PROGRAMACION ORIENTADA A OBJETOS II

PRACTICAS DE LABORATORIO

ESTAS PRACTICAS DE LABORATORIO DEBEN SER REALIZADA DE MANERA INDIVIDUAL. LEA DETENIDAMENTE LAS INSTRUCCIONES, PUES EL NO SEGUIRLAS TAL COMO SE INDICA PUEDE RESULTAR EN UNA PRACTICA INCORRECTA. CUANDO LA PRÁCTICA SEA TERMINADA, ASEGURESE DE QUE CUMPLE CON TODOS LOS REQUERIMIENTOS SOLICITADOS Y DE QUE FUNCIONE CORRECTAMENTE ANTES DE ENTREGARLA.

PRACTICA 6. USO DE LA CLASE DriverManager EN JDBC

PRACTICA 7. USO DE LA INTERFACE DataSource EN JDBC

Estas prácticas tienen que ver con la API de JDBC en cuanto a la creación de tablas a través de la clase DriverManager y de la conexión a través de la interface DataSource. Se asume que la práctica 4 fue realizada exitosamente, por lo cual ya tiene instalado MySQL y configurada la variable de entorno CLASSPATH (necesaria solo en caso de que vaya a desarrollar la práctica sin usar IntelliJ IDEA). El desarrollo de estas prácticas puede hacerse usando IntelliJ IDEA, o desde la terminal de comandos si es que no tiene suficientes recursos su computadora para cargar IntelliJ.

Paso 1: Deberá crear la base de datos `controlescolar_ej2021` y darle permisos a un usuario de nombre `IngSW`, para ello, primero entre a la consola de MySQL tecleando el siguiente comando desde la ventana de comandos del sistema operativo: **`mysql -u root -p`**

y una vez en la consola de MySQL teclee los siguientes comandos:

```
CREATE DATABASE IF NOT EXISTS controlescolar_ej2021;
DROP USER IF EXISTS 'IngSW'@'localhost';
CREATE USER 'IngSW'@'localhost' IDENTIFIED BY 'UAZsw$021';
GRANT ALL ON controlescolar_ej2021.* TO 'IngSW'@'localhost';
```

Paso 2. Clone el repositorio usando la técnica que le convenga (ya sea usando Intellij IDEA o desde la terminal de comandos usando **git clone**). Los detalles están en el archivo README.md

NOTA: Para los siguientes pasos, edite los archivos mencionados usando ya sea Intellij IDEA o el editor de su preferencia. Los lugares donde debe hacer cambios están marcados en el código base con un comentario ***TODO***, no modifique ninguna otra parte del código.

Paso 3. Edite el archivo ***MainPractica06_07.java***, edite la línea siguiente:

```
public static final String matricula="xxx";
```

para que tenga como valor su matrícula. En este archivo es lo único que debe cambiar. La clase incluida en este archivo la puede ejecutar, **DE MANERA OPCIONAL**, para que el código que implemente en las otras cosas vaya mostrando mensajes de lo que va realizando.

Paso 4. Edite el archivo `CreaTablasConDriverManager.java`, para hacer lo siguiente (**vaya haciendo commit conforme vaya haciendo cada cambio**):

a) **Si no usará Intellij IDEA** debe indicar que paquetes importará, los cuales son los siguientes:

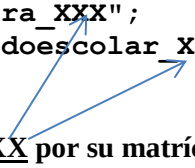
```
import java.io.*;
import java.sql.*;
import java.util.Scanner;
```

b) Coloca en los comentarios siguientes tu nombre en vez de las NNNNNNNN y tu matricula en vez de las XXXXXXXX:

```
/**
    @author:  NNNNNNNN
    @author:  XXXXXXXX
 */
```

c) Coloque los siguientes atributos dentro de la clase `CreaTablasConDriverManager`:

```
private Connection conn;
private final String nomCarrera = "carrera_XXX";
private final String nomPeriodo = "periodoescolar_XXX";
private final String[] nomtablas =
      {nomCarrera, nomPeriodo};
```



