SDET Course

Design Patterns - State

3 Types of Design Patterns

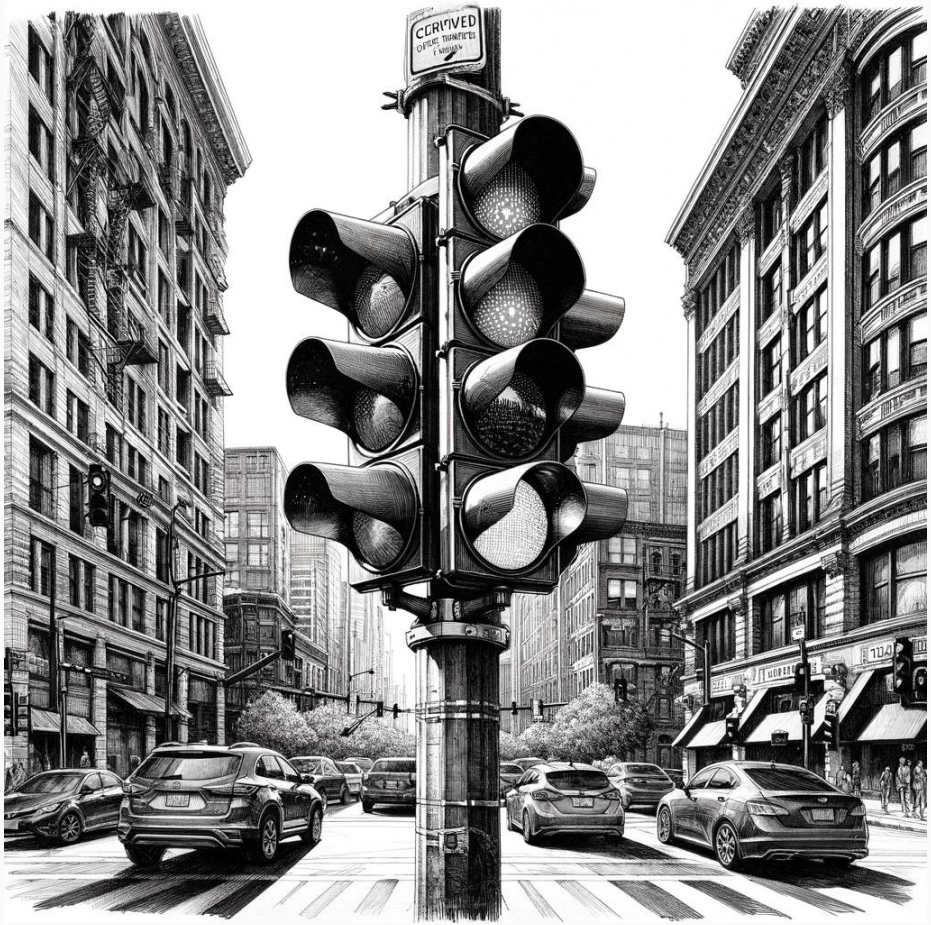
- Creational
  - Singleton
  - Builder
  - Prototype
  - Factory Method
  - Abstract Factory
- Structural
  - Adapter
  - Composite
  - Proxy
  - Flyweight
  - Bridge
  - Facade
  - Decorator
- Behavioral
  - Strategy
  - Observer
  - Command
  - Memento
  - **State**
  - Template Method
  - Mediator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

