# Aplicaciones Empresariales I

## Esquema de Notas

| | | |
|---|---|---|
| Ex. Parcial | → | 20% |
| Ex. Final | → | 30% |
| Pr. Pract. | → | 20% |
| Proy. Final | → | 30% |

**Proyecto Final** (30%)

| | | |
|---|---|---|
| Avance 1 | → | 20% |
| Avance 2 | → | 20% |
| Avance 3 | → | 20% |
| Av. Final y Sustentación | → | 40% |

Si no exponen la nota de proyecto es CERO

emaravi@cjavaperu.com

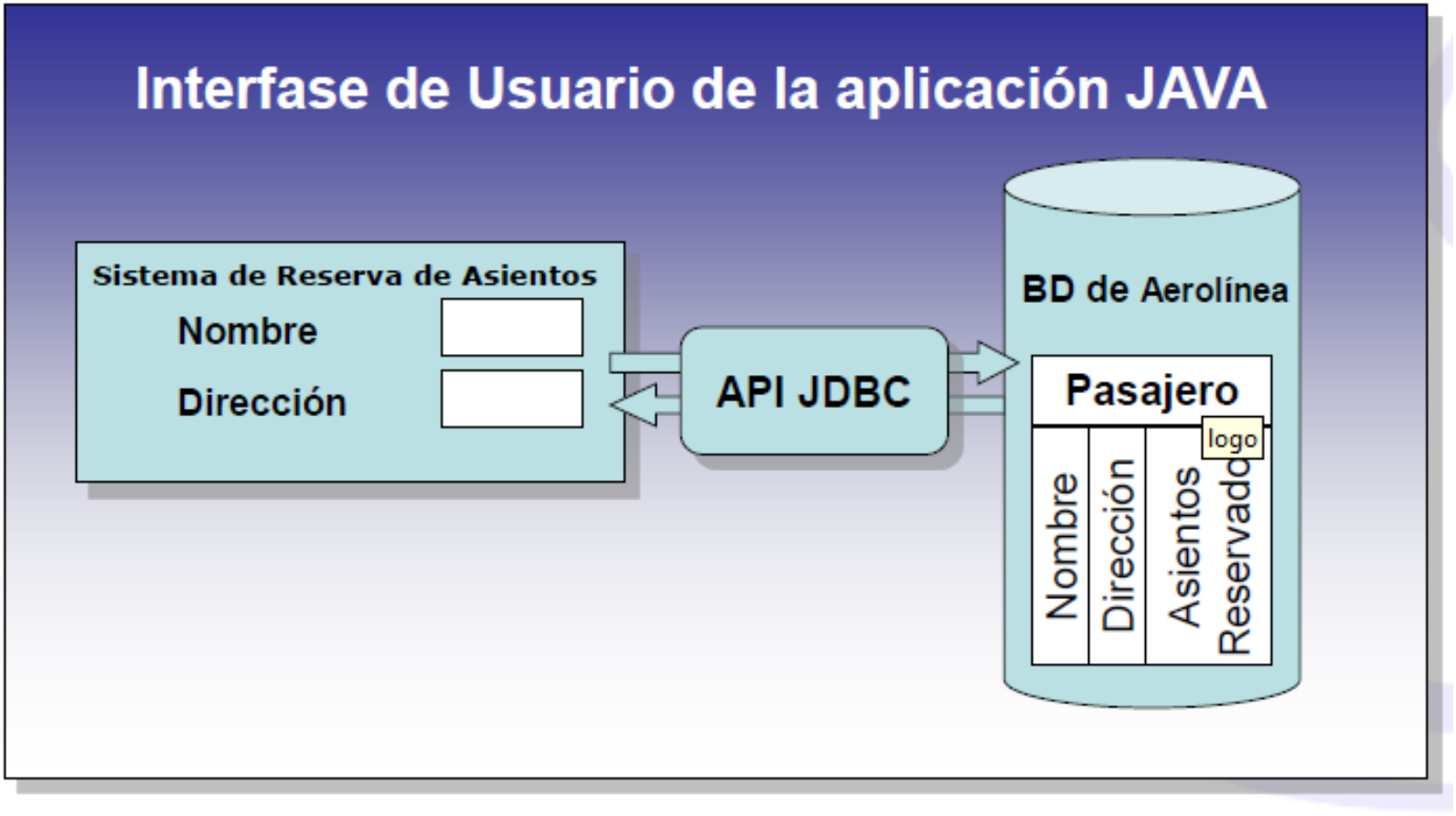