NOTA: En los valores de estas últimas constantes deben reemplazar XXX por su matrícula.

d) Coloque dentro el primer constructor de ***CreaTablasConDriverManager*** (el que recibe 4 strings), el siguiente código, el cual crea un URL a partir del primer y cuarto argumento y usa tal URL en conjunto con el segundo y tercer argumento para crear una conexión a través del método getConnection de la clase DriverManager:

```
String url = String.format("jdbc:mysql://%s/%s",
     direccionServidor,nomBD);
conn = DriverManager.getConnection(url,usuario,clave);
```

e) Coloque dentro el segundo constructor de ***CreaTablasConDriverManager*** (el que recibe 1 solo argumento), el siguiente código, el cual asocia el objeto de tipo Connection recibido como argumento al atributo con (que representa la conexión a usar en los métodos de esta clase), asegurándose que no sea null:

```
if (conn==null) {
   throw new Exception("Conector recibido es nulo");
}
this.conn = conn;
```

f) Coloque el siguiente código dentro del método `creaTablas()`. Este método creará las tres tablas necesarias (**note que en algunas sentencias SQL involucradas en este paso y pasos posteriores hay espacios entre las comillas, estos espacios son necesarios, de otra manera su programa no funcionará**)

```
Statement sentencia;
String prefijo="DROP TABLE IF EXISTS ";
String stringEliminacionCarrera = prefijo + nomCarrera;
String stringEliminacionPeriodo = prefijo + nomPeriodo;

if (debug) {
    System.out.println("Iniciando creacion de tablas ...");
}
sentencia = con.createStatement();
sentencia.executeUpdate(stringEliminacionCarrera);
sentencia.executeUpdate(stringEliminacionPeriodo);

Scanner lector =
      new Scanner(new File("tablas_ej2021.txt"));
int i=0;
while (lector.hasNext()) {
```

```
        String sql = lector.nextLine();
        if (debug) {
              System.out.println("Creando tabla "+
                    nomtablas[i]+"..");
        }
        sentencia.executeUpdate(String.format(sql,
          nomtablas[i]));
        i++;
  }
  sentencia.close();
```

g) Coloque el siguiente código dentro del método `llenaTablaCarrera()`. Este método insertará datos a la tabla cuyo nombre está en la variable **`nomCarrera`** a través del uso de un objeto `ResultSet` actualizable:

```
  if (debug) {
        System.out.println("\nColocando datos en tabla " +
        nomCarrera+":");
  }

  // Se crea un BufferedReader para leer archivo
  BufferedReader lector = new BufferedReader(
     new InputStreamReader(
        new FileInputStream("datosCarreras.txt")) );
  Statement sentElimina = conn.createStatement();
  sentElimina.executeUpdate("DELETE FROM "+nomCarrera);

  Statement sentencia = conn.createStatement(
              ResultSet.TYPE_SCROLL_SENSITIVE,
              ResultSet.CONCUR_UPDATABLE);
  ResultSet resultado = sentencia.executeQuery(
             "SELECT * FROM "+nomCarrera);
  String linea=lector.readLine();
  while (linea!=null) {
         // Indicamos que queremos usar un registro nuevo
         resultado.moveToInsertRow();
         String[] elems=linea.split(",");
         if (debug) {
           System.out.println("Agregando carrera "+elems[1]+" ...");
         }
         resultado.updateString(1, elems[0]);
         resultado.updateString(2,elems[1]);
         linea=lector.readLine();
         // Agregamos el registro a la tabla
         resultado.insertRow();
  }
  lector.close();
  sentencia.close();
```

h) Coloque el siguiente código dentro del método `close()`. Este método cierra la conexión, si es que existe:

```
if (conn!=null) {
    conn.close();
}
```

Paso 6. Ejecute las prueba de esta práctica dando click en ***CreaTablasConDriverManagerTest*** con el botón derecho y seleccionado Run si está en Intellij o emitiendo el comando **`gradle test --tests poo2.prac06.CreaTablasConDriverManagerTest`** desde la terminal de comando