Agenda

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

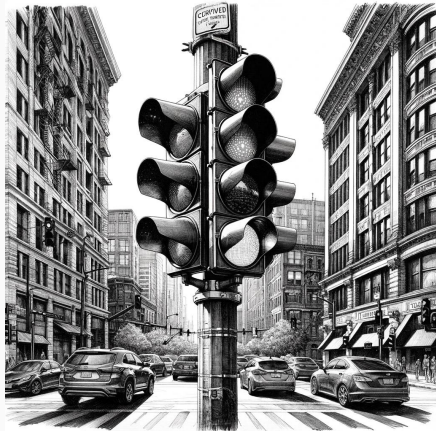
VERISOFT

Descrip
on

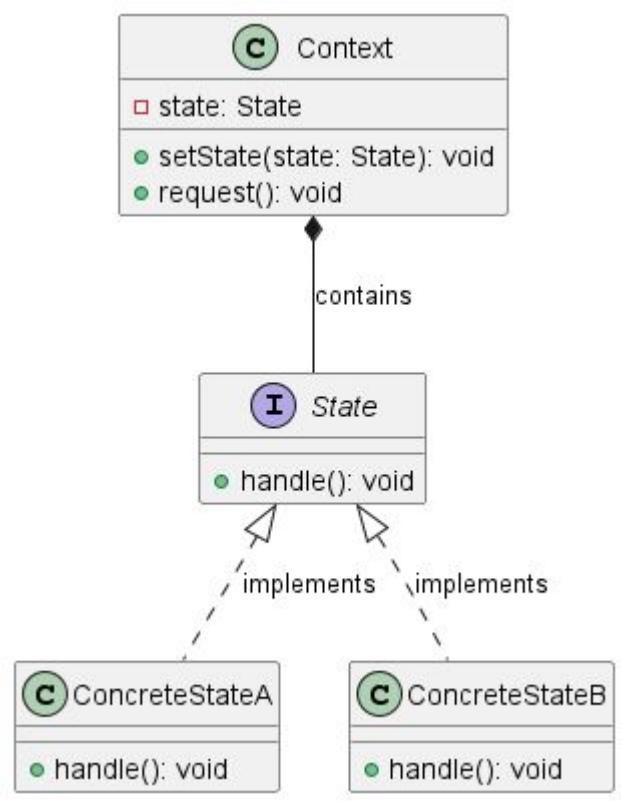


Description

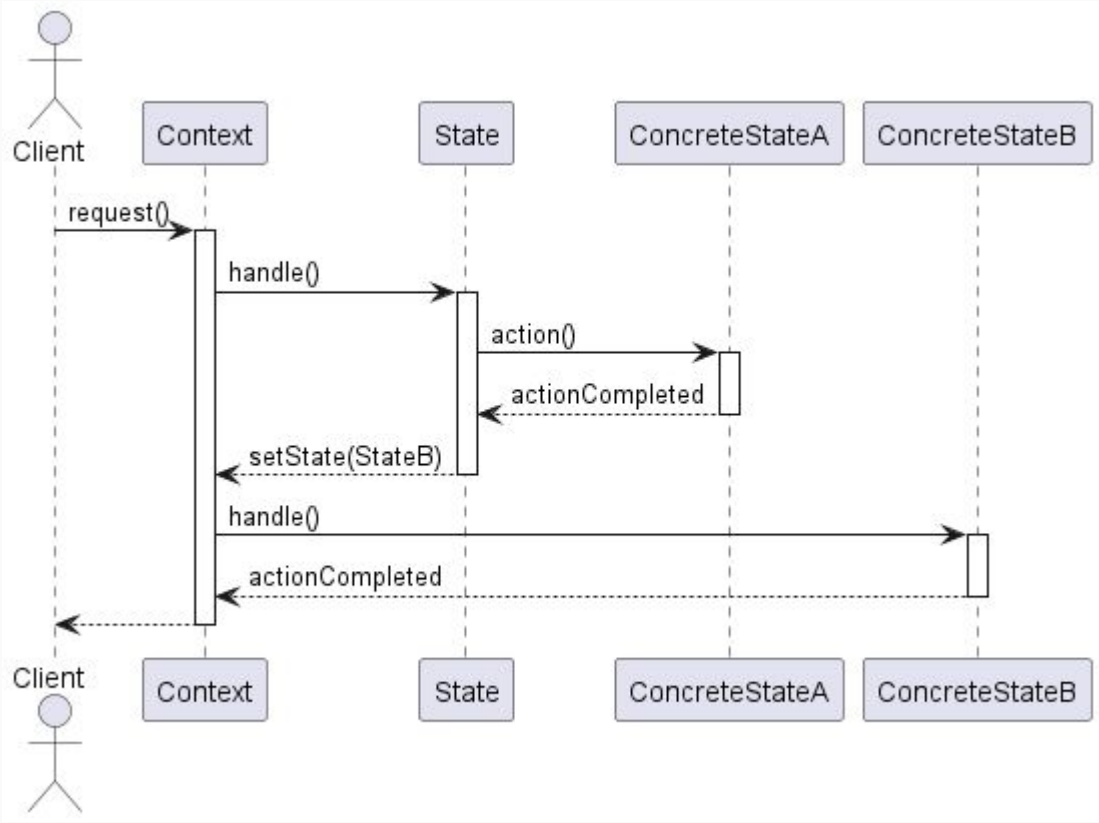
The State design pattern is a behavioral design pattern that allows an object to alter its behavior when its internal state changes. This pattern is akin to implementing a state machine within an object. The core idea is to have a context object that maintains an instance of a concrete state subclass that represents the current state of the object. As the object's state changes, the context object replaces the current state instance with another, corresponding to the new state. This switch changes the behavior of the context object, as it delegates state-specific behavior to the different state objects. Instead of the context object performing behavior directly, it relies on its current state object to do so. This approach simplifies the management of state-dependent behavior and promotes cleaner, more modular code by encapsulating the state-specific behaviors within separate classes.



