CICLO FORMATIVO GRADO SUPERIOR CCFF GS

DESAROLLODO DE APLICACIONES WEB

Práctica Combinada Final 3Ev-BBDD-EEDD-DAW-C21-22

JESÚS SÁNCHEZ TORRES
BASE DE DATOS\ENTORONOS DE DESARROLLO
Profesor: Francisco Javier Jurado Jurado

CURSO 2021/2022

Torremolinos, Mayo de 2022

Indicé

Introducción 3
Explicación cambios realizado por versión 5
- Versión 1.0: 5
- Versión 2.0: 5
- Versión 3.0: 6
- Versión 3.1: 9
- Versión 4.0: 13
- Versión 4.1: 15
- Versión 5.0: 16

INTRODUCCIÓN

Vamos a trabajar con una aplicación, un proyecto Java, ya diseñada en parte y comentado en clase. El proyecto ya es funcional, pero podría mejorarse y podría tener algún error.
El proyecto lo componen, principalmente, 2 ficheros: **DBManager**, que implementa una clase **DBManager** que gestiona una Base De Datos en MySQL con JDBC, y **GestionClientes**, una aplicación java simple de consola que utiliza la Clase **DBManager.**

Al Proyecto le faltan unas cuantas cosas, probarlo con el framework JUnit, documentarlo, y generar su documentación con Javadoc, diseñar unos cuantos diagramas con el lenguaje de modelado UML para describir y terminar de documentar el proyecto y, lo más importante, MODIFICARLO para que haga unas cuantas cosas más y, de camino, intentar refactorizarlo o limpiarlo y eliminar cualquier código para dejarlo más limpio, más claro o entendible.

Pero claro, no queremos perder el proyecto de partida y queremos utilizar las posibilidades que nos ofrecen GitHub y Git para controlar la evolución del proyecto. Git, como sistema de Control de Versiones y GitHub, como repositorio y escaparate, nos van a permitir tener en nuestro repositorio público una clonación del proyecto de partida y tantas ramas como deseemos.

Requisitos:

- Crear una cuenta en **GitHub** y comunicársela al profe, téngase en cuenta que se utilizará para que este pueda ver la evolución del proyecto.
- Crear el proyecto en dicha cuenta y hacerlo público y gestionar la práctica a realizar utilizando las posibilidades que ofrece **GitHub**, **Git** y **Eclipse** en el desarrollo de la misma.
- Crear las **ramas** necesarias para mantener todas las **versiones** desde la inicial versión v1.0 hasta la final Vn.0 y todas sus posibles versiones previas. Se propone crear tantas ramas como versiones se quieran tener.
- Crear una rama o versión propia para el proyecto en su faceta Gráfica.
- Crear e incorporar un **Paquete** Java para la clase **DBManager** y cualquier otra que se considere en base a la evolución que tenga el proyecto dadas las modificaciones que sufra según se especifica en los próximos párrafos.
- Habrá que documentar “adecuadamente” el paquete, las clases, métodos, etc. con objeto de poder generar una buena documentación con la herramienta **Javadoc**. Habrá que identificar claramente al autor y las diferentes versiones .
- Habrá que generar con el framework de pruebas **JUnit** un conjunto de pruebas suficiente para acreditar que el proyecto cumple lo especificado.
- Crear los **Diagramas UML** Estructurales y de Comportamiento mínimos que se consideren o se especifiquen (al menos **Diagrama de Clases** y de **Paquetes**, y Diagrama **de Casos de Usos** y **Secuencia**).

