Práctica JDBC: Gestión Donaciones Sangre

Sean las tablas:

1. *hospital (id_hospital(PK), nombre, localidad)*
2. *tipo_sangre (id_tipo_sangre (PK), descripcion)*
3. *reserva_hospital (id_tipo_sangre (PK, FK tipo_sangre), id_hospital (PK, FK hospital), cantidad)*
4. *donante (NIF (PK), nombre, ape1, ape2, fecha_nacimiento, id_tipo_sangre (FK tipo_sangre))*
5. *donación (id_donacion (PK), nif_donante (FK donante), cantidad, fecha_donacion)*
6. *traspaso (id_traspaso (PK), id_hostpial (FK hospital), id_tipo_sangre (FK tipo_sangre), cantidad, fecha_traspaso)*

Se facilita el script SQL *`sql/gestion_donaciones_sangre.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 *tipo_sangre, hospital, donación y traspaso.*

Se pide: Implementar las siguientes transacciones:

```
public static void realizar_traspaso(int m_ID_Hospital_Origen, int
m_ID_Hospital_Destino, int m_ID_Tipo_Sangre, float m_Cantidad, Date
m_Fecha_Traspaso)
      throws SQLException {
```

Escribirá un registro dentro de la tabla *traspaso* y sumará el contenido de m_m_Cantidad (en litros) al campo *cantidad* de la reserva del tipo *m_ID_Tipo_Sangre* del *m_ID_Hospital_Destino* y decrementará la misma cantidad en la reserva del tipo *m_ID_Tipo_Sangre* del hospital *m_ID_Hospital_Origen*. Si la reserva de un *hospital* no existe deberá crearse para el *tipo_sangre* correspondiente.

```
public static void realizar_doancion(String m_NIF, float m_Cantidad,int
m_ID_Hospital, Date m_Fecha_Donacion)
      throws SQLException {
```

Añadirá una entrada en la tabla *donaciones* con la información relativa a la la donación realizada e incrementará la *reserva* del *hospital* correspondiente. Hay que tener en cuenta que el máximo de donación es de 0,45 litros y que una persona no podrá donar más de una vez cada 15 días.

```
public static void consulta_traspasos(String m_Tipo_Sangre)
      throws SQLException {
```

Mostrará por salida estándar un listado ordenado por *id_hospital _destino* y *fecha_traspaso* de los traspasos realizados de un determinado *tipo de sangre* (el parámetro recibe la descripción del tipo de sangre que se encuentra en la tabla *tipo_sangre*). Se deben mostrar el contenido de la tabla *traspasos, hospital, reserva_hospital y tipo_sangre*.

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

Tratamiento de Excepciones:

En cualquiera de las excepciones se hará *rollback.*

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

- Si el NIF del donante no existe en la tabla de donantes lanzará una excepción *GestionDonacionesSangreException*, con el código 1, y el mensaje "Donante inexistente".
- Si el Tipo de sangre no existe en la tabla *tipo_sangre* lanzará una excepción *GestionDonacionesSangreException*, con el código 2, y el mensaje "Tipo Sangre inexistente".
- Si el hospital no existe en la tabla *hospital* se lanzará una excepción *GestionDonacionesSangreException*, con el código 3, y el mensaje "Hospital Inexistene".
- Si el donante ha donado en menos de 15 días se lanzará una excepción *GestionDonacionesSangreException*, con el código 4, y el mensaje "Donante excede el cupo de donación".
- Si el campo *cantidad* de la tabla *donacion* tiene un valor menor que 0 o superior a 0,45 se lanzará una excepción *GestionDonacionesSangreException*, con el código 5, y el mensaje "Valor de cantidad de donación incorrecto".
- Si el campo *cantidad* de la tabla *traspaso* tiene un valor menor o igual que 0 se lanzará una excepción *GestionDonacionesSangreException*, con el código 6, y el mensaje "Valor de cantidad de traspaso por debajo de lo requerido".

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

Las S*QLException* propiamente dichas (i.e., que no son *GestionDonacionesSangreException*), 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, **se debe** 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 GestionDonacionesSangre_2C, comprimido en un fichero .zip con nombre "GestionDonacionesSangre_2C_*NombreApellidos*.zip".

El código de las transacciones debe de implementarse en `GestionDonacionesSangre.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. Se puede seguir la guía en el siguiente enlace.

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