Class Diagram



Sequence Diagram

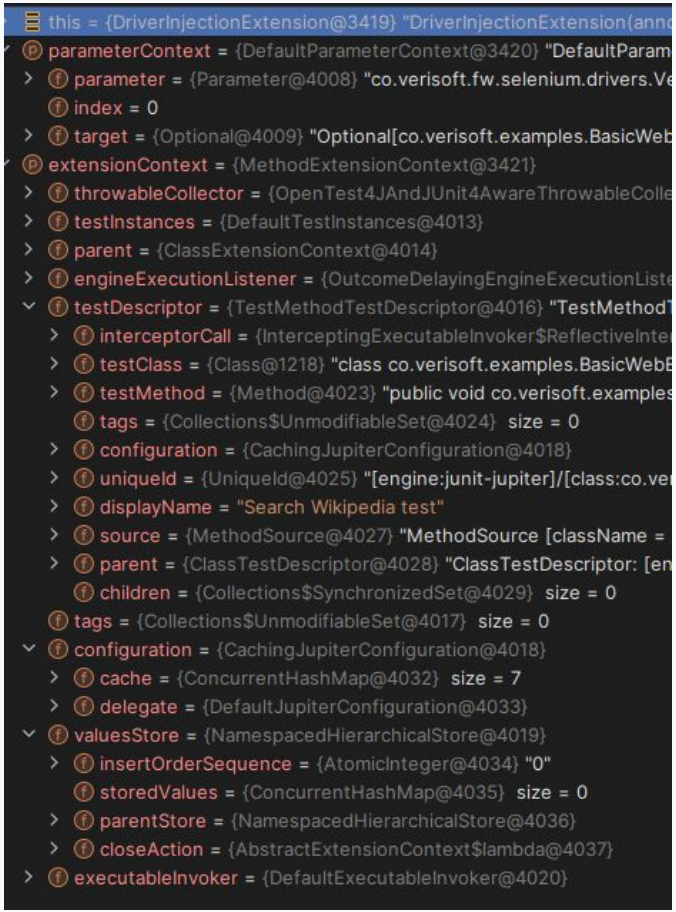


Code Sample

Use cases

- General
  - Workflow Management Systems
  - Document Status
  - E-commerce Order Processing

- In Test Automation
  - Test Data Setup and Teardown
  - Adaptive Test Flows
  - Feature Toggle Management
  - Workflow or Process Testing
  - Context Handling



Happy Coding