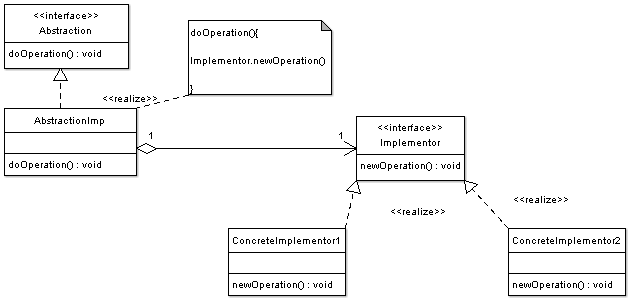
# Bridge Pattern

**Definition**: The Bridge Pattern is used to separate out the interface (decouple abstraction) from its implementation so that the two can vary independently.

**Problem**: Sometimes an abstraction should have different implementations, inheritance binds an implementation to the abstraction and thus it would be difficult to modify, extend, and reuse abstraction and implementation independently.

**Solution**:



* **Abstraction** — Abstraction defines abstraction interface.
* **AbstractionImp** — Implements the abstraction interface using a reference to an object of type Implementor.
* **Implementor** — Implementor defines the interface for implementation classes. This interface does not need to correspond directly to the abstraction interface and can be very different. AbstractionImp provides an implementation in terms of operations provided by Implementor interface.
* **ConcreteImplementor1, ConcreteImplementor2** — Implements the Implementor interface.
* **Shape (Abstraction)**: CircleShape, DrawingAPI \*m\_drawingAPI;
* **DrawingAPI (Implementor)**: DrawingAPI1, DrawingAPI2