

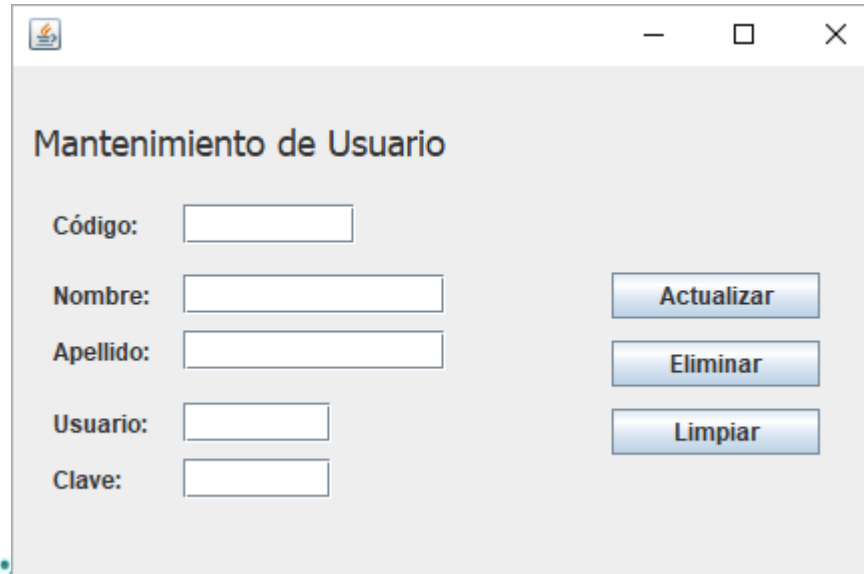
Lenguaje de programación I



Caso



- ✓ El administrador de la empresa CiberFarma necesita realizar los siguientes procesos:
 - Eliminar un usuario según su código. La GUI debe mostrar un mensaje en caso de éxito o de error según corresponda
 - Actualizar los datos de un usuario según su código y los datos del formulario.
- ✓ Diseña las GUI respectivas y los procesos de interface y gestión.



Mantenimiento de Usuario

Código:

Nombre:

Apellido:

Usuario:

Clave:

Actualizar

Eliminar

Limpiar

Contenido

- Uso de Bases de datos
- Ejercicio: mantenimiento de BD.

Logros de la Unidad

- Crear aplicaciones de manejo de bases de datos



Estructura del Proyecto con BD

▼ LPI-semana06-plantilla

▼ src

▼ interfaces

> UsuarioInterface.java

1

Listado de métodos para el mantenimiento

▼ mantenimientos

> GestionUsuario.java

3

Implementación de los métodos de mantenimiento

▼ model

> Usuario.java

2

Entidades utilizadas para transferir los datos desde la base de datos hacia la capa lógica y viceversa

▼ utils

> MySQLConexion.java

▼ vista

> FrmMantenimiento.java

> JRE System Library [JavaSE-1.8]

> jcalendar-1.4.jar

▼ bd

bd.sql

▼ libs

jcalendar-1.4.jar

mysql-connector-java-5.1.6-bin.jar

GUI

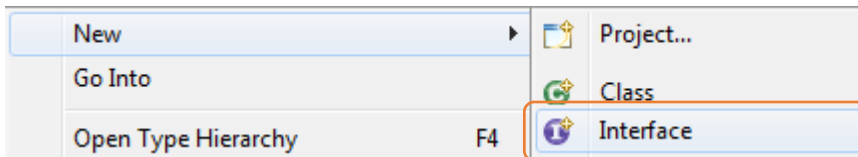


- ✓ El administrador de la empresa CiberFarma necesita actualizar los datos de los usuarios del sistema o eliminar (cambiar el estado)
- ✓ Diseñe el formulario y los procesos de Gestión

A screenshot of a Windows-style window titled "Mantenimiento de Usuario". The window has a standard title bar with minimize, maximize, and close buttons. Inside the window, there is a form with five input fields labeled "Código:", "Nombre:", "Apellido:", "Usuario:", and "Clave:". To the right of the "Nombre:" and "Apellido:" fields, there is a button labeled "Actualizar". Below the "Apellido:" field, there is a button labeled "Eliminar". Below the "Usuario:" field, there is a button labeled "Limpiar". The "Clave:" field is at the bottom left of the form.

1. 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
5     // para eliminar un usuario según su código
6     public int eliminar (int codigo);
7
8     // para actualizar los datos de un usuario
9     public int actualizar (Usuario u);
10
11
12 }
```



2. 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)  
);
```

