Bridge Pattern:

**When we need bridge design pattern**

The Bridge pattern is an application of the old advice, “prefer composition over inheritance”. It becomes handy when you must subclass different times in ways that are orthogonal with one another.

The Bridge design pattern allows you to separate the abstraction from the implementation.It is a structural design pattern.

1. Abstraction
2. Implementation

This is a design mechanism that encapsulates an implementation class inside of an interface class.

* The bridge pattern allows the Abstraction and the Implementation to be developed independently and the client code can access only the Abstraction part without being concerned about the Implementation part.
* The abstraction is an interface or abstract class and the implementor is also an interface or abstract class.
* The abstraction contains a reference to the implementor. Children of the abstraction are referred to as refined abstractions, and children of the implementor are concrete implementors. Since we can change the reference to the implementor in the abstraction, we are able to change the abstraction’s implementor at run-time. Changes to the implementor do not affect client code.
* It increases the loose coupling between class abstraction and it’s implementation.

**Advantages**

1. Bridge pattern decouple an abstraction from its implementation so that the two can vary independently.
2. It is used mainly for implementing platform independence feature.
3. It adds one more method level redirection to achieve the objective.
4. Publish abstraction interface in a separate inheritance hierarchy, and put the implementation in its own inheritance hierarchy.
5. Use bridge pattern to run-time binding of the implementation.
6. Use bridge pattern to map orthogonal class hierarchies
7. Bridge is designed up-front to let the abstraction and the implementation vary independently.

<https://www.geeksforgeeks.org/bridge-design-pattern/>