Use the bridge pattern to vary not only your implementations, but also your abstractions.

The ***bridge pattern*** allows you to vary the implementation and the abstraction by placing the two separate class hierarchies.

***Benefits:***

* Decouples an implementation so that it is not bound permanently to an interface.
* Abstraction and implementation can be extended independently.
* Changes to the concrete abstraction classes don’t affect the client.

***Uses and Drawbacks:***

* Useful in graphics and windowing systems that need to run over multiple platforms.
* Useful any time you need to vary an interface and an implementation in different ways.
* Increases complexity.

Implementor.tuneChannel(channel);

TV

on()

off()

tuneChannel()

ConcreteRemote

currentStation

on()

off()

setChannel()

nextChannel()

previousChannel()

RemoteControl

implementor

on()

off()

setChannel()

Has-A

Sony

on()

off()

tuneChannel()

RCA

on()

off()

tuneChannel()

setChannel(currentSituation + 1)