ISIL APRENDE HACIENDO

# Arquitectura de JDBC

# Conectividad de Base de Datos

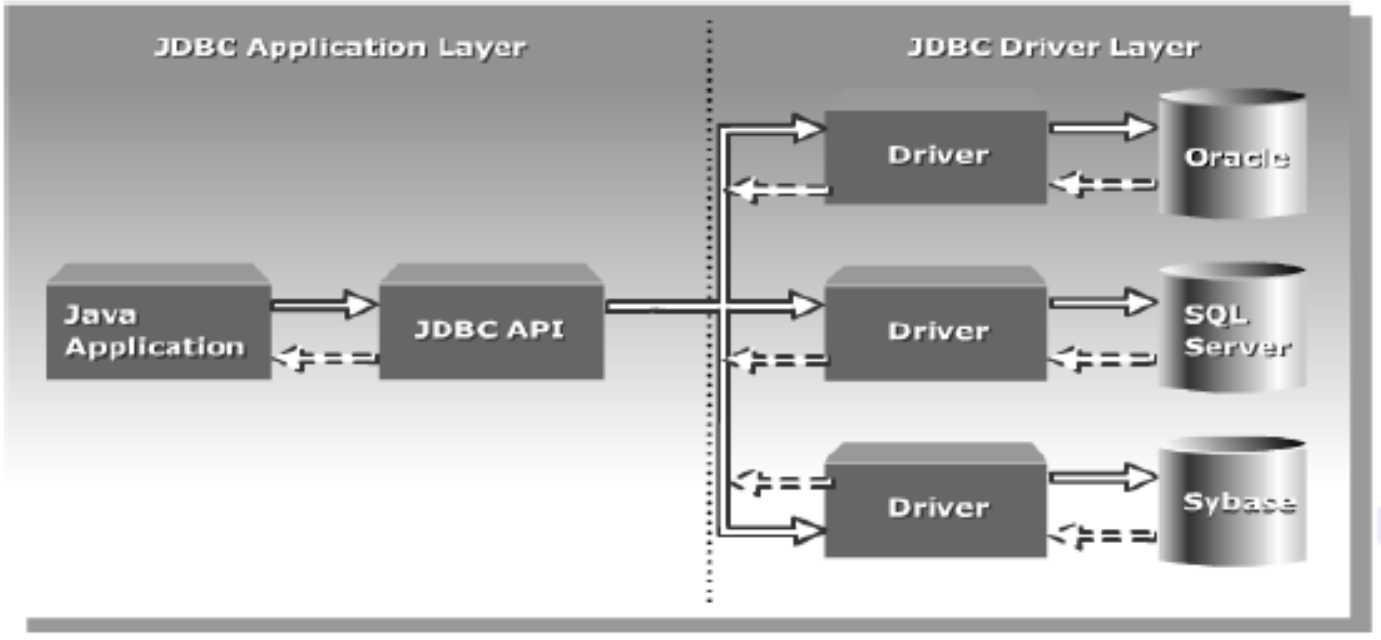
- Sun Microsystems ha incluído la API JDBC como parte del J2SDK para desarrollar aplicaciones Java que puedan comunicarse con bases de datos.
- La siguiente figura muestra el sistema de reservas de aerolineas desarrollado en Java interactuando con la base de datos de Aerolineas usando la API JDBC:

# Conectividad de Base de Datos



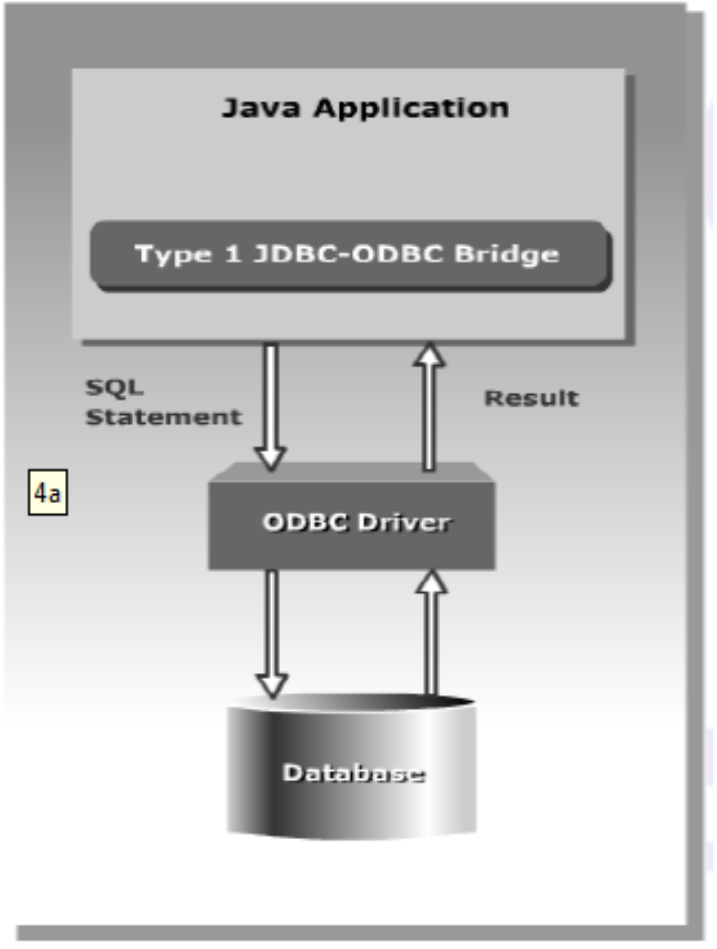
# Conectividad de Base de Datos

- Arquitectura JDBC :
  - Provee el mecanismo para traducir sentencias Java en sentencias SQL.
  - Puede ser clasificada en 2 capas:
    - Capa de aplicación JDBC.
    - Capa de driver JDBC