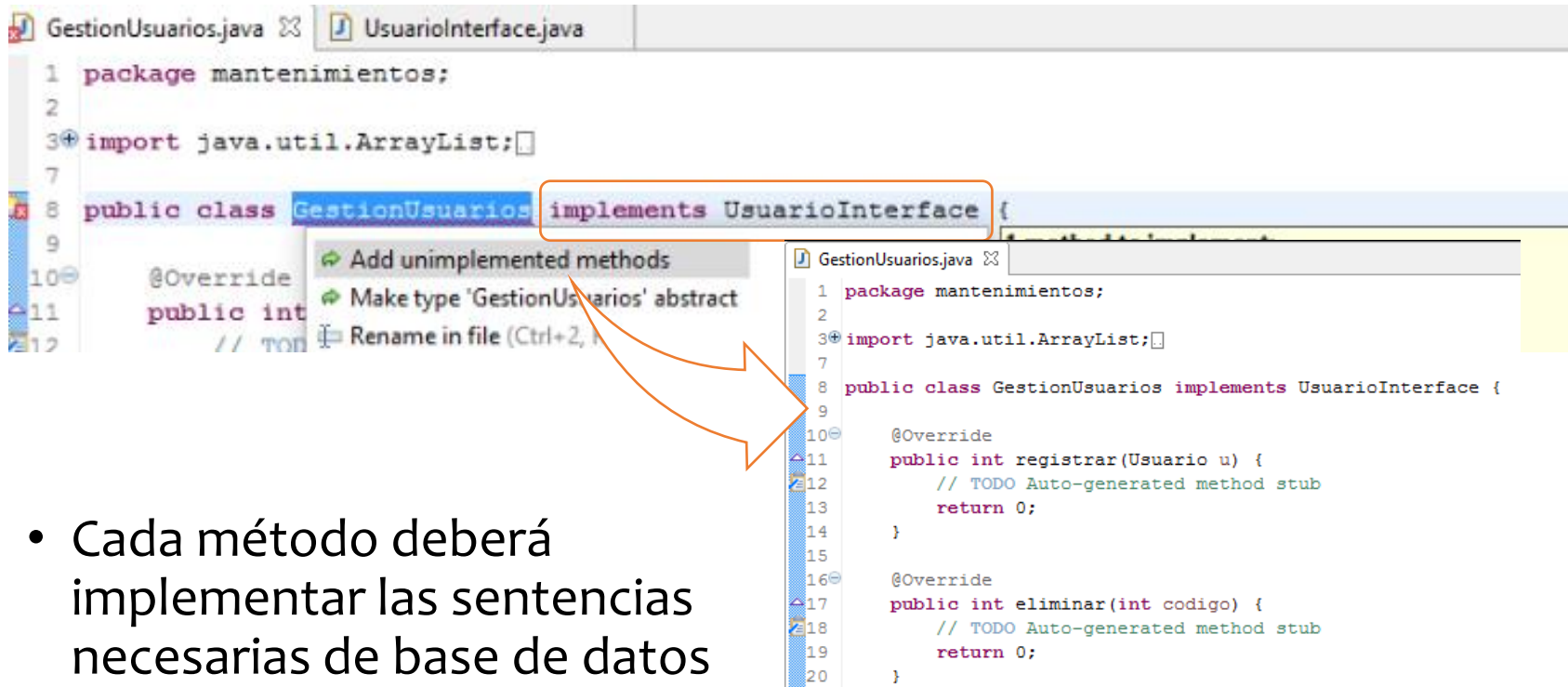
```
2  
3 public class Usuario {  
4     private int codigo, tipo, estado;  
5     private String nombre, apellido, usuario, clave, facceso;  
6  
7     public int getCodigo() {  
8         return codigo;  
9     }  
10    public void setCodigo(int codigo) {  
11        this.codigo = codigo;  
12   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

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     }
22 }
```

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

3. Clases de Control o Gestión. Aplicación



✓ Ej. Método actualizar donde se actualiza el nombre según el código del usuario

```
public int actualizar(Usuario u) {  
    int rs = 0;  
    Connection con = null;  
    PreparedStatement pst = null;  
    try {  
        con = MySQLConexion.getConnection();  
        String sql = "update tb_usuarios set nombre = ? where codigo = ?";  
        // sql -> cadena con la sentencia a utilizar  
        pst = con.prepareStatement(sql);  
        // parámetros  
        pst.setString(1, u.getNombre());  
        pst.setInt(2, u.getCodigo());  
        rs = pst.executeUpdate();  
    } catch (Exception e) {  
        System.out.println("Error en la sentencia ..." + e);  
    } finally {  
        try {  
            if (pst != null)  
                pst.close();  
            if (con != null)  
                con.close();  
        } catch (SQLException e) {  
            System.out.println("Error al cerrar ");  
        }  
    }  
    return rs;  
}
```

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

String sql = "update tb_usuarios set nombre = ? where codigo = ?";
// sql -> cadena con la sentencia a utilizar

Sentencia a ejecutar en el gestor de BD

// parámetros
pst.setString(1, u.getNombre());
pst.setInt(2, u.getCodigo());

rs = pst.executeUpdate();

return rs;



Sentencia y parámetros

- Por ejemplo:
- Sentencia en MySQL:

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

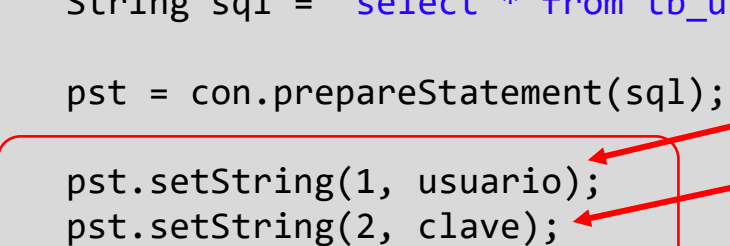


- Sentencia en clase mantenimiento:

```
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);
```

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