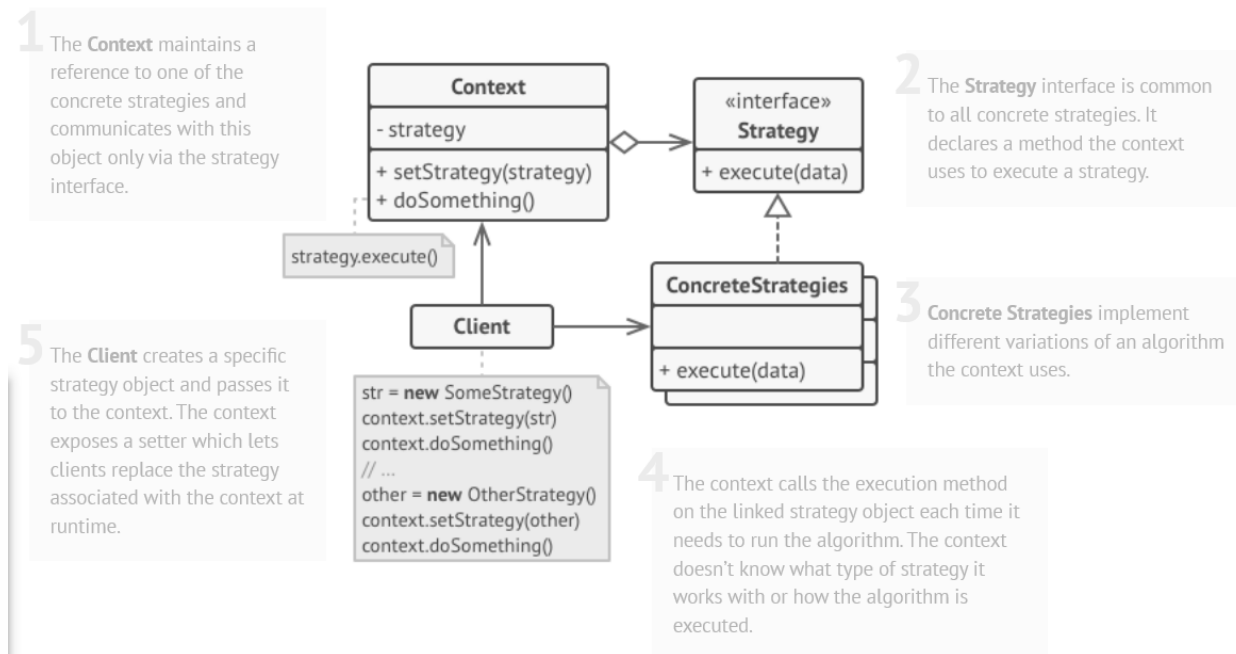# Intent

Strategy is a behavioral design pattern that lets you define a family of algorithms, put each of them into a separate class, and make their objects interchangeable.

Structure

1 The **Context** maintains a reference to one of the concrete strategies and communicates with this object only via the strategy interface.

2 The **Strategy** interface is common to all concrete strategies. It declares a method the context uses to execute a strategy.

3 **Concrete Strategies** implement different variations of an algorithm the context uses.

4 The context calls the execution method on the linked strategy object each time it needs to run the algorithm. The context doesn't know what type of strategy it works with or how the algorithm is executed.

5 The **Client** creates a specific strategy object and passes it to the context. The context exposes a setter which lets clients replace the strategy associated with the context at runtime.

```
str = new SomeStrategy()
context.setStrategy(str)
context.doSomething()
// ...
other = new OtherStrategy()
context.setStrategy(other)
context.doSomething()
```

Applicability

- Use the Strategy pattern when you want to use different variants of an algorithm within an object and be able to switch from one algorithm to another during runtime.
- Use the Strategy when you have a lot of similar classes that only differ in the way they execute some behavior.
- Use the pattern to isolate the business logic of a class from the implementation details of algorithms that may not be as important in the context of that logic.
- Use the pattern when your class has a massive conditional operator that switches between different variants of the same algorithm.