Paso 7. Edite el archivo `ColocaDatosUsandoDataSource.java`, para hacer lo siguiente (**vaya haciendo commit conforme vaya haciendo cada cambio**):

a) **Si no usará Intellij IDEA** debe indicar que paquetes importará, los cuales son los siguientes:

```
import com.mysql.cj.jdbc.MysqlDataSource;
import java.sql.*;
```

b) Coloca en los comentarios siguientes tu nombre en vez de las NNNNNNNN y tu matricula en vez de las XXXXXXXX:

```
/**
    @author:  NNNNNNNN
    @author:  XXXXXXXX
 */
```

c) Coloque los siguientes atributos dentro de la clase `ColocaDatosUsandoDataSource`:

```
  private final Connection conn;
  private final String nomPeriodo = "periodoescolar_XXX";
```

NOTA: En el valor de la última constante debe reemplazar XXX por su matrícula.

d) Coloque dentro del constructor de `ColocaDatosUsandoDataSource` (el que recibe 4 strings), el siguiente código, el cual crea un DataSource específico a MySQL, lo configura con los argumentos recidos e inicializa el objeto Connection que representa a la conexión a usarse en los otros métodos de esta clase:

```
MysqlDataSource fuente = new MysqlDataSource();
fuente.setServerName(direccionServidor);
fuente.setUser(usuario);
fuente.setPassword(clave);
fuente.setDatabaseName(nomBD);
conn = fuente.getConnection();
```

e) Coloque el siguiente código dentro del método `llenaTablaPeriodoEscolar()`. Este método meterá datos a la tabla cuyo nombre está en la variable **`nomPeriodo`** de 3 maneras diferentes y demuestra cómo acceder al valor de un campo que se incrementa de manera automática (**`id_periodo`**). **NO SE LIMITE A SIMPLEMENTE COPIAR EL CÓDIGO, YA QUE AQUÍ SE ENCUENTRAN TÉCNICAS QUE NECESITARÁ USAR PARA POSTERIORES PROGRAMAS Y/O PRACTICAS, COPIAR LOS COMENTARIOS ES OPCIONAL, ESTAN PARA EXPLICAR LO QUE SE ESTA HACIENDO**:

```
if (debug) {
      System.out.println("\nColocando datos en tabla "
            + nomPeriodo +":");
}
```

```
Statement sentencia=conn.createStatement();
sentencia.executeUpdate("DELETE FROM "+nomPeriodo);
sentencia.executeUpdate("ALTER TABLE "+nomPeriodo+
         " AUTO_INCREMENT=1");
// 1a forma de obtener el valor de un campo autoincrementable
// A traves del metodo getGeneratedKeys() de la clase Statement
if (debug) {
   System.out.println("1a forma de obtener valor autoincrement:");
   System.out.println("Agregando el Periodo Ago-Dic del 2020...");
}
String sql = "INSERT INTO "+nomPeriodo +
     "(year, periodo) VALUES (2020,1)";
sentencia.executeUpdate(sql,
     Statement.RETURN_GENERATED_KEYS);
int valorLlaveAutoIncremento = -1;

// getGeneratedKeys devuelve un ResultSet con el valor
//    de los campos que se incrementaron de manera automatica
ResultSet rs = sentencia.getGeneratedKeys();
if (rs.next()) {
      valorLlaveAutoIncremento = rs.getInt(1);
}

if (debug) {
  System.out.println("El id_periodo que le toco fue : "
       + valorLlaveAutoIncremento+"\n");
}

// 2da forma de obtener el valor de un campo autoincrementable
// A traves de la funcion LAST_INSERT_ID() de MySQL
if (debug) {
   System.out.println("2a forma de obtener valor autoincrement:");
   System.out.println("Agregando el Periodo Ene-Jun del 2021...");
}

sql = "INSERT INTO "+nomPeriodo+"(year, periodo) VALUES (2021,2)";
sentencia.executeUpdate(sql);
valorLlaveAutoIncremento = -1;

// La funcion LAST_INSERT_ID() de MySQL devuelve el valor del
//    campo autoincrementable en el ultimo registro insertado
rs = sentencia.executeQuery("SELECT LAST_INSERT_ID()");
if (rs.next()) {
   valorLlaveAutoIncremento = rs.getInt(1);
}

if (debug) {
   System.out.println("El id_periodo que le toco fue : "
       + valorLlaveAutoIncremento+"\n");
 }
```

```
// 3a forma de obtener el valor de un campo autoincrementable
// A traves de un objeto ResultSet actualizable
if (debug) {
  System.out.println("3a forma de obtener valor autoincrement:");
  System.out.println("Agregando el Periodo Verano del 2021");
}
sentencia = conn.createStatement(
ResultSet.TYPE_SCROLL_SENSITIVE,
   ResultSet.CONCUR_UPDATABLE);
sql = "SELECT * FROM "+nomPeriodo;
rs = sentencia.executeQuery(sql);

// Creamos un registro nuevo en el ResultSet
rs.moveToInsertRow();
// Actualizamos campos
rs.updateInt("year",2021);
rs.updateInt("periodo",3);
// Agregamos registro en base de datos
rs.insertRow();
// Nos movemos al ultimo registro para obtener el valor
// del campo id_periodo
rs.last();
valorLlaveAutoIncremento = rs.getInt("id_periodo");

if (debug) {
   System.out.println("El id_periodo que le toco fue : "
     + valorLlaveAutoIncremento+"\n");
}
sentencia.close();
```

