion (id_anulacion (PK), motivo_anulacion, fecha_anulacion, id_consulta (FK → consulta)).*

Se facilita el script SQL *`sql/gestion_medicos.sql`* que borra las tablas, las crea de nuevo y las rellena con varias filas de ejemplo. Se utilizan secuencias para implementar la clave primaria de *medico, consulta y anulacion.*

Se pide: Implementar las siguientes transacciones:

```
public static void reservar_consulta(String m_NIF_cliente, String
m_NIF_medico,  Date m_Fecha_Consulta)
      throws SQLException {
```

Escribirá un registro dentro de la tabla *consulta* y sumará un 1 al campo *consultas* de la tabla *medico* del médico en cuestión. Tendrá en cuenta que el médico en cuestión **solamente puede tener una consulta al día.**

```
public static void anular_consulta(String m_NIF_cliente, String
m_NIF_medico,  Date m_Fecha_Consulta, Date m_Fecha_Anulacion)
      throws SQLException {
```

Escribirá un registro en la tabla *anulacion*, restará un 1 el número de billetes del campo *consultas* de la tabla *consulta.* Para que se pueda anular debe existir una consulta para la Fecha de Consulta, Medico y Cliente en la tabla *consulta* y la Fecha Anulacion debe ser como mínimo 2 días antes de la Fecha Consulta, si no es así no se podrá anular.

```
public static void consulta_medico(String m_NIF_medico)
      throws SQLException {
```

Mostrará por salida estándar un listado ordenado por Fecha Consulta de las entradas en la tabla *consulta* del Medico introducido por parámetro.

Las transacciones deben implementarse en la clase **`lsi.ubu.solucion.GestionMedicos.java`** (que puedes comenzar a elaborar como una copia de `lsi.ubu.enunciado.EsqueletoGestionMedicos.java`):

Tratamiento de Excepciones:

En cualquiera de las excepciones se hará *rollback.*

Las situaciones particulares provocarán una excepción *`GestionMedicosException`* que será una subclase de *`SQLException`*. Las *`GestionMedicosException`* se te dan ya implementadas en el paquete `lsi.ubu.enunciado`, y son las siguientes:

- Si el NIF del cliente no existe en la tabla de clientes lanzará una excepción *`GestionMedicosException`*, con el código 1, y el mensaje “Cliente inexistente”.

- Si el NIF del medico no existe en la tabla *medico* lanzará una excepción `GestionMedicosException`, con el código 2, y el mensaje “Medico inexistente”.
- Si el médico ya tiene una consulta el mismo día se lanzará una excepción `GestionMedicosException`, con el código 3, y el mensaje “Médico ocupado”.
- Si la Fecha Consulta para un médico no tiene su correspondencia en la tabla *consulta* se lanzará una excepción `GestionMedicosException`, con el código 4, y el mensaje “Consulta inexistente”.
- Si no se puede anular una consulta se lanzará una excepción `GestionMedicosException`, con el código 5, y el mensaje “La consulta no se puede anular para la fecha introducida”.

Las `GestionTrenesException` simplemente se lanzarán al método que haya invocado la ejecución de la transacción (por ejemplo, a *pruebaAlquilar()*).

Las S`QLException` propiamente dichas (i.e., que no son `GestionMedicosException`), se registrarán con nivel error en el *logger* y también se lanzarán al método que haya invocado la ejecución de la transacción.

Importante:

- Utiliza la clase *PoolDeConexiones.java* que se te provee.
- Utiliza sentencias preparadas, aprovechando que esa clase te implementa una *caché* de sentencias.
- Libera todos los recursos utilizados con un bloque *finally*.

Además de estas 3 transacciones, tienes que implementar una batería de pruebas para cada una de ellas que recoja tanto casos de ejecución normal como casos extremos, incluyendo dentro de estos últimos cada uno de los que levantan las excepciones anteriormente descritas.

Importante: Las baterías de pruebas deberán de poder ejecutarse desde el main de la clase que implemente las 3 transacciones, pero puedes estructurarlas en otros métodos o incluso en una clase aparte (o usar frameworks de pruebas, si es que casualmente conoces alguno).

Condiciones de entrega

Se entregará el proyecto de Eclipse con nombre GestionMedicos_1C, comprimido en un fichero .zip con nombre "GestionMedicos_*NombreApellidos*.zip".

El código de las transacciones debe de implementarse en `GestionMedicos.java`, que ha de crearse dentro del paquete `lsi.ubu.solucion`.

El proyecto debe contener TODOS los ficheros necesarios (fuentes, recursos, etc.) y utilizar rutas relativas, para poder ejecutarse sin problemas en otro ordenador.
Se asumirá que en el equipo donde se corrige se dispone de una *User Library* de Eclipse (o equivalente) con nombre **user_library** con las bibliotecas indicadas en Material de Práctica. Es obligatorio respetar el nombre de la biblioteca de usuario. (Se recomienda revisar el enlace al vídeo de configuración de la librería de usuario con todos los JAR).

Recuerda que **se requiere** el uso de un **repositorio GIT** en la carpeta entregada, y deberá tener varias confirmaciones (commits) a medida que el proyecto se va desarrollando.

En caso de no compilar o no ejecutar correctamente, la calificación, será de cero en este proyecto.