Bridge

Propósito

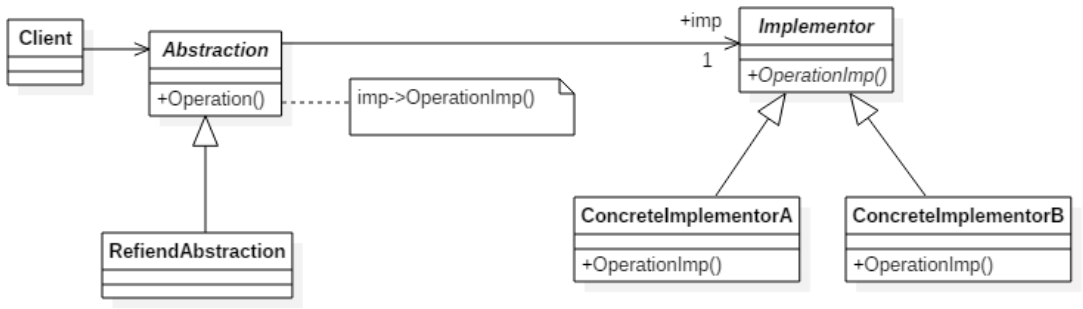
Desacopla una abstracción de su implementación de modo que las dos puedan variar de forma independiente.

Aplicabilidad

El patrón Bridge se utiliza cuando:

- Se desea evitar un enlace permanente entre abstracción e implementación. Esto podría ser útil cuando se desea cambiar la implementación en tiempo de ejecución
- Tanto las abstracciones como sus implementaciones deben extenderse mediante subclasificación. En este caso, el patrón Bridge te permite combinar diferentes abstracciones e implementaciones y extenderlas independientemente
- Los cambios en la implementación de una abstracción no deberían tener impacto en los clientes, esto es, no debería ser recompilado
- Se tiene una proliferación de clases, que indicaría que el objeto se puede dividir en dos partes
- Quieres compartir una implementación entre múltiples objetos

Estructura



Participantes

- Abstraction: define la interface de la abstracción. Tiene una referencia a un objeto de tipo Implementor
- RefinedAbstraction: extiende la interface definida por Abstraction
- Implementor: define la interface las clases de implementación. Este interface no tiene porqué ajustarse a la interface de Abstraction
- ConcreteImplementor: implementa la interface de Implementor y define su implementación concreta

Colaboraciones

- La Abstraction reenvía las peticiones del cliente a su objeto Implementor

Consecuencias

El patrón Bridge tiene las siguientes consecuencias:

- Desacopla interface e implementación: Una implementación no está permanentemente atada a una interface. La implementación de una abstracción se puede configurar en tiempo de ejecución. Es incluso posible que un objeto cambie su implementación en tiempo de ejecución
  Desacoplar Abstraction e Implementor también elimina las dependencias en tiempo de compilación de la implementación. Cambiar una clase de implementación no requiere recompilar una clase Abstraction y sus clientes. Esta propiedad es esencial cuando se desea asegurar compatibilidad binaria entre diferentes versiones de una biblioteca de clases.
  Además, este desacoplamiento favorece la aparición de capas, lo cual lleva a un sistema mejor estructurado. La parte de más alto nivel del sistema le basta con conocer a Abstraction e Implementor
- Mejora la extensibilidad. Puedes extender de forma independiente las jerarquías de Abstraction e Implementor.
- Oculta los detalles de implementación a los clientes. Puedes proteger a los clientes de los detalles de implementación.