Bridge Pattern

CSC 7400-51 Object Oriented Analysis And Design

Instructor: Prof. Nguyen Thai

Student: Lina Mi

Agenda

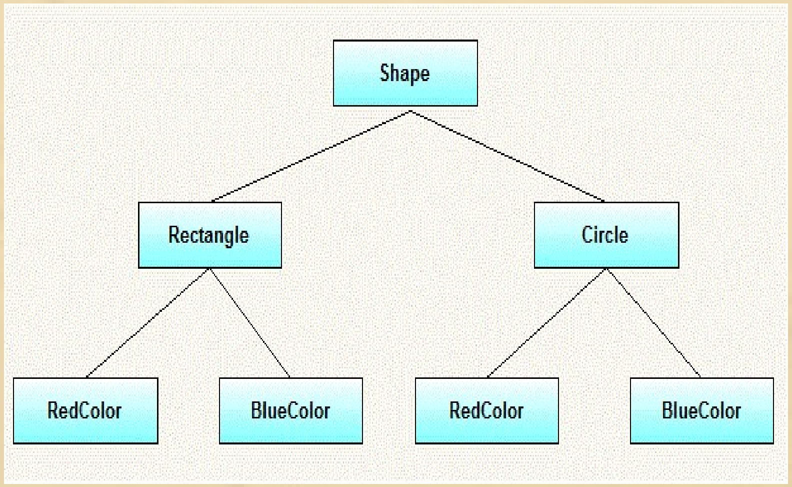
- What is Bridge Pattern
- Why Bridge Pattern and How it works
- Components of Bridge Pattern
- Implementation
- Summary
- References

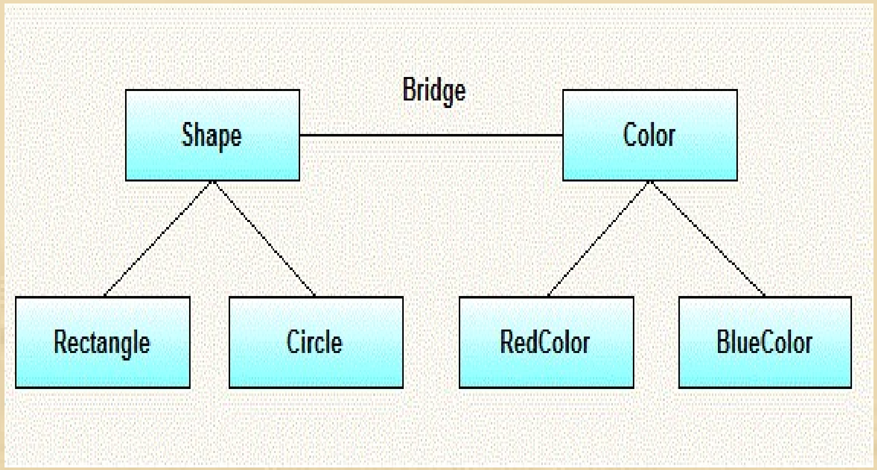
What is Bridge Pattern

- **Design Pattern: Best solutions to general problems, classified into :**
    - Creational Patterns
    - Structural Patterns
    - Behavioral Patterns
    - J2EE Patterns
- **What is Bridge Pattern:**
  a design pattern used in software engineering, comes under **structural patterns** as it **decouples** an abstraction from its implementation class so that the two can vary independently by providing a bridge structure between them.