- **Implementar algunas de las modificaciones** propuestas a continuación:
  - Modificar la Clase DBManager para utilizar la clase **PreparedStatement** en vez de **Statement** con objeto de mejorar el rendimiento haciendo los cambios necesarios en el código.
  - Añadir algún procedimiento almacenado y utilizarlo con la Clase **CallableStatement**. .
  - Modificar la Clase DBManager para que permita manejar cualquier Base de Datos y cualquier Tabla de dicha Base de Datos.
  - Modificar la Aplicación para Añadir varias opciones al menú:
    - Poder crear una Tabla nueva
    - Poder filtrar filas de una Tabla
  - Modificar la Aplicación para permitir volcar en un fichero una Tabla de una Base de Datos. El contenido del fichero deberá indicar en la primera línea el nombre de la Base de Datos Origen y el de la Tabla Origen, en la segunda las cabeceras de los campos y el las restantes líneas los campos de cada registro por línea.
  - Modificar la aplicación para permitir Insertar, Actualizar y Borrar registros a partir de la información contenida en un archivo cuyo contenido y estructura es conocida.
    - Ejemplo de estructura para **INSERTAR**:
      1. Primera línea contendrá ***el nombre de la Base de Datos*****.**
      2. Segunda línea contendrá ***el nombre de la Tabla***.
      3. Tercera línea contendrá separados por "," los ***nombres de los campos*** o sus ***posiciones***.
      4. Cuarta línea y siguientes contendrán ***los datos de los campos*** en el orden indicado en la tercera línea.
    - Ejemplo de estructura para **ACTUALIZAR** :
      1. Primera línea contendrá ***el nombre de la Base de Datos*****.**
      2. Segunda línea contendrá ***el nombre de la Tabla***.
      3. Tercera línea contendrá separados por "," los ***nombres de los campos*** o sus ***posiciones***.
      4. Cuarta línea y siguientes contendrán ***el identificador (clave)*** del registro a modificar y ***los datos de los campos*** en el orden indicado en la tercera línea.
    - Ejemplo de estructura para **BORRAR** :
      1. Primera línea contendrá ***el nombre de la Base de Datos*****.**
      2. Segunda línea contendrá ***el nombre de la Tabla***.
      3. Tercera línea contendrá ***el identificador clave del registro a borrar.***
      4. Tercera línea contendrán separados por "," ***los datos de los identificadores (clave)*** de los registros a borrar.

EXPLICACIÓN CAMBIOS REALIZADO POR VERSIÓN

- **Versión 1.0:**
  La versión 1.0 es la versión del proyecto original, donde podemos gestionar la tabla clientes de la bases de datos tienda.

- **Versión 2.0:**
  La versión 2.0 he modificado la clase DBManager para utilizar la clase PreparedStatement en vez de Statement, para ello los métodos getCliente y getTablaClientes que usaba la clase Statement los he cambiado con PreparedStatement para mejorar el rendimiento del programa.

```
public static ResultSet getCliente(int id) {
    try {
        // Realizamos la consulta SQL
            String sql = DB_CLI_SELECT + " WHERE " + DB_CLI_ID + "='" + id + "';";
        PreparedStatement stmt = conn.prepareStatement(sql,ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE);
        ResultSet rs = stmt.executeQuery();
        //stmt.close();

        // Si no hay primer registro entonces no existe el cliente
        if (!rs.first()) {
            return null;
        }

        // Todo bien, devolvemos el cliente
        return rs;

    } catch (SQLException ex) {
        ex.printStackTrace();
        return null;
    }
}
```

```
public static ResultSet getTablaClientes(int resultSetType, int resultSetConcurrency) {
    try {
        PreparedStatement stmt = conn.prepareStatement(DB_CLI_SELECT, resultSetType, resultSetConcurrency);
        ResultSet rs = stmt.executeQuery();
        return rs;
    } catch (SQLException ex) {
        ex.printStackTrace();
        return null;
    }

}
```

- **Versión 3.0:**
  La versión 3.0 he borrado todos los métodos para gestionar la tabla clientes y he realizado una serie de métodos para gestionar cualquier tabla de cualquier base de datos.
  El orden que he seguido para manejar cualquier tabla de cualquier base de datos es el siguiente:
  - He realizado un método llamado **nombreBaseDatos** que preguntara al usuario a que base de datos quiere conectarse. Este método nos conectara a la base de datos escrita por teclado.

```
public static String nombreBaseDatos()
{
        try {
            Scanner ent = new Scanner(System.in);
            System.out.println("A que base de datos quiere conectarse: ");
            DB_NAME = ent.nextLine();
            return DB_NAME;
        } catch (Exception e) {
            System.out.println("No has introducido una cadena de texto. Vuelve a intentarlo.");
            return null;
        }
}
```

  - He realizado un método llamado **verTablas** que nos mostrara los nombres de la tablas de la base de datos a la que estamos conectados. Con este metodo podré ver los nombres de las tablas de cualquier base de datos.

```
public static void verTablas()
{
      try {
              DatabaseMetaData datos = conn.getMetaData();
              System.out.println("Lista de tablas");
              ResultSet rs = datos.getTables(DB_NAME, null, null, null);

              while(rs.next())
              {
                      System.out.println(rs.getString("TABLE_NAME"));
              }
      } catch (SQLException ex) {
          ex.printStackTrace();
      }
}
```

