---Teoria---

Definicion de la teoria: JDBC (Java DataBase Connectivity) es la tecnología Java que permite a las aplicaciones

interactuar directamente con motores de base de datos relacionales.

Definicion de internet: JDBC (Java Database Connectivity) es una interfaz de programación de aplicaciones (API) de Java que permite a las aplicaciones escribir en este lenguaje conectarse a bases de datos y ejecutar consultas SQL para manipular datos. Actúa como un puente, traduciendo las operaciones de la aplicación Java al lenguaje y protocolo entendidos por la base de datos, gracias a los controladores JDBC específicos de cada sistema de gestión de bases de datos ([SGBD](https://www.google.com/search?sca_esv=764700c983d0c1df&sxsrf=AE3TifOIIITgDRl5CfB3MgIT2DoYsL5NAA:1759273400684&q=SGBD&sa=X&ved=2ahUKEwif8vCQzIGQAxX7rJUCHVhDKDQQxccNegQILRAB&mstk=AUtExfBKrjyrmFv74K0Djvbfqp4S0dHZnHh8JUy8l8HGHVNLYMG5ANCbAP75ZM1tWo6uKnnpj8CtX0Bs6ASBUBl-OTctts3rGh3z1dnDfUufirUVQu-Y2voRuxjDbTmcq1_Lrv0IGyVKq5ROgCC8bDyRNicyr16drpVufsrHp7lCC4ybf9ROt-3gV_06hTqUe5-XA5-z-wmEh0TnJCc33JCeBoou2vEbNGD6kAH7ONDzcQfiC0a7FLiVxLuPsDilEwMM2JBgLVhlRSfVaHSlDb1qeGb6&csui=3" \t "https://www.google.com/_blank)).

N: Java Database Connectivity viene ya con la plataforma java

driver: Un driver JDBC es usado por la JVM (Java Virtual Machine) para introducir las invocaciones

JDBC en invocaciones que la base de datos entiende....

... jdbc consta de un conjunto de clases e interfaces Java que nos permiten acceder de una forma genérica a las bases de datos independientemente del proveedor del SGBD

-----------------------

|| COMPONENTES ||

-----------------------

- El gestor de drivers (java.sql.drivermanager)

- La conexion a la base de datos (java.sql.connection)

- La sentencia a ejecutar (java.statement)

- El resultado (java.sql.ResultSet)

Básicamente una aplicación que utiliza JDBC realiza los siguientes pasos

a. Establece una conexión con una base de datos

b. Crea y envía una sentencia SQL a la base de datos

c. Procesa el resultado

-/-/-/-/-/-/-/-/-/ pasos /-/-/-/-/-/-/-/-/-

La conexion se establece con 2 simples sentencias:

Class.forname(“driver”)

DriverManager.GetConnection (url,username,password)

Pasos para implementar la conexion jdbc - sql

1. Añadir a librerias del proyecto el jar de la base de datos
2. new Class.forname(“”) { Explicacion: instancia una clase de tipo (esta) }

Class.forName(“org.mariadb.jdbc.Driver”)

Se encierra con un catch en caso de class.forname no haya encontrado una clase (es como decir que no encuentra al objeto que se quiera instanciar, en este caso es una clase)

N: aclara la existencia de este driver para que lo use luego con la clase conection

N: le dice al neatbens como tiene que comunicarse con el netbeans cuando le toque usar el metodo : Connectionconn=DriverManager.getConnection

("jdbc:mariadb://localhost:3306/miBD", "user", "pass");

(en caso de que no funcione importar)

import java.sql.DriverManager;

import java.sql.Connection;

1. DriverManager.getConnection() {Explicacion:}

DriverManager.getConnection("jdbc:mariadb://localhost:3306

/pruebaconecion","root",""

Java busca un driver que se pueda conectar con la base de datos

Si se pudo conectar devuelve un connection sino lanza una sqlExce N: es el enlace o puente que conecta a java con la base de datos, permanece abierta

Connection conexion = DriverManager... ¿POR QUE?

El driver manager devuelve un objeto de tipo connecion por eso se guarda en una variable de tipo connection. Para luego usar esa variable para usar metodos

¿Que hace un Conection?

Puede usar un metodo llamado CreateStatement(), prepareStatement()

Explicacion hasta ahora (pasos 1-3)

Ya creada la clase de conexion que usa el metodo del driver manager get.connection, tenemos un metodo en una clase (que generalmente se suele llamar Conexion) que se puede utilizar en otras clases u otras partes del programa (en caso de que el metodo getconection sea static)

Connection con = Conexion.getconexion

1. Utilizar ese metodo conexion de la clase Conexion en otras clases

En la clase que se quiera utilizar la conexion con la base de datos

Se crea un atributo de tipo conecion

Public class ClaseX {

Private connecion con = null;

}

este mismo connection no pose una clase Conexion configurada, por lo que se enlaca la clase Conexion con el connection en el constructor de la clase.

Public ClaseX(){

con = Conexion.getConexion();

}

1. Crear una Consulta a a la base de datos
2. rear el metodo ej = public void guardaralumno(Alumno alumno){}
3. crear un String llamado sql con la consulta ej= insert into alumno()...
4. en los values se colocan los caracteres comodines(?,?,?)

estos comodines tienen que ser tantos como los atributos

1. Crear un PreparedStatement ps con.prepareStatement

a ese statement se le pasa por parametro el (sql)

1. Setear con ps.set(posicion\_num\_del\_atr, alumno.getdni)

se usa los metodos set del ps y get del argumento

1. ps.executeUpdate/ps.executeQuery para ejecutar la consulta
2. Cambiar un valor de un atributo por informacion de una base de datos

Como extra se crea el result set para guardar las tablas introduc.

ResultSet rs= ps.getGeneratedKeys();

If(rs.next) si de las columnas introducidas hay una tupla

{alumno.setIdAlumno(rs.getInt(1))} se le va a setear el id alimno

1. Opcional, where atributo = ?

asi se puede poner una condicion con un valor que se puede especificar usando get o otro valor

1. Opcional. Cerrar ps.close()

N: date value of (una fecha) para convertir una fecha en tipo date

1. Opcional. ps.executeUpdate devuele la cantidad de tuplas insertadas

las cuales se pueden guardar int introducidas = ps.executeUpd...