

Lenguaje de programación I



Caso



- ✓ El administrador de la empresa CiberFarma necesita **registrar** los datos de los usuarios del sistema. Diseñe el formulario y los procesos de Gestión



Registro de Usuario

Código: Autogenerado

Nombre: (15 cara...

Apellido: (25 cara...

Usuario:

Clave:

Fecha:  (cuadro de fecha)

Contenido

- Uso de Bases de datos
- MySQL
- JDBC – definición
- El API JDBC – clases e interfaces
- Empezando a trabajar con JDBC
- Ejercicio: Crear un registro.
- Aplicaciones

Logros de la Unidad

- Crear aplicaciones de manejo de bases de datos



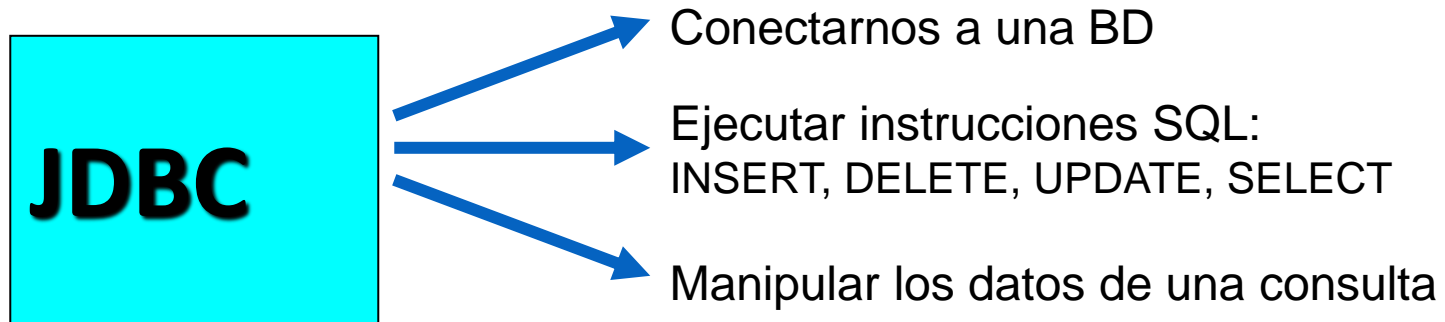
¿Por qué MySQL?

- ✓ Sistema gestor de base de datos multiplataforma
- ✓ Desarrollado en C
- ✓ Licencia código abierto GPL:
 - Software base: [sitio web](#)
 - Herramienta visual: [Workbench](#)
- ✓ Soporte de un subconjunto de SQL 99
- ✓ Dispone driver JDBC para Java
- ✓ Abre por defecto el puerto 3306 para aceptar posibles conexiones