Why Bridge Pattern and How It works

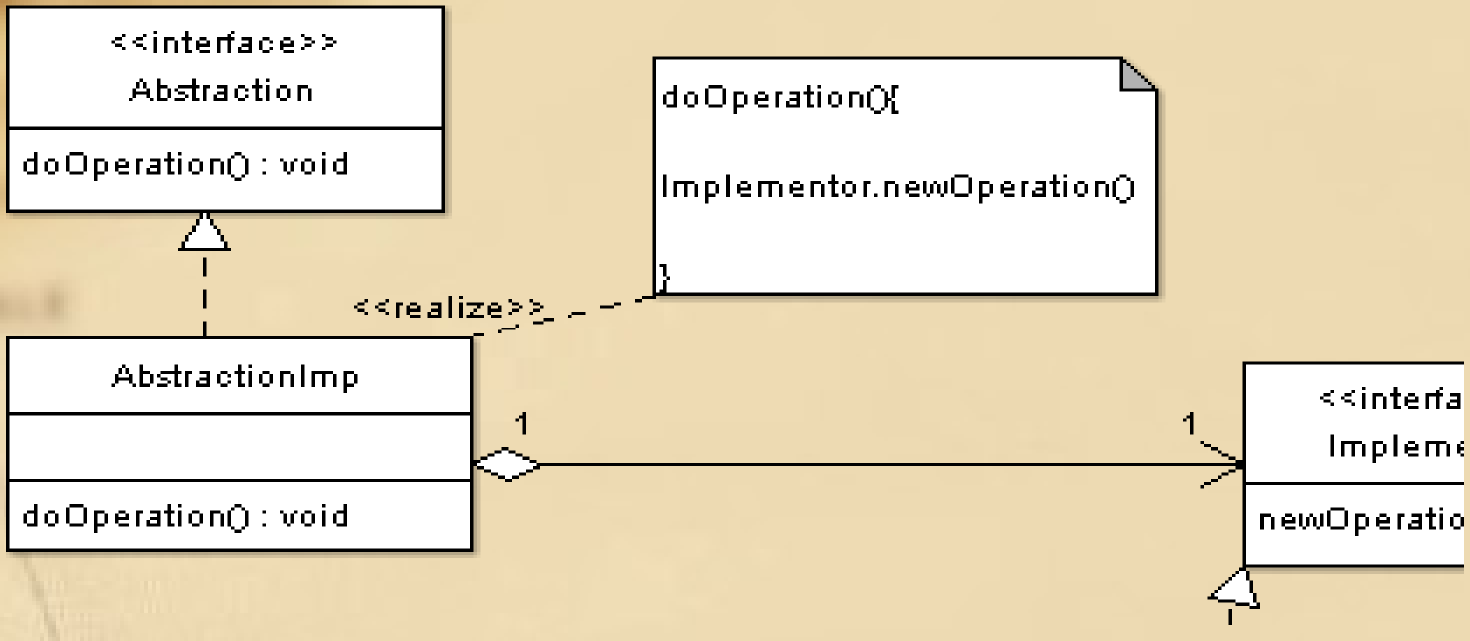
- **Why: Decouple an abstraction** from its **implementation** so that the two can vary independently, it is very useful when both Class (abstraction)and what it does (implementation) **vary** often
- **How:** add an interface which acts as a bridge which uses encapsulation, aggregation and inheritance to separate an abstraction from its implementation and put them into two different class hierarchies so that both can be altered structurally without affecting each other.





Components of Bridge Pattern

- **UML class diagram for the Bridge pattern**



- **Elements of Bridge pattern:**
  - **Abstraction**
  - **AbstractionImpl**
  - **Implementor**
  - **ConcretImplementor1, ConcretImplementtor2**

Implementation

- **Shape**: Abstraction
- **Rectangle**: RefinedAbstraction
- **Circle**: RefinedAbstraction
- **Color**: Implementor
- **Redcolor**: ConcretImplementor
- **Bluecolor:** ConcretImplementor

Summary

- Creates two different hierarchies. One for abstraction and the other for implementation.
- Avoids permanent binding by removing the dependency between abstraction and implementation.
- It create a bridge that coordinates between abstraction and implementation.
- Abstraction and implementation can be extended separately.
- Should be used when it needs to switch implementation at runtime.
- Client should not be impacted if there is modification in implementation of abstraction.
- Best used when there are multiple implementations.

References

- https://en.wikipedia.org/wiki/Bridge_pattern
- https://sourcemaking.com/design_patterns/bridge
- http://en.proft.me/2017/01/7/bridge-design-pattern-java-and-python/
- design_pattern_tutorial.pdf
- Design patterns.pdf
- DesignPatternInPython.pdf

Questions ?