# Bridge

**Definición:**

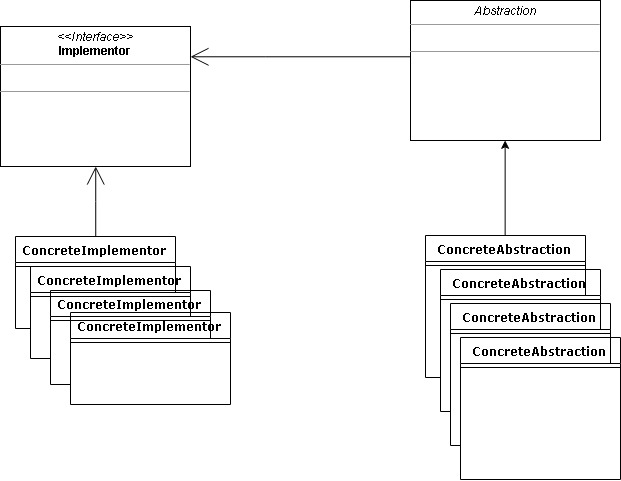
“Desacoplar una abstracción de su implementación de modo que los dos puedan ser modificados de forma independiente.”  
Design Patterns: Elements of Reusable Object-Oriented Software

El patrón bridge nos permite desde una clase abstracta / interfase poder variar sus funcionalidades o sus propiedades sin que afecten las funcionalidades a las propiedades o las propiedades a las funcionalidades.

Ejemplo:

Se necesitan implementar varios tipos de controles para controlar varios tipos de tvs, dvds, radios, etc. Aplicando bridge Creariamos una clase abstracta Control que se implementaria en control1 control2 control3 y tendria un atributo de tipo Dispositivo que tambien seria una clase abstracta la cual se implementaria en tv, dvd, radio y estos implementarian sus funciones específicas. Estos nos permite hacer cualquier permutación entre las implementaciones de Control y las implementaciones de Dispositivo si tener que crear clases nuevas para cada una de esta Ej: control1tv.

Diagrama de Clase



**Motivacion**

El patrón de bridge se aplica de manera preventivamente para el control de problemas que se podrían generar en el futuro y así evitarlos de antemano.

A diferencia del patrón de Adapter que se utiliza para resolver un problema generado para poder variar dos interfaces independientemente.