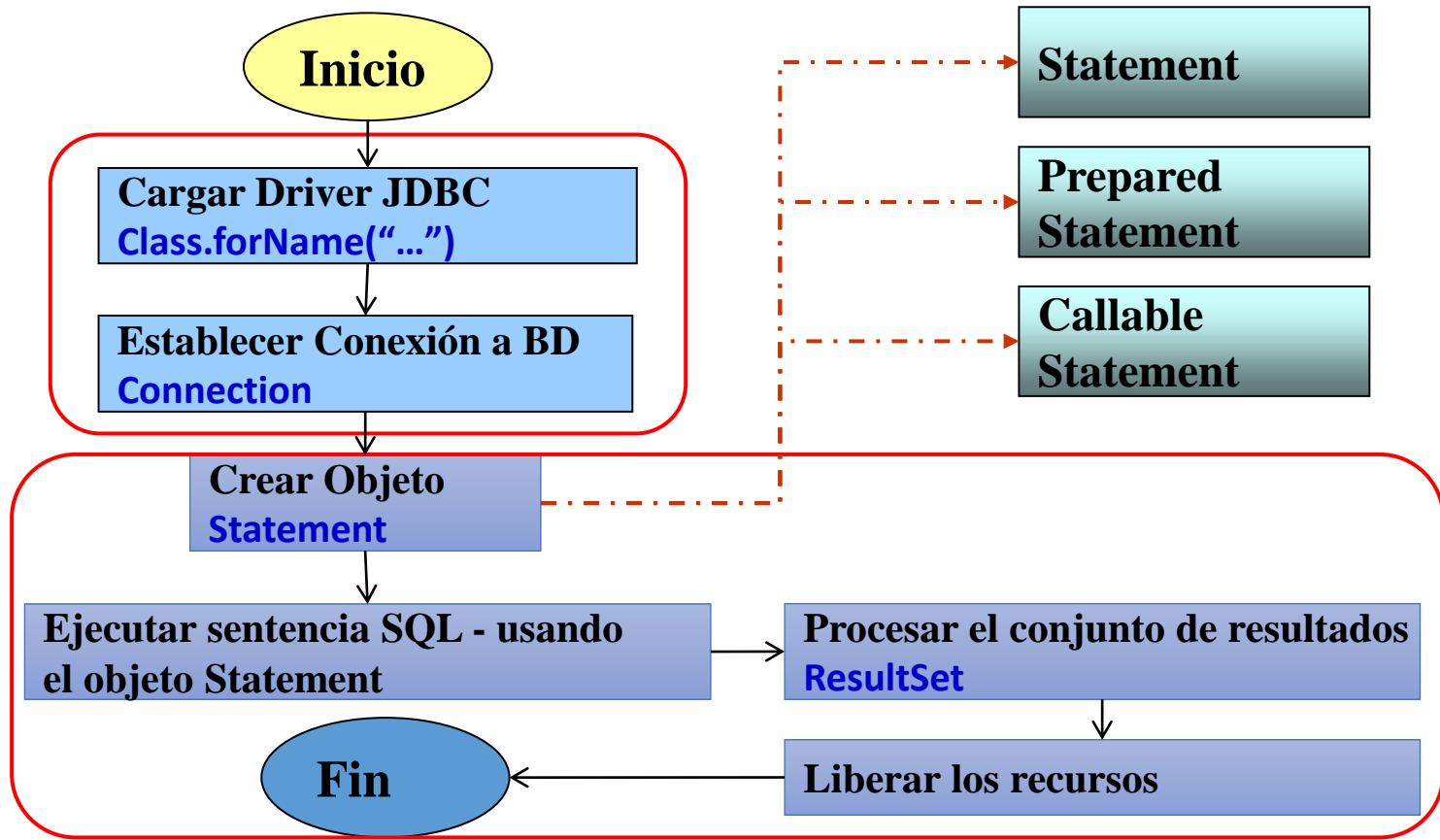
Conexión a BD usando Java

- **Java DataBase Connectivity (JDBC)** es la API (librería) estándar de acceso a base de datos desde Java
- Está incluida en Java SE (*Standard Edition*)
- Las clases necesarias **están agrupadas** en el paquete **java.sql**



