**Bridge Design Pattern**

When we have interface hierarchies in both interfaces as well as implementations, then **bridge design pattern** is used to decouple the interfaces from implementation and hiding the implementation details from the client programs.

ust like [Adapter pattern](https://www.journaldev.com/1487/adapter-design-pattern-java), bridge design pattern is one of the **Structural design pattern**.

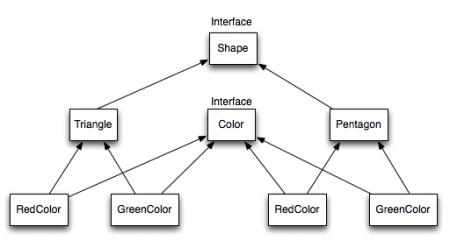
According to GoF bridge design pattern is:

Decouple an abstraction from its implementation so that the two can vary independently

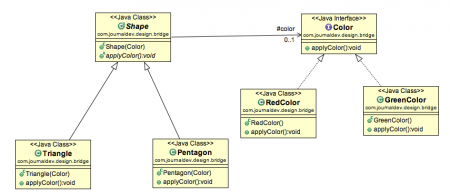
The implementation of bridge design pattern follows the notion to prefer [Composition](https://www.journaldev.com/1325/composition-in-java-example) over [inheritance](https://www.journaldev.com/644/inheritance-java-example).

### Bridge Design Pattern in Java Example

If we look into bridge design pattern with example, it will be easy to understand. Lets say we have an interface hierarchy in both interfaces and implementations like below image.



Now we will use bridge design pattern to decouple the interfaces from implementation. UML diagram for the classes and interfaces after applying bridge pattern will look like below image.

[](https://cdn.journaldev.com/wp-content/uploads/2013/07/bridge-design-pattern.png)

Notice the bridge between Shape and Color interfaces and use of composition in implementing the bridge pattern.

Bridge design pattern can be used when both abstraction and implementation can have different hierarchies independently and we want to hide the implementation from the client application.