SDET Course

Design Patterns - Strategy

3 Types of Design Patterns

VERISOFT

- 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

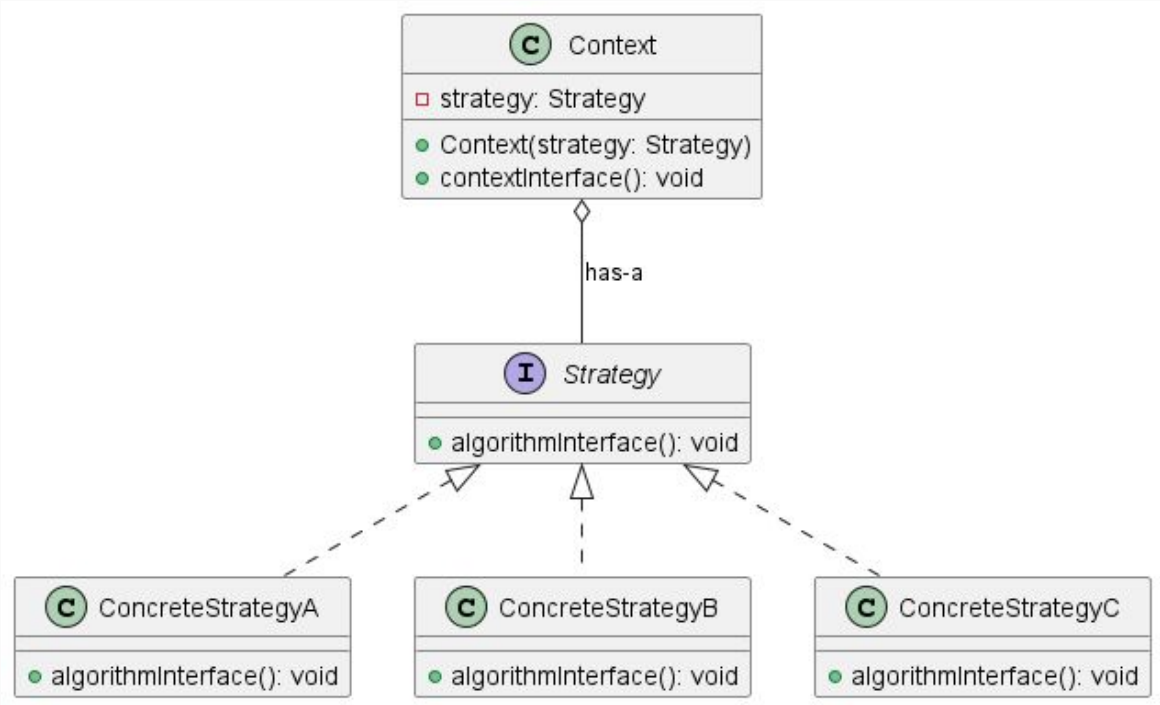
VERISOFT

Descrip
on

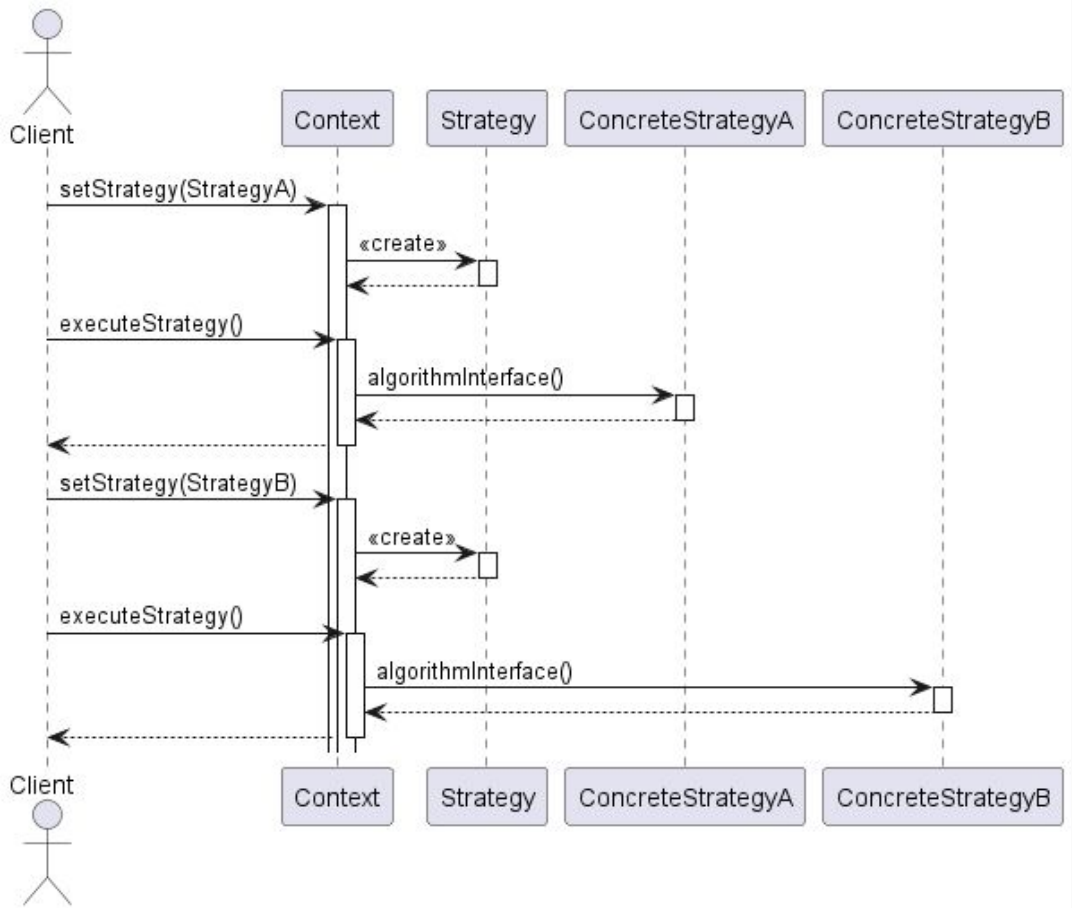
Description

The Strategy Design Pattern is a behavioral design pattern that enables an object to change its behavior or algorithm at runtime. It involves defining a family of algorithms, encapsulating each one, and making them interchangeable within that family. The strategy pattern allows for the variation of the algorithm being used independently from the clients that use it. Essentially, it delegates the responsibility of executing a particular algorithm to a set of interchangeable classes known as strategies. This approach helps in avoiding conditional statements for selecting desired behaviors and promotes the use of composition over inheritance. By applying the strategy pattern, developers can choose the most suitable algorithm at runtime, enhancing flexibility and enabling easier testing and maintenance of the codebase.

Class Diagram



Sequence Diagram



Code Sample

Use cases

- General
  - Different payment methods (credit card, PayPal, bank transfer, etc.)
  - Various compression algorithms (ZIP, RAR, TAR, etc.)
  - Social Media Sharing (Facebook, Twitter, Instagram, etc.)

- In Test Automation
  - Test Data Generation (static datasets, dynamic data generation, data from files, etc.)
  - API Testing (REST, SOAP, GraphQL)

Happy Coding