SourceMaking
teaching IT professionals

twitter

→ Log in   ✉ Contact

Design Patterns ▾   Antipatterns ▾   Refactoring ▾   UML ▾

- ⬤ Design Patterns Reference
- 🌟 Design Patterns Book

Strategy Design Pattern

Search   ✖

Intent

01
- Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from the clients that use it.

- Capture the abstraction in an interface, bury implementation details in derived classes.

Problem

02
One of the dominant strategies of object-oriented design is the "open-closed principle".

03
Figure demonstrates how this is routinely achieved - encapsulate interface details in a base class, and bury implementation details in derived classes. Clients can then couple themselves to an interface, and not have to experience the upheaval associated with change: no impact when the number of derived classes changes, and no impact when the implementation of a derived class changes.
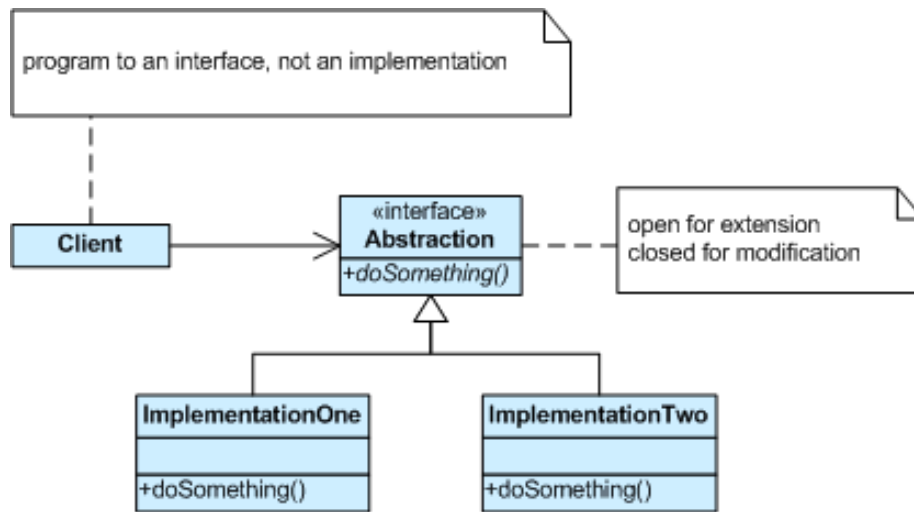
04   A generic value of the software community for years has been, "maximize cohesion and minimize coupling". The object-oriented design approach shown in figure is all about minimizing coupling. Since the client is coupled only to an abstraction (i.e. a useful fiction), and not a particular realization of that abstraction, the client could be said to be practicing "abstract coupling" . an object-oriented variant of the more generic exhortation "minimize coupling".

05   A more popular characterization of this "abstract coupling" principle is "Program to an interface, not an implementation".

06   Clients should prefer the "additional level of indirection" that an interface (or an abstract base class) affords. The interface captures the abstraction (i.e. the "useful fiction") the client wants to exercise, and the implementations of that interface are effectively hidden.

Structure

The Interface entity could represent either an abstract base class, or the method signature expectations by the client. In the former case, the inheritance hierarchy represents dynamic polymorphism. In the latter case, the Interface entity represents template code in the client and the inheritance hierarchy represents static polymorphism.