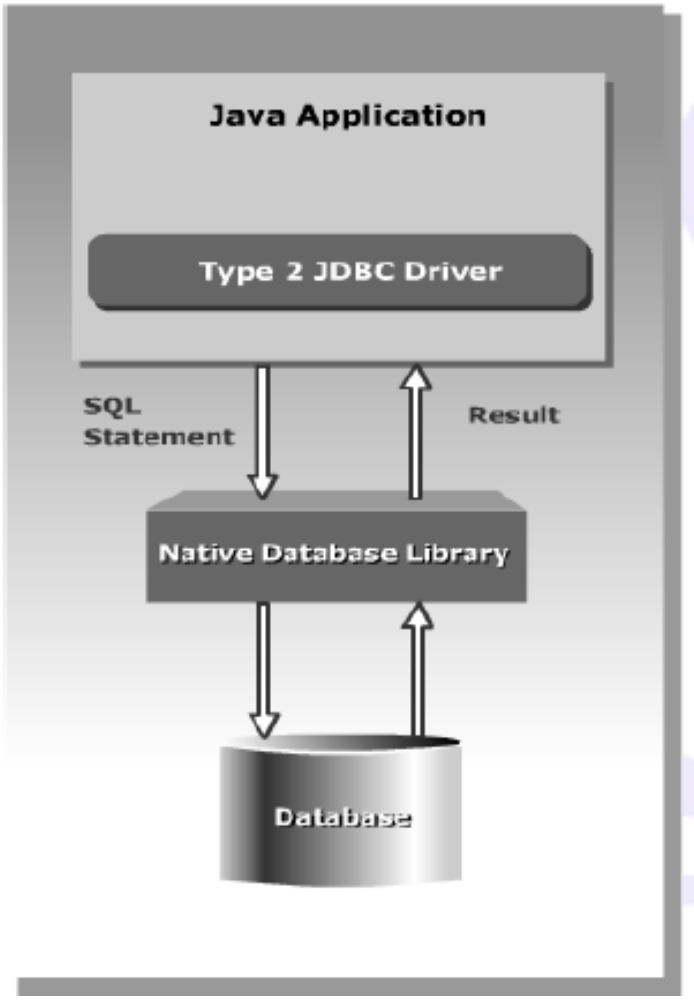
# Conectividad de Base de Datos

- Driver puente JDBC-ODBC



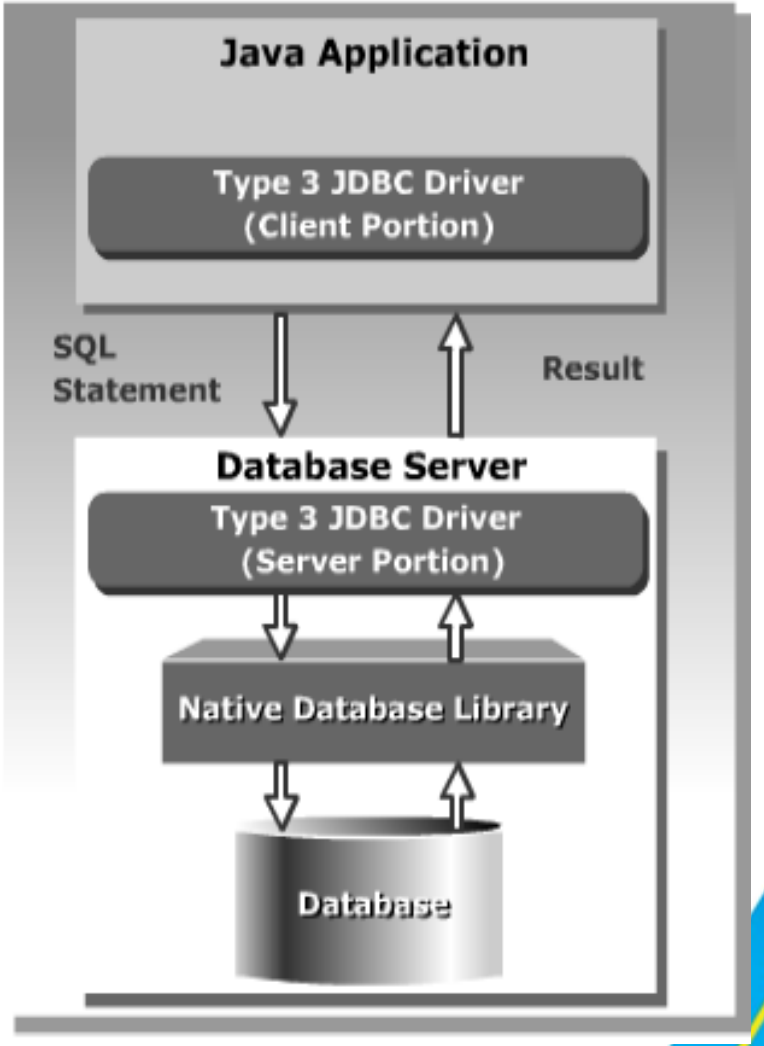
# Conectividad de Base de Datos

- Driver Nativo-API Partly-Java



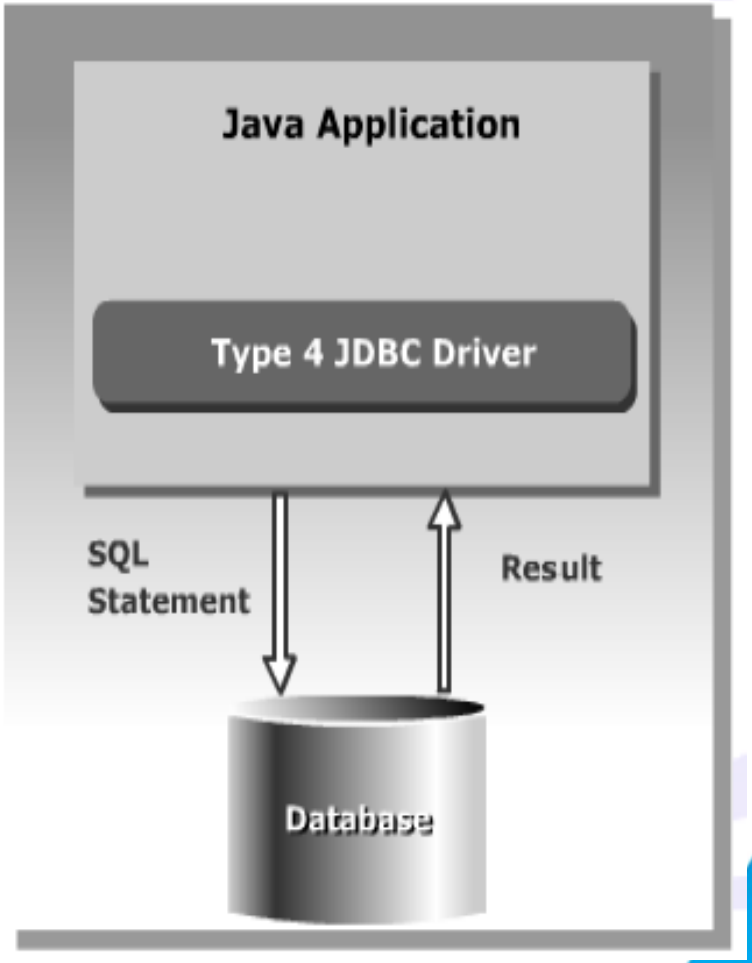
# Conectividad de Base de Datos

- Driver JDBC-Net Pure-Java



# Conectividad de Base de Datos

- Driver Nativo Protocol Pure-Java



# Drivers JDBC

- **Genérico de ODBC** - sun.jdbc.odbc.JdbcOdbcDriver
- **MySQL** - com.mysql.jdbc.Driver
- **Oracle** - oracle.jdbc.driver.OracleDriver
- **SQLServer** - com.microsoft.jdbc.sqlserver.SQLServerDriver

**Ahora te toca.**

**Busca en internet las cadenas de conexión de las siguien**

- Postgress
- Derby
- DB2

# Usando la API JDBC

- Las clases e interfaces de la API JDBC estan en los paquetes `java.sql` y `javax.sql` .
- Las clases mas usadas son::
  - **`Clase DriverManager`**: carga el driver para la base de datos.
  - **Interface `Driver`** : representa un driver de base de datos. Todas las clases del driver JDBC deben implementar la interface `Driver`.
  - **interface `Connection`** : Le da la capacidad para establecer una conexión entre la aplicación y la base de datos.
  - **interface `Statement`** : Le da la cpacacidad de ejecutar sentencias SQL.
  - **interface `ResultSet`** : Representa la información extraida de la base de datos.
  - **`Clase SQLException`**: Provee la inforación de las excepciones que sucedan durante la interacción con la base de datos..

# Usando la API JDBC

- Usando el método `forName()`
  - El método **`forName`**`()` esta en la clase `java.lang.Class`
  - El método `forName()` carga el driver JDBC driver y registra el driver con el Driver Manager.
  - La forma de usar el método es:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

# Cargando Drivers

Para cargar el driver JDBC se usa:

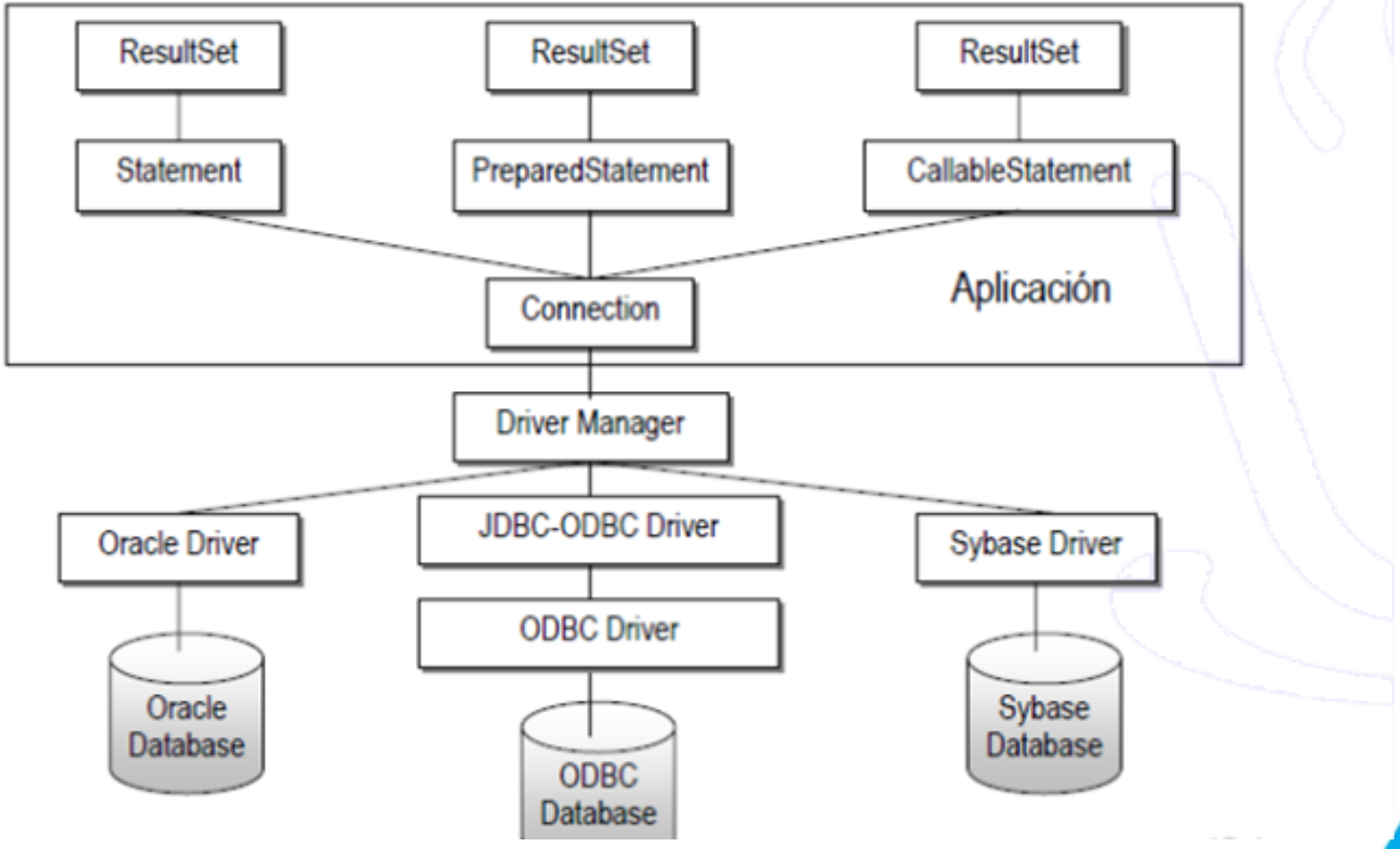
```
//Cargar el Driver JDBC-ODBC
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");


//Cargar el Driver para MySQL
Class.forName("com.mysql.jdbc.Driver");


//Cargar el Driver para Oracle
Class.forName("oracle.jdbc.driver.OracleDriver");
```

# Java DataBase Conectivity

- Es un conjunto de clases e interfaces que permiten la ejecución de sentencias SQL contra una base de datos.
- El paquete que contiene la API es **java.sql** (JSE) y **javax.sql** (JEE)



# URLs para JDBC

Los URL son cadenas que representa la información de la base de dat
que va a utilizar para establecer la conexión.

```
//URL para JDBC-ODBC (Usando ODBC debe crearse dsnBD1)
String URL = "jdbc:odbc:<dsn>";

//URL para MySQL
String URL = "jdbc:mysql://<host>:<port>/<bd>";

//URL para Oracle
String URL = "jdbc:oracle:thin:@<host>:<port>:<sid>";

//URL para Microsoft SQL Server
String URL="jdbc:microsoft:sqlserver://<host>:<port>";
```

# Driver Manager

La clase DriverManager usa el método **getConnection** para establecer la conexión a la BD, este devuelve un objeto **Connection** que contiene la conexión activa a la base de datos. Una alternativa a DriverManager es usar **DataSource**, paquete **javax.sql.DataSource**

```
//Importación
import java.sql.Connection;
import java.sql.DriverManager;

//Objeto Connection
Connection cn;

//Cargar el Driver
Class.forName("com.mysql.jdbc.Driver");

//URL para MySQL
String URL = "jdbc:mysql://localhost:3306/BD";

//Establecer la conexión a la BD
cn = DriverManager.getConnection(URL, "user", "pass");
```

# Usando la API JDBC

Creando y ejecutando sentencias JDBC

- El objeto `Connection` provee el método `createStatement()` que crea el objeto `statement`
- Puede ejecutar sentncias SQL estáticas pra enviar solicitudesa bases de datos para recibir resultados
- La interfase `Statement` contiene los siguientes métodos para enviar sentencias SQL estáticas a una base de datos:
  - `ResultSet executeQuery(String str)`
  - `int executeUpdate(String str)`
  - `boolean execute(String str)`

# Usando la API JDC

Las operaciones que ustede puede realizar usando una aplicación JAVA son:

- Consulta de tablas
- Inserción de registros a una tabla
- Actualización de registros de una tabla
- Eliminar registros de una tabla
- Crear una tabla
- Alterar y eliminar tablas.