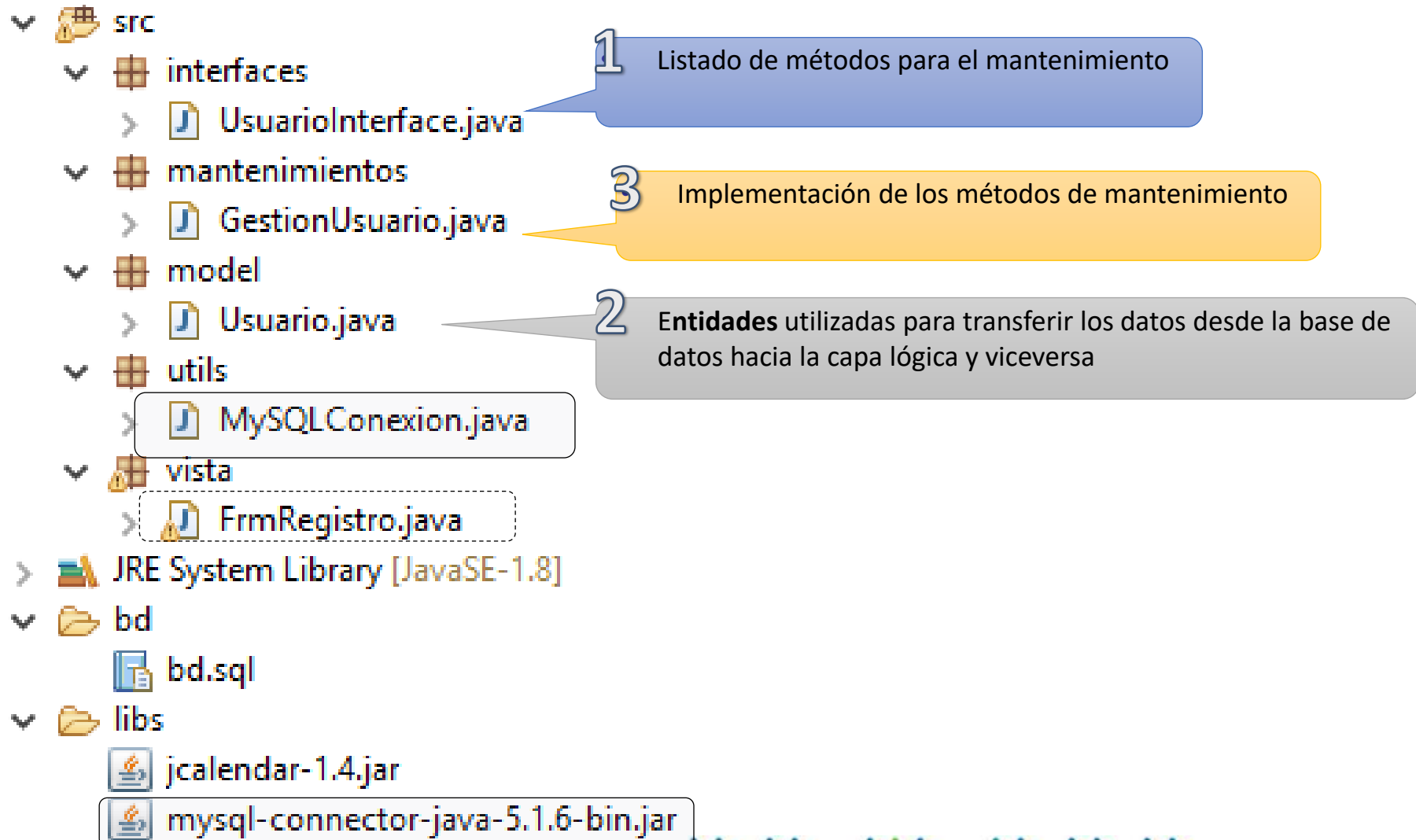
Conexión a BD usando Java

✓ Los pasos al desarrollar aplicaciones con base de datos son:



Estructura del Proyecto con BD

 LPI-semana05-plantilla-sol



GUI



- ✓ El administrador de la empresa CyberFarma necesita registrar los datos de los usuarios del sistema. Diseñe el formulario y los procesos de Gestión

Registro de Usuario

Código: Autogenerado

Nombre: (15 cara...)

Apellido: (25 cara...)

Usuario:

Clave:

Fecha: (cuadro de fecha)

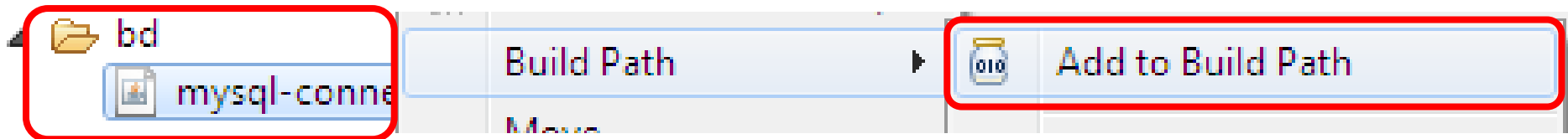
Registrar

Limpiar

```
void registraDatos() {  
    // variables  
    String nombre, apellido, usuario, clave, fecha;  
  
    // entradas  
    nombre = leerNombre();  
    apellido = leerApellido();  
    usuario = leerUsuario();  
    clave = leerClave();  
    fecha = leerFecha();  
  
    // procesos -> Crea un objeto de la clase Usuario  
    Usuario u = new Usuario();  
    // ingresa los valores al objeto  
    u.setNombre(nombre);  
    u.setApellido(apellido);  
    u.setUsuario(usuario);  
    u.setClave(clave);  
    u.setFacceso(fecha);  
    // Instanciamos la clase de Gestion  
    GestionUsuario gu = new GestionUsuario();  
    // llamamos al método de registro de usuarios  
    int ok = gu.registrar(u);  
  
    // salidas -> valida que el registro no debe ser 0  
    if (ok == 0) {  
        System.out.println("Error");  
    } else {  
        System.out.println("Registro Ok");  
    }  
}
```