f) Coloque el siguiente código dentro del método `close()`. Este método cierra la conexión, si es que existe:

```
if (conn!=null) {
    conn.close();
}
```

Paso 8. Ejecute las pruebas de esta práctica dando click en ***ColocaDatosUsandoDataSourceTest*** con el botón derecho y seleccionado Run si está en Intellij o emitiendo el comando **`gradle test --tests poo2.prac07.ColocaDatosUsandoDataSourceTest`** desde la terminal de comando

Paso 9 (OPCIONAL Y NO NECESARIO). Si desea ver mensajes de lo que va haciendo su código puede ejecutar la clase ***MainPractica06_07*** dándole click con el botón derecho y seleccionado Run si está en Intellij o emitiendo el comando **gradle run** desde la terminal de comando.

La salida generada por ***MainPractica06_07*** debe ser algo como lo siguiente (las XXX representan su matrícula):

```
Iniciando creacion de tablas ...
Creando tabla carrera_XXX..
Creando tabla periodoescolar_XXX..
```

```
Colocando datos en tabla carrera_XXX:
Agregando carrera Ingenieria en Computacion ...
Agregando carrera Ingenieria de Software ...
Agregando carrera Ingenieria en Electronica Industrial ...

Colocando datos en tabla periodoescolar_XXX:
1a forma de obtener valor autoincrement:
Agregando el Periodo Ago-Dic del 2020...
El id_periodo que le toco fue : 1

2a forma de obtener valor autoincrement:
Agregando el Periodo Ene-Jun del 2021...
El id_periodo que le toco fue : 2

3a forma de obtener valor autoincrement:
Agregando el Periodo Verano del 2021
El id_periodo que le toco fue : 3
```

Paso 10. Verifique entrando a la consola de MySQL que la base de datos creada en el Paso 1 contenga las tablas `carrera_XXX` y `periodoescolar_XXX` y que tengan los registros insertados por las sentencias SQL ejecutadas en el método llenaTablaCarrera. (Nuevamente, recuerde que XXX es sustituido por su matrícula)

Paso 11. Haga un `git push` para subir los cambios a su repositorio privado

Recuerda que cada vez que hagas **`git push`** se realizarán automáticamente pruebas sobre tu programa para verificar si funciona correctamente. Verifica que las pruebas que se hacen no marquen errores. Recuerda que en la página de tu repositorio en la sección **Pull Requests**, se encuentra una subsección de nombre **Feedback**, donde podrás encontrar los resultados de las pruebas en la pestaña denominada **Check** (expandiendo la parte que dice **Run education/autograding@v1**), y cualquier comentario general que el profesor tenga sobre tu código en la pestaña **Conversation**.