¿QUÉ ES EL PATRÓN BRIDGE?

El patrón Bridge es un **patrón estructural** de diseño que tiene como objetivo separar la abstracción de su implementación, permitiendo que ambas evolucionen independientemente. Es especialmente útil cuando tienes una clase con varias implementaciones posibles y deseas evitar acoplamientos fuertes entre la interfaz y las implementaciones concretas.

¿Cuándo usar el Patrón Bridge?

Es adecuado usarlo cuando:

- Deseas evitar vínculos permanentes entre una abstracción y sus implementaciones concretas.
- Necesitas poder cambiar fácilmente la implementación sin afectar la lógica de negocio que la usa.
- Quieres que las abstracciones y sus implementaciones evolucionen de forma independiente.

Estructura general del Patrón Bridge:

El patrón Bridge está formado por cuatro elementos fundamentales:

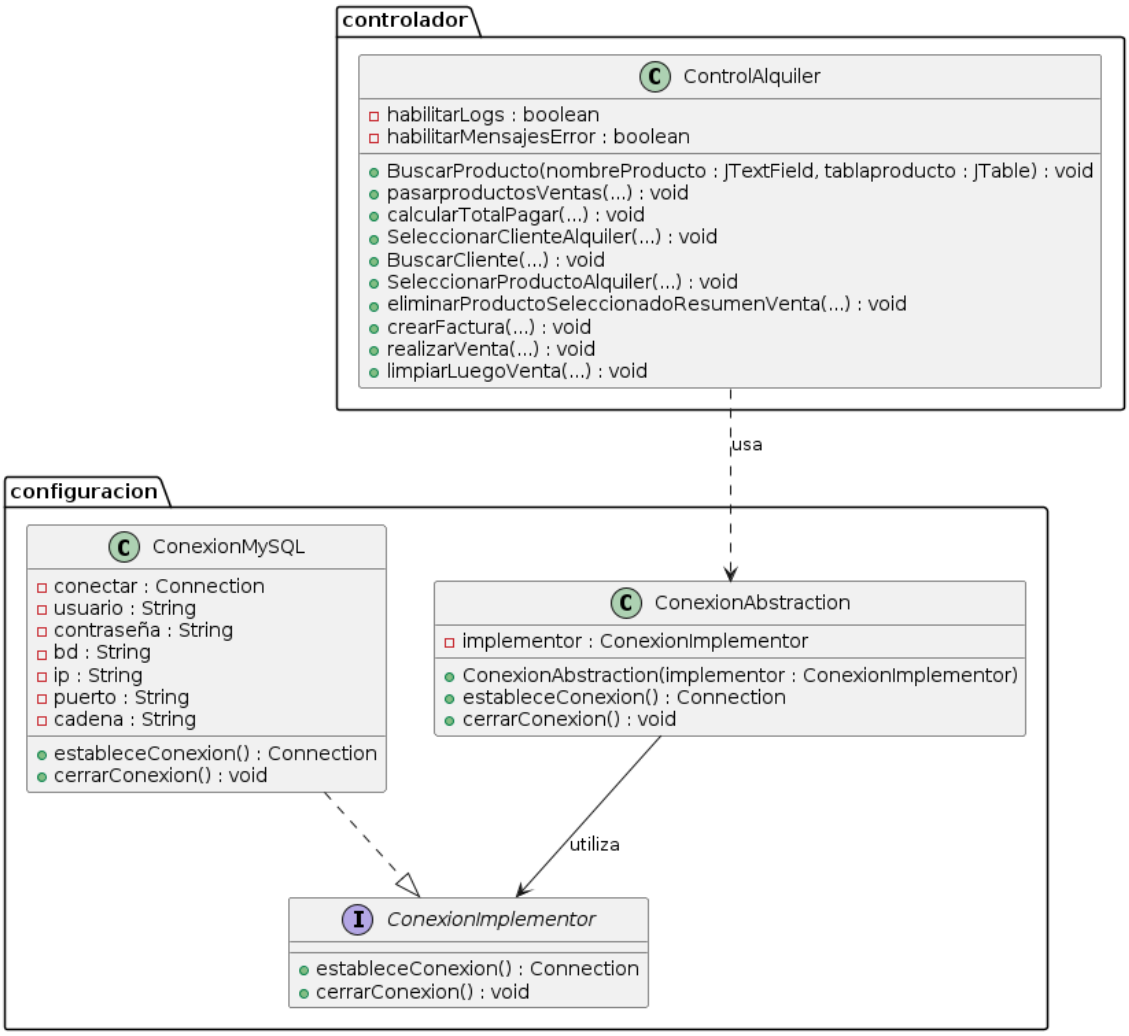
1. Abstraction (Abstracción)
Es una clase abstracta o una interfaz que define la abstracción general y contiene una referencia hacia el implementador.