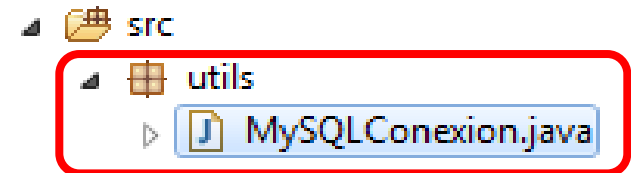

Driver de conexión

- Para conectarse a una base de datos concreta, es necesario un **archivo conector** o **driver JDBC**.
- El driver es un **fichero JAR** que se añade a la aplicación como cualquier otra librería pulsando **clic derecho...**



Plantilla de Conexión

- ✓ Esta clase **define** el driver de base de datos a usar, y retorna la conexión con la BD.



```
public static Connection getConexion(){
    Connection con = null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        String url = "jdbc:mysql://localhost/nombrebd";
        String usr = "root";
        String psw = "contraseña de mysql";
        con = DriverManager.getConnection(url,usr,psw);
    } catch (ClassNotFoundException ex) {
        System.out.println("Error >> Driver no Instalado!!");
    } catch (SQLException ex) {
        System.out.println("Error >> de conexión con la BD");
    }
    return con;
}
```

Clase Entidades

- Toma como **referencia las tablas o consultas** de donde obtendremos la información o guardaremos los datos.
- Ej: Para los datos del **usuario...**



```
CREATE TABLE tb_usuarios(  
codigo int auto_increment,  
nombre varchar(15),  
apellido varchar(25),  
usuario char(4) NOT NULL,  
clave char(5),  
facceso date null,  
tipo int DEFAULT 2,  
estado int(1) DEFAULT 1,  
primary key (codigo)  
);
```

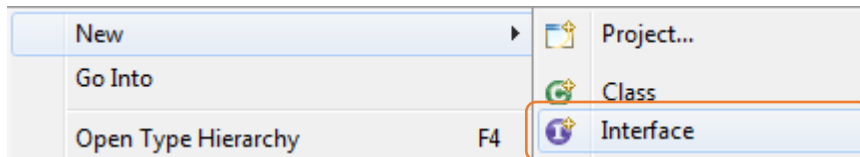
```
2  
3 public class Usuario {  
4     private int codigo, tipo, estado;  
5     private String nombre, apellido, usuario, clave, facceso;  
6  
7     public int getCodigo() {  
8         return codigo;  
9     }  
10    public void setCodigo(int codigo) {  
11        this.codigo = codigo;  
12    }  
13 }  
14  
15 // Crear los get y set
```

- También permite **obtener** datos adicionales como: totales, o concatenados:

```
// nombre completo  
public String getNombreCompleto() {  
    return nombre + " " + apellido;  
}
```

Interfaces

- ✓ La interface, **lista** los métodos de mantenimiento a implementar (registro, actualización, listados, etc.)
- ✓ Ej. Para el mantenimiento de Usuarios, creamos la interface...