- He realizado un método llamado **verCamposTabla** que nos mostrara los campos de la tabla que le indicemos como parametro. Con este método podré ver los campos de cualquier tabla de la base de datos a la que esté conectada.

```
public static boolean verCamposTabla(String tabla)
{
            contadorColumnas.clear();
    DatabaseMetaData datos;
            ResultSet rs;
            String nombreColumna;
            try {
                    datos = conn.getMetaData();
                    rs = datos.getColumns(DB_NAME, null, tabla, null);
                    while(rs.next())
                    {
                            nombreColumna = rs.getString("COLUMN_NAME");
                            contadorColumnas.add(nombreColumna);
                            System.out.printf(nombreColumna + "\t");
                    }
                    System.out.println();
                    return true;
            } catch (SQLException ex) {
                    ex.printStackTrace();
                    return false;
            }
}
```

- He realizado un método llamado **mostrarContenidoTabla** que nos mostrara los datos de los campos de la tabla que le indiquemos como parámetro. Con este método podré ver el contenido de cualquier tabla de la base de datos a la que esté conectada.

```
public static boolean mostrarContenidoTabla(String tabla)
{
            PreparedStatement stmt;
            ResultSet rs;
            try {
                    stmt = conn.prepareStatement("select * from " + tabla);
                    rs = stmt.executeQuery();
                    while (rs.next())
                    {
                            for (int i=0;i<contadorColumnas.size();i++)
                            {
                                    String contadorAux = contadorColumnas.get(i);
                                    System.out.printf(rs.getString(contadorAux) + "\t");
                            }
                            System.out.println();
                    }
                    return true;
            } catch (SQLException e) {
                    e.printStackTrace();
                    return false;
            }
}
```

- He realizado un método llamado **crearTabla** que creara una tabla con el nombre y el número de columnas le indicemos. Le pasaremos un método que pida el nombre de la columna y el tipo de dato. Con este método podremos crear una tabla en cualquier base de datos.

```
public static boolean crearTabla(String tabla, int numColumnas) {
    Scanner ent = new Scanner(System.in);
    String tipoCampo = null, nombreCampo = null, consulta,  informacion = "";
    String crearTabla[] = new String[numColumnas];
    int rs;
    PreparedStatement stmt;
    try {
                pedirDatosCreacionTabla(numColumnas, nombreCampo, tipoCampo, crearTabla);

                for (int i=0;i<crearTabla.length;i++)
                {
                        if (i == crearTabla.length-1)
                        {
                                informacion += crearTabla[i] + ");";
                        }
                        else
                        {
                                informacion += crearTabla[i] + ",";
                        }
                }
                consulta = "create table " + tabla + "(";
                stmt = conn.prepareStatement(consulta.concat(informacion));
                rs = stmt.executeUpdate();
                return true;
            } catch (SQLException e) {
                    e.printStackTrace();
                    return false;
            }
}
```

- He realiza un método llamado **pedirDatosCreaciónTabla**. Este método será invocado en el método de crear una tabla para pedir el nombre del campo y el tipo de dato que tendrá ese campo.

```
public static void pedirDatosCreacionTabla(int numColumnas, String nombreCampo, String tipoCampo, String crearTabla[])
{
    Scanner ent = new Scanner(System.in);
    for (int i=0;i<numColumnas;i++)
            {
                    System.out.println("Nombre del campo:");
                    nombreCampo = ent.nextLine();
                    System.out.println("Tipo Campo:");
                    System.out.println("Opciones: String | char | int | double | boolean");
                    tipoCampo = ent.nextLine();
                    switch (tipoCampo) {
                            case "String":
                            {
                                    crearTabla[i] = nombreCampo + " varchar(50)";
                                    break;
                            }
                            case "char":
                            {
                                    crearTabla[i] = nombreCampo + " char";
                                    break;
                            }
                            case "int":
                            {
                                    crearTabla[i] = nombreCampo + " int";
                                    break;
                            }
                            case "double":
                            {
                                    crearTabla[i] = nombreCampo + " double";
                                    break;
                            }
                            case "boolean":
                            {
                                    crearTabla[i] = nombreCampo + " boolean";
                                    break;
                            }
                            default:
                            {
                                    System.err.println("No existe el tipo de dato");
                            }
                    }
            }
}
```