2. RefinedAbstraction (Abstracción refinada)
Es una clase concreta derivada de la abstracción, que extiende la funcionalidad definida por la abstracción base.

3. Implementor (Implementador)
Es una interfaz que declara los métodos que deberán ser implementados por las implementaciones concretas.

4. ConcreteImplementor (Implementador concreto)
Son las clases concretas que implementan la interfaz del implementador, proporcionando la lógica específica.

UML PATRON BRIDGE



Implementación del patrón Bridge

1. Interfaz Implementor (ConexionImplementor)

Primero, se creó una interfaz llamada **ConexionImplementor**, que define dos métodos esenciales para cualquier tipo de conexión: estableceConexion() y cerrarConexion(). Esta interfaz permite tener varias implementaciones distintas sin alterar la lógica general.

```java
package configuracion;

import java.sql.Connection;

public interface ConexionImplementor {
    Connection estableceConexion();
    void cerrarConexion();
}
```

2. Implementación Concreta (ConexionMySQL)

Posteriormente, la clase original **Conexion.java** fue adaptada a una clase llamada **ConexionMySQL**, que implementa la interfaz antes mencionada. Esta clase contiene toda la lógica específica para conectarse a una base de datos MySQL, incluyendo detalles como usuario, contraseña, base de datos, dirección IP y puerto. Su función es encargarse exclusivamente de la conexión real a MySQL.

```
public class Conexion implements ConexionImplementor {
    Connection conectar = null;

    String usuario = "root";
    String contraseña = "patata12";
    String bd = "alquileres";
    String ip = "localhost";
    String puerto = "3306";

    String cadena = "jdbc:mysql://" + ip + ":" + puerto + "/" + bd;

    @Override
    public Connection estableceConexion() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conectar = DriverManager.getConnection(cadena, usuario, contrasei
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "No se conectó correctamente
        }
        return conectar;
    }
```

3. Abstracción (ConexionAbstraction)

Se creó una nueva clase denominada **ConexionAbstraction**, que actúa como puente (bridge). Esta clase no conoce detalles específicos sobre la base de datos, sino que mantiene una referencia hacia la interfaz ConexionImplementor. Los métodos estableceConexion() y cerrarConexion() de esta abstracción simplemente delegan el trabajo a la implementación concreta (en este caso, ConexionMySQL).

```
import java.sql.Connection;

public class ConexionAbstraction {
    protected ConexionImplementor implementor;

    public ConexionAbstraction(ConexionImplementor implementor) {
        this.implementor = implementor;
    }

    public Connection estableceConexion() {
        return implementor.estableceConexion();
    }

    public void cerrarConexion() {
        implementor.cerrarConexion();
    }
}
```

4. Uso del Bridge en ControlAlquiler

Finalmente, en la clase **ControlAlquiler**, en lugar de crear directamente una instancia de la clase concreta de conexión, ahora se utiliza la abstracción ConexionAbstraction. Cuando se crea una instancia de la abstracción, se le pasa la implementación concreta que se quiere utilizar (en este caso, ConexionMySQL). Así, el controlador no depende directamente de la clase concreta, sino que interactúa únicamente con la abstracción.

```
,

// ♡ Método que ahora usa el Bridge para la conexión a la base de datos
public void BuscarProducto(JTextField nombreProducto, JTable tablaproducto) {
    // Se utiliza la abstracción de conexión, pasando la implementación concreta (ConexionMySQL)
    ConexionAbstraction objetoConexion = new ConexionAbstraction(new Conexion());

    DefaultTableModel modelo = new DefaultTableModel();
    modelo.addColumn("id");
    modelo.addColumn("Nombre");
    modelo.addColumn("PrecioProducto");
    modelo.addColumn("stock");
```