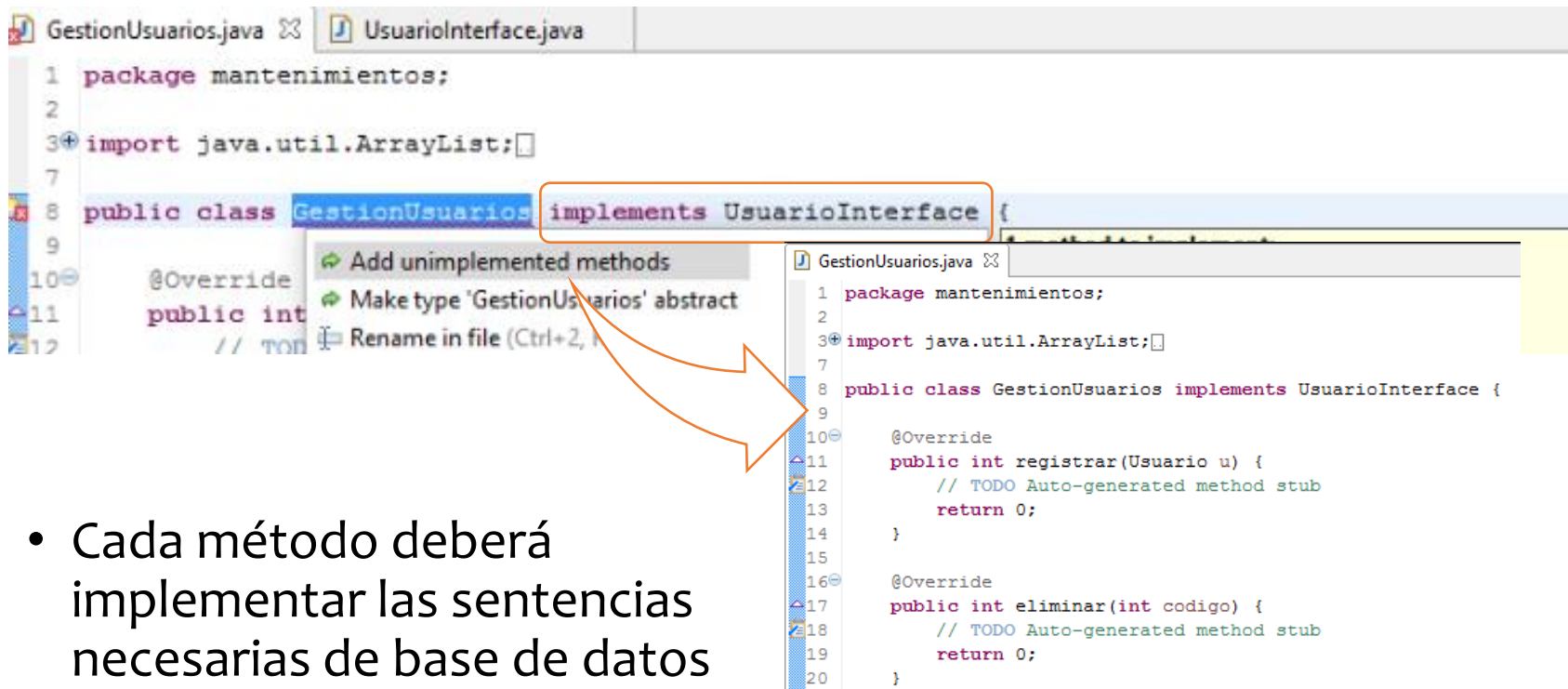
- Con los siguientes métodos

```
UsuarioInterface.java
1 package interfaces;
2
3 public interface UsuarioInterface {
4     // para registrar los datos de un usuario
5     public int registrar (Usuario u);
6
7     // para eliminar un usuario según su código
8     public int eliminar (int codigo);
9
10    // para el listado de los datos de todos los usuarios
11    public ArrayList<Usuario> listado();
12 }
```



3. Clases de Control o Gestión

- Estas clases **implementarán** los métodos para el mantenimiento.
- Ej. Para la **gestión de usuarios**, se implementa la **Interface**



```
1 package mantenimientos;
2
3 import java.util.ArrayList;
4
5
6
7
8 public class GestionUsuarios implements UsuarioInterface {
9
10
11 @Override
12 public int registrar(Usuario u) {
13     // TODO Auto-generated method stub
14     return 0;
15 }
16
17 @Override
18 public int eliminar(int codigo) {
19     // TODO Auto-generated method stub
20     return 0;
21 }
```

- Cada método deberá implementar las sentencias necesarias de base de datos

Aplicación



- ✓ El administrador de la empresa Ciberfarma necesita **registrar** los datos de los usuarios del sistema.

```
public int registrar(Usuario u) {  
    int rs = 0;  
    Connection con = null;  
    PreparedStatement pst = null;
```

rs, variable de control del resultado de la operación con la BD

```
    try {  
        con = MySQLConexion.getConnection();
```

```
        String sql = "insert into tb_usuarios values (null,?,?,?,?,?, default, default)";
```

```
        pst = con.prepareStatement(sql);  
        // parámetros  
        pst.setString(1, u.getNombre());  
        pst.setString(2, u.getApellido());  
        pst.setString(3, u.getUsuario());  
        pst.setString(4, u.getClave());  
        pst.setString(5, u.getFacceso());
```

Sentencia a ejecutar en el gestor de BD

```
        rs = pst.executeUpdate();
```

Cambia el valor de rs al realizar exitosamente la actualización o registro

```
    } catch (Exception e) {  
        System.out.println("Error en la sentencia " + e.getMessage());  
    } finally {  
        try {  
            if(pst!=null) pst.close();  
            if(con!=null) con.close();  
        } catch (SQLException e) {  
            System.out.println("Error al cerrar ");  
        }  
    }  
    return rs;  
}
```



Sentencia y parámetros

- Por ejemplo:
- Sentencia en MySQL:

```
select * from tb_usuarios where usuario = 'U001' and clave = '10001';
```

