SDET Course

Design Patterns - Bridge

3 Types of Design Patterns

- Creational
  - Singleton
  - Builder
  - Prototype
  - Factory Method
  - Abstract Factory
- Structural
  - Adapter
  - Composite
  - Proxy
  - Flyweight
  - **Bridge**
  - Facade
  - Decorator
- Behavioral
  - Strategy
  - Observer
  - Command
  - Memento
  - State
  - Template Method
  - Mediator
  - Chain of Responsibility
  - Interpreter
  - Visitor
  - Iterator

Agenda

- Description
- Diagram
- Code sample (Java)
- Use cases

Descri

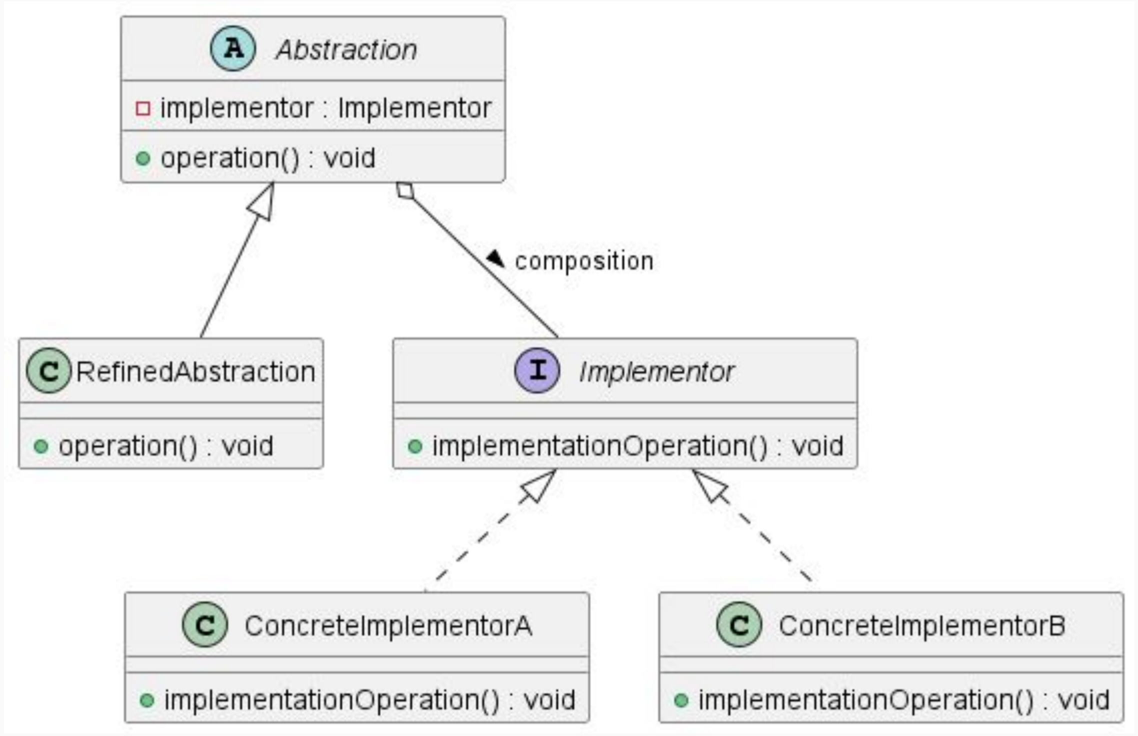
on

Introduction - A Universal Remote Control

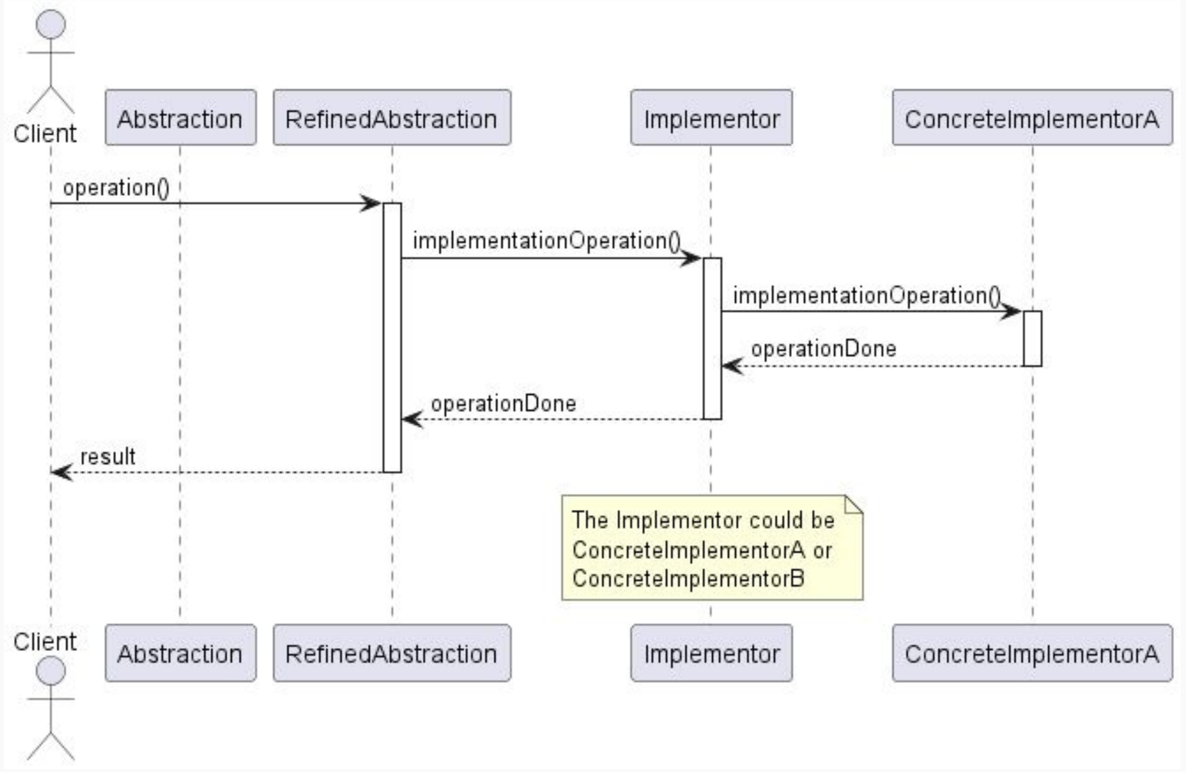
Description

The Bridge design pattern is a structural pattern used in software engineering to decouple an abstraction from its implementation, allowing the two to vary independently. This pattern involves an interface which acts as a bridge between the abstraction and its implementation. By doing so, it enables the modification and extension of both the abstraction and its implementation without affecting each other. The key benefit of the Bridge pattern is increased flexibility in code structure and functionality. It's especially useful in scenarios where an application needs to support multiple platforms, or when it's expected to evolve both in terms of its core functionalities and the ways they are implemented. The pattern also helps in adhering to the principle of open/closed principle, one of the SOLID principles, making software easier to manage and extend over time.

Class Diagram



Sequence Diagram



Code Sample

Use cases

- General
  - Cross Platform Applications
  - Database Persistence Layers (e.g Fillo)

- In Test Automation
  - Appium
  - Test Reporting
  - Different Environments

VERISOFT

Happy Coding