# Ejercicios

- Generar el código necesario para hacer una consulta a la tabla Curso de la base de Datos Escuela.

- Genero el código necesario par poder insertar un registro a la tabla Curso.

ISIL APRENDE HACIENDO

# Mantenimiento de Bases de Datos con JAVA JDBC

# Interacción con bases de datos

- Usamos objetos Statement para
  - Consultas a la base de datos
  - Mantenimientos a la base de datos
- Usaremos :

  Statement, PreparedStatement, CallableStatement
- Para crearlas necesitamos de una Connection

# Consultas con Statement

```
String queryStr =
        "SELECT * FROM employee " +
        "WHERE lname = 'Wong'";

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(queryStr);
```

- El método executeQuery retorna un objeto ResultSet que represetna el resultado de la consulta.

# Mantenimiento de BD con Statement

```
String deleteStr =
        "DELETE FROM employee " +
        "WHERE lname = 'Wong'";

Statement stmt = con.createStatement();
int delnum = stmt.executeUpdate(deleteStr);
```

- executeUpdate se usa para manipular bases de datos: insert, delete, update, create table, etc.
- executeUpdate retorna el número de filas afectadas con la sentencia

# Prepared Statements

- Prepared Statements usada para consultas usadas varias veces.
- Son interpretadas(compiladas) por el motor de base de datos una sola vez.
- El valor de las columnas puede ser seteado después de la compilación
- En lugar de los valores , use '?'

# Consutas con PreparedStatement

```
String queryStr =
        "SELECT * FROM employee " +
        "WHERE superssn= ? and salary > ?";

PreparedStatement pstmt =
        con.prepareStatement(queryStr);

pstmt.setString(1, "333445555");
pstmt.setInt(2, 26000);

ResultSet rs = pstmt.executeQuery();
```

# Actualizaciones con PreparedStatement

```
String deleteStr =
        “DELETE FROM employee " +
        "WHERE superssn = ? and salary > ?";

PreparedStatement pstmt =
        con.prepareStatement(deleteStr);

pstmt.setString(1, "333445555");
pstmt.setDouble(2, 26000);

int delnum = pstmt.executeUpdate();
```

# Statements vs. PreparedStatements:

```
String val = "abc";
PreparedStatement pstmt =
        con.prepareStatement("select * from R where A=?");
pstmt.setString(1, val);
ResultSet rs =  pstmt.executeQuery();
```

```
String val = "abc";
Statement stmt =  con.createStatement( );
ResultSet rs =
        stmt.executeQuery("select * from R where A=" + val);
```

# Statements vs. PreparedStatements!

- Funcionará?

```
PreparedStatement pstmt =
        con.prepareStatement("select * from ?");


pstmt.setString(1, myFavoriteTableString);
```

- No!!! un "?" solo puede ser usado para representar valores de campos

# Ejercicios

1. Desarrollar una aplicación que permita ingresar al sistema usando usuario y password de una tabla de base de datos con:
   - Statement
   - PreparedStatement
   - CallabledStatement

2. Desarrolle una aplicación que permita hacer el mantenimiento de una tabla de base de datos usando Stored Procedures.

ISIL APRENDE HACIENDO

# Mantenimiento de Bases de Datos con JAVA JDBC

# Interacción con bases de datos

- Usamos objetos Statement para
  - Consultas a la base de datos
  - Mantenimientos a la base de datos
- Usaremos :

  Statement, PreparedStatement, CallableStatement
- Para crearlas necesitamos de una Connection

# Un procedimiento Almacenado

Crear un procedimiento almacenado:

```
CREATE PROCEDURE sp_ingreso( in param1 CHAR(10))
    select * from administrador where chrAdmLogin=param1;
```

Probar un procedimiento almacenado desde MySQL:

```
set @parametro = 'admin';
call sp_ingreso(@parametro);
```

# Usando CallableStatement

El código java necesario para ejecutarlos sería el siguiente

```
CallableStatement cs = connection.prepareCall("{CALL sp_ingreso(?)}");

//Aca va el texto que se necesita pasar como parametro al sp
callableStatement.setString(1,"");

ResultSet rs = cs.executeQuery();

while(rs.next()){
    //agregue el código que crea necesario
}
```