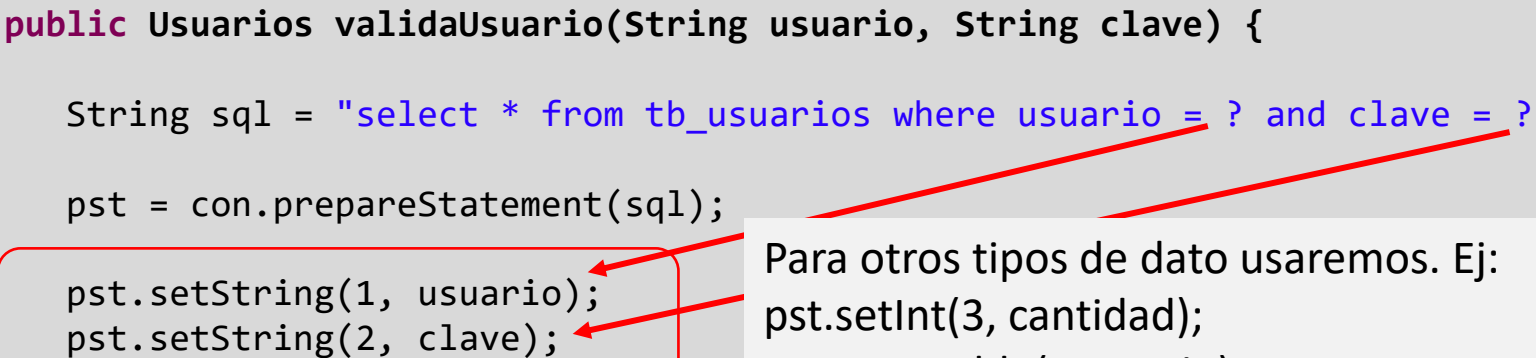


- Sentencia en clase mantenimiento:

```
sql = "select * from tb_usuarios where usuario = ? and clave = ?";
```

- Cada ? Representa un parámetro (empezando desde 1) el cual se reemplazará con los valores respectivos, Ej:

```
public Usuarios validaUsuario(String usuario, String clave) {  
  
    String sql = "select * from tb_usuarios where usuario = ? and clave = ?";  
  
    pst = con.prepareStatement(sql);  
  
    pst.setString(1, usuario);  
    pst.setString(2, clave);  
}
```



Para otros tipos de dato usaremos. Ej:

```
pst.setInt(3, cantidad);  
pst.setDouble(4, precio);
```

Tipo de resultado y ejecución

✓ El tipo de resultado y **ejecución** dependerá del tipo de sentencia.

Para	Sentencia SQL a usar	Valor devuelto	Tipo de Ejecución
Registrar, Insertar, Modificar, eliminar	INSERT, UPDATE, DELETE	int rs = 0; 0 es el valor por default en caso de error	executeUpdate ()
Consultas, Listados	SELECT	ResultSet rs = null; null, valor por default en caso no se obtengan datos	executeQuery ()



Referencia

- ✓ <http://java.sun.com/javase/technologies/database>
- ✓ <http://java.sun.com/docs/books/tutorial/jdbc/>

GRACIAS



SEDE MIRAFLORES

Calle Díez Canseco Cdra 2 / Pasaje Tello
Miraflores – Lima
Teléfono: 633-5555

SEDE INDEPENDENCIA

Av. Carlos Izaguirre 233
Independencia – Lima
Teléfono: 633-5555

SEDE BREÑA

Av. Brasil 714 – 792
(CC La Rambla – Piso 3)
Breña – Lima
Teléfono: 633-5555

SEDE TRUJILLO

Calle Borgoño 361
Trujillo
Teléfono: (044) 60-2000

SEDE SAN JUAN DE LURIGANCHO

Av. Próceres de la Independencia 3023-3043
San Juan de Lurigancho – Lima
Teléfono: 633-5555

SEDE SAN MIGUEL

Av. Federico Gallese 847
San Miguel – Lima
Teléfono: 632-4900

SEDE BELLAVISTA

Av. Mariscal Oscar R. Benvides 3866 – 4070
(CC Mall Aventura Plaza)
Bellavista – Callao
Teléfono: 633-5555

SEDE AREQUIPA

Av. Porongoche 500
(CC Mall Aventura Plaza)
Paucarpata - Arequipa
Teléfono: (054) 60-3535