- **Versión 3.1:**
  En la versión 3.1 sigo creando métodos para gestionar cualquier tabla de cualquier base de datos. He realizado métodos para modificar y borrar cualquier cualquiera tabla de una base de datos. Los métodos son los siguientes:
  - He realizado un método que llamara a un método un menú de modificación de tablas con varias opciones donde elegiremos una opción y este método de **modificarTabla** ejecuta la acción elegida en el menú.

```
public static boolean modificarTabla(String tabla)
{
    int opcion;
    opcion = menuModificarTabla();

    switch(opcion)
    {
            case 1:
            {
                    anadirNuevoRegistro(tabla);
                    return true;
            }
            case 2:
            {
                    cambiarTipoDatoRegistro(tabla);
                    return true;
            }
            case 3:
            {
                    renombarRegistro(tabla);
                    return true;
            }
            default:
            {
                    System.err.println("No existe esta opcion");
                    return false;
            }
    }
}
```

  - He realiza un método llamado **menuModificarTabla**. Este método será invocado en el método de modifcar tabla para mostrar las opciones que hay para modificar la tabla.

```
public static int menuModificarTabla()
{
    Scanner ent = new Scanner(System.in);
    int opcion;
    try
    {
            System.out.println("Que deseas hacer: ");
            System.out.println("1. Añadir nuevo registro");
            System.out.println("2. Cambiar tipo de dato registro");
            System.out.println("3. Renombrar columna");
            return opcion = ent.nextInt();
    }
    catch(InputMismatchException ex)
    {
            ex.printStackTrace();
    }
            return 0;
}
```

- He realizado un método **añadirNuevoRegistro** que añadirá un nuevo registro a una tabla indicada como parámetro. Con este método podremos añadir una nueva columna a cualquier tabla.

```
public static void anadirNuevoRegistro(String tabla)
{
    PreparedStatement stmt;
    int rs;
    String nombreCampo = null, tipoCampo = null, resultado;
    resultado = pedirDatosTabla(nombreCampo, tipoCampo);
    try {
                stmt = conn.prepareStatement("alter table " + tabla + " add " + resultado);
                rs = stmt.executeUpdate();
            } catch (SQLException e) {
                e.printStackTrace();
            }
}
```

- He realizado un método **cambiarTipoDatoRegistro** que modificará el tipo de dato de una columna de una tabla pasado como parámetro. Con este método podremos modificar cualquier columna de cualquiera tabla.

```
public static void cambiarTipoDatoRegistro(String tabla)
{
    PreparedStatement stmt;
    int rs;
    String nombreCampo = null, tipoCampo = null, resultado;
    resultado = pedirDatosTabla(nombreCampo, tipoCampo);
    try {
                stmt = conn.prepareStatement("alter table " + tabla + " modify " + resultado);
                rs = stmt.executeUpdate();
            } catch (SQLException e) {
                e.printStackTrace();
            }
}
```

- He realizado un método **renombrarRegistro** que renombrara el nombre de una columna de una tabla pasado como parámetro. Con este método podremos renombrar cualquier columna de cualquiera tabla.

```
public static void renombarRegistro(String tabla)
{
        PreparedStatement stmt;
        Scanner ent = new Scanner(System.in);
        int rs;
        String nuevonombreCampo , viejonombreCampo;
        System.out.println("Nombre de campo a modificar: ");
        viejonombreCampo = ent.nextLine();
        System.out.println("Nuevo nombre: ");
        nuevonombreCampo = ent.nextLine();
        try {
            stmt = conn.prepareStatement("alter table " + tabla + " change " + viejonombreCampo + nuevonombreCampo);
            rs = stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
}
```

- He realizado un método que llamara a un método un menú de borrado de tablas con varias opciones donde elegiremos una opción y este método de **borradoTabla** ejecuta la acción elegida en el menú.

```
public static boolean borradoTabla(String tabla)
{
    int opcion;
    opcion = menuBorradoTabla();

    switch(opcion)
    {
            case 1:
            {
                    borrarTablaEntera(tabla);
                    return true;
            }
            case 2:
            {
                    borrarCampoTabla(tabla);
                    return true;
            }
            default:
            {
                    System.err.println("No existe esta opcion");
                    return false;
            }
    }
}
```

- He realiza un método llamado **menuBorradoTabla**. Este método será invocado en el método de borrado tabla para mostrar las opciones que hay para borrar la tabla.

```
public static int menuBorradoTabla()
{
    Scanner ent = new Scanner(System.in);
    int opcion;
    try
    {
            System.out.println("Que deseas hacer: ");
            System.out.println("1. Eliminar tabla");
            System.out.println("2. Eliminar campo de tabla");
            return opcion = ent.nextInt();
    }
    catch(InputMismatchException ex)
    {
            ex.printStackTrace();
    }
            return 0;
}
```

- He realizado un método **borrarTablaEntera** que borrara una tabla pasado como parámetro. Con este método podremos borrar cualquier tabla de cualquiera base de datos.

```
public static void borrarTablaEntera(String tabla)
{
    PreparedStatement stmt;
            int rs;
            try {
                    stmt = conn.prepareStatement("drop table " + tabla);
                    rs = stmt.executeUpdate();
            } catch (SQLException e) {
                    e.printStackTrace();
            }
}
```

- He realizado un método **borrarCampoEntera** que borrara una columna de una tabla pasado como parámetro. Con este método podremos borrar cualquier columna de una tabla de cualquiera base de datos.

```
public static void borrarCampoTabla(String tabla)
{
    PreparedStatement stmt;
            Scanner ent = new Scanner(System.in);
            String campo;
            System.out.println("Introduce el nombre de la columna a eliminar: ");
            campo = ent.nextLine();
            int rs;
            try {
                    stmt = conn.prepareStatement("alter table " + tabla + " drop " + campo);
                    rs = stmt.executeUpdate();
            } catch (SQLException e) {
                    e.printStackTrace();
            }
}
```

- **Versión 4.0:**
  En la versión 4.0 he creado métodos para volcar en un fichero una tabla de una base de datos. El contenido del fichero contendrá en la primera línea el nombre de la base de datos y el de la tabla, en la siguiente las cabeceras de los campos y los siguientes los campos de cada registro por línea. También he implementado un método para leer un fichero que contendrá una estructura para insertar datos en la tabla de la base de datos. Los métodos son los siguientes:
  - He realizado un método **volcarTablaFichero** que volcara en un fichero una tabla pasado como parámetro. Con este método podremos volcar cualquier tabla de cualquiera base de datos en un fichero.

```
public static boolean volcarTablaFichero(String tabla) {
    Scanner ent = new Scanner(System.in);
    String ruta = null , sql;

    try {
            System.out.println("Escribe la ruta donde quieres volcar la informacion");
            ruta = ent.nextLine();

            File archivo = new File(ruta);
            FileWriter archivoEscritura = new FileWriter(archivo);
                    archivoEscritura.write(DB_NAME + " " + tabla + "\n");
                    archivoEscritura.write(mostrarContenidoTabla(tabla));
                    archivoEscritura.close();
                    return true;
    } catch (FileNotFoundException ex) {
            ex.printStackTrace();
            return false;
            } catch (IOException ex) {
                    ex.printStackTrace();
            return false;
            }
}
```

  - He realizado un método **insertarDatosDesdeFichero** que insertara datos desde un fichero pasado como parámetro a una tabla de una base de datos. Con este método podremos insertar cualquier datos en una tabla de de cualquiera base de datos.

```
public static boolean insertarDatosDesdeFichero(String ruta)
{
    String array[] = leerDatosFichero(ruta);
    String informacion = "";
    String nombreBaseDatos = array[0];
    String nombreTabla = array[1];
    String nombreCampos = array[2];

    for (int j=3;j<array.length;j++)
    {
            String datos[] = array[j].split(",");
            for (int i=0;i<datos.length;i++)
            {
                    if (i == datos.length-1)
                    {
                            informacion += "'" + datos[i] + "'";
                    }
                    else
                    {
                            informacion += "'" + datos[i] + "',";
                    }
            }
                    PreparedStatement miStatement;
                            try {
                                    miStatement = conn.prepareStatement("insert into " + nombreBaseDatos + "." + nombreTabla + "(" + nombreCampos + ") values(" + informacion + ");" );
                                    int setencia = miStatement.executeUpdate();
                            } catch (SQLException e) {
                                    e.printStackTrace();
                                    return false;
                            }
                    informacion = "";
    }
    return true;
}
```

- He realizado un método **leerDatosFichero** que leerá y guardara los datos desde un fichero pasado como parámetro. Este método será invocado en los métodos de insertar, modificar y borrar desde un fichero para que devuelva los datos del fichero.

```
public static String[] leerDatosFichero(String ruta)
{
    String informacion;
    Scanner lectorFichero = null;
    Scanner lectorFichero1 = null;
    File fichero = new File(ruta);
    String nombreBaseDatos, nombreTabla, nombreCampos, datosCampos;
    int numLineas = 0;
            try {
                    lectorFichero = new Scanner(fichero);
                    lectorFichero1 = new Scanner(fichero);

            } catch (FileNotFoundException e) {
                    e.printStackTrace();
            }

            do {
                    informacion = lectorFichero1.nextLine();
                    numLineas++;
            }
            while(lectorFichero1.hasNext());
            lectorFichero1.close();

            String array[] = new String[numLineas];
            numLineas = 0;
            do {
                    informacion = lectorFichero.nextLine();

                    array[numLineas] = informacion;
                    numLineas++;
            }
            while(lectorFichero.hasNext());
            lectorFichero.close();
            return array;
}
```

- **Versión 4.1:**
  En la versión 4.1 he creado métodos para modificar y borrar los datos de una tabla a través de los datos del fichero. También he realizado un método para cambiarnos de base de datos. Los métodos son los siguientes:
  - He realizado un método **modificarDatosDesdeFichero** que modificara los datos desde un fichero pasado como parámetro a una tabla de una base de datos. Con este método podremos modificar cualquier dato de una tabla de cualquiera base de datos.

```
public static boolean modificarDatosDesdeFichero(File ruta)
{
    String array[] = leerDatosFichero(ruta);
    String informacion = "";
    String nombreBaseDatos = array[0];
    String nombreTabla = array[1];
    String nombreCampos[] = array[2].split(",");
    PreparedStatement miStatement;
    int setencia;

            for (int j=3;j<array.length;j++)
            {
                    String datos[] = array[j].split(",");
                    for (int i=1;i<datos.length;i++)
                    {
                            informacion = "'" + datos[i] + "'";
                            try {
                                    miStatement = conn.prepareStatement("update " + nombreBaseDatos + "." + nombreTabla + " set " + nombreCampos[i] + " = " + informacion + " where "
                                    setencia = miStatement.executeUpdate();
                            } catch (SQLException e) {
                                    e.printStackTrace();
                                    return false;
                            }
                    }
    }
            return true;
}
```

  - He realizado un método **borrarDatosDesdeFichero** que borrara los datos desde un fichero pasado como parámetro a una tabla de una base de datos. Con este método podremos borrar cualquier dato de una tabla de cualquiera base de datos.

```
public static boolean borrarDatosDesdeFichero(File ruta)
{
    String array[] = leerDatosFichero(ruta);
    String nombreBaseDatos = array[0];
    String nombreTabla = array[1];
    String campoClave = array[2];
    String identificadorClave[] = array[3].split(",");
    String informacion = "";
    PreparedStatement miStatement;
    int setencia;

    for (int i=0;i<identificadorClave.length;i++)
    {
            informacion += "'" + identificadorClave[i] + "'" ;
            try {
                            miStatement = conn.prepareStatement("delete from " + nombreBaseDatos + "." + nombreTabla + " where " + campoClave + " = " + informacion + ";");
                            setencia = miStatement.executeUpdate();
                            informacion = "";
                    } catch (SQLException e) {
                            e.printStackTrace();
                            return false;
                    }
    }
    return true;
}
```

- He realizado un método **cambiarBaseDatos** que cambiara a una base de datos pasada como parámetro. Con este método podremos cambiar a cualquier base de datos.

```
public static boolean cambiarBaseDatos(String BaseDatos)
{
    PreparedStatement stmt;
    int rs;
     DB_NAME = BaseDatos;
    try {
                stmt = conn.prepareStatement("use " + BaseDatos);
                rs = stmt.executeUpdate();
            } catch (SQLException e) {
                e.printStackTrace();
                return false;
            }
    return true;
}
```

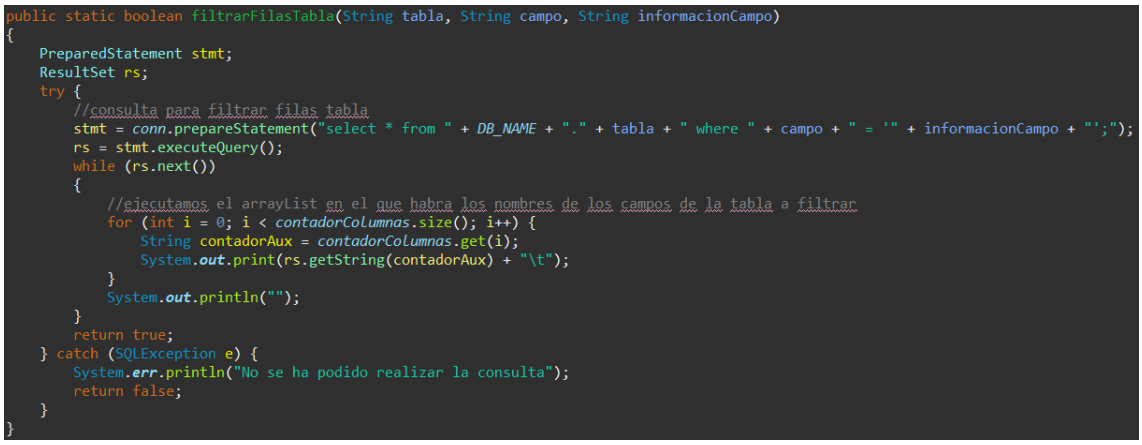
- **Versión 5.0:**
  En la versión 5.0 he creado métodos para llamar a 2 procedimientos almacenados prestablecidos. También he realizado un método para filtrar filas de una tabla. Los métodos son los siguientes:
  - He realizado un método **procedimientoAlmacenadomostrarProductoFabricante** donde llamo al procedimiento almacenado de mostrar los productos del fabricante, como fabricante pasado como parámetro**.** Con este método podremos ver los productos de un fabricante concreto.

```
public static boolean procedimientoAlmacenadomostrarProductoFabricante(String fabricante)
{
    java.sql.CallableStatement procedimiento;
    ResultSet setencia;
    try {
        //consulta para llamar a un procedimiento
        procedimiento =  conn.prepareCall("{call " + DB_NAME +  ".mostrarProductoFabricante('" + fabricante + "')}");
        setencia = procedimiento.executeQuery();
        while (setencia.next())
        {
            System.out.println(setencia.getString(1));
        }
        return true;
    } catch (SQLException e) {
        System.err.println("No se ha podido ejecutar el procedimiento almacenado");
        return false;
    }
}
```

  - He realizado un método **procedimientoAlmacenadoMostrarClientes** donde llamo al procedimiento almacenado de mostrar los clientes**.** Con este método podremos ver todos los clientes de la tabla clientes.

```
public static boolean procedimientoAlmacenadoMostrarClientes() {
    java.sql.CallableStatement procedimiento;
    ResultSet setencia;
    try {
        //consulta para llamar  a un procedimiento
        procedimiento =  conn.prepareCall("{call " + DB_NAME + ".mostrarClientes()}");
        setencia = procedimiento.executeQuery();
        while (setencia.next())
        {
            System.out.println(setencia.getInt(1) + " " + setencia.getString(2) + " " + setencia.getString(3));
        }
        return true;
    } catch (SQLException e) {
        System.err.println("No se ha podido ejecutar el procedimiento almacenado");
        return false;
    }
}
```

- He realizado un método **filtrarFilasTabla** donde podremos cualquier fila de una tabla con una tabla,un campo y la información de esa campo pasada como parámetro**.** Con este método podremos filtrar filas de cualquier tabla de cualquiera base de datos.



```
public static boolean filtrarFilasTabla(String tabla, String campo, String informacionCampo)
{
    PreparedStatement stmt;
    ResultSet rs;
    try {
        //consulta para filtrar filas tabla
        stmt = conn.prepareStatement("select * from " + DB_NAME + "." + tabla + " where " + campo + " = '" + informacionCampo + "';");
        rs = stmt.executeQuery();
        while (rs.next())
        {
            //ejecutamos el arrayList en el que habra los nombres de los campos de la tabla a filtrar
            for (int i = 0; i < contadorColumnas.size(); i++) {
                String contadorAux = contadorColumnas.get(i);
                System.out.print(rs.getString(contadorAux) + "\t");
            }
            System.out.println("");
        }
        return true;
    } catch (SQLException e) {
        System.err.println("No se ha podido realizar la consulta");
        return false;
    }
}
```