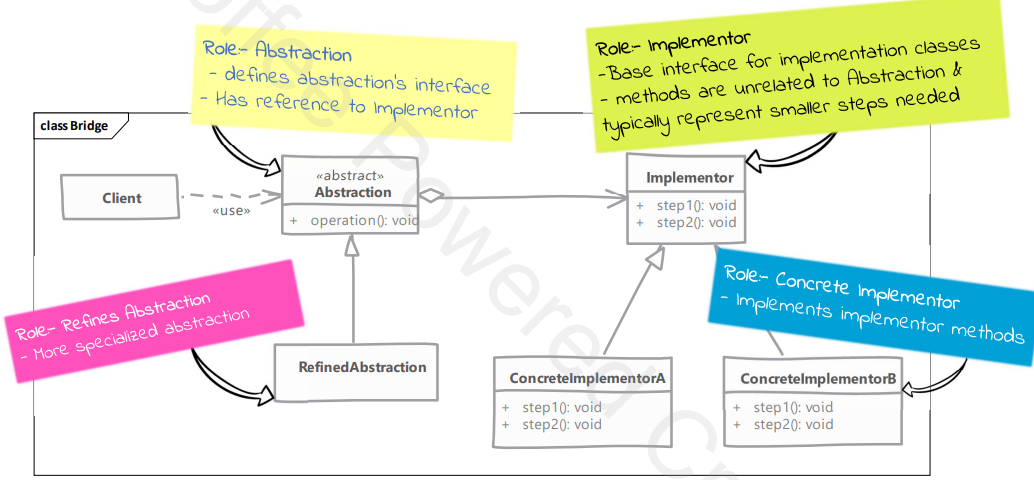
Bridge Pattern

Structural Design Pattern

**What is Bridge:**

1. Lets you split a large class or a set of closely related classes into two separate hierarchies - abstraction and implementation - which can be developed independently of each other.
2. Using the bridge pattern we can decouple implementations & abstractions so they can both change without affecting each other.
3. Achieved by creating two separate inheritance hierarchies; one for implementation and another for abstraction.
4. We use composition to bridge these two hierarchies.

**UML:**



**Implementation:**

1. Create Bridge Implementer interface. (Implementer)
2. Create concrete bridge implementer classes implementing the Implementor.
3. Create an abstract class (Abstraction) using the Implementor interface.
4. Create a concrete class implementing the Abstraction interface.
5. Use the Abstraction and Implementer to do actions. Call methods from implementers.

**Design Considerations:**

1. Bridge provides great extensibility by allowing us to change abstraction and implementor independently.
2. By using abstract factory pattern to create abstraction objects with correct implementation you can decouple concrete implementers from abstraction.

**Comparison with Adapter:**

1. Bridge Pattern’s intent is to allow abstraction and implementation to vary independently.
2. Bridge Pattern has to be designed up front then only we can have varying abstractions & implementations.
3. Adapter is meant to make unrelated classes work together.
4. Adapter Pattern finds its usage typically where a legacy system it to be integrated with new code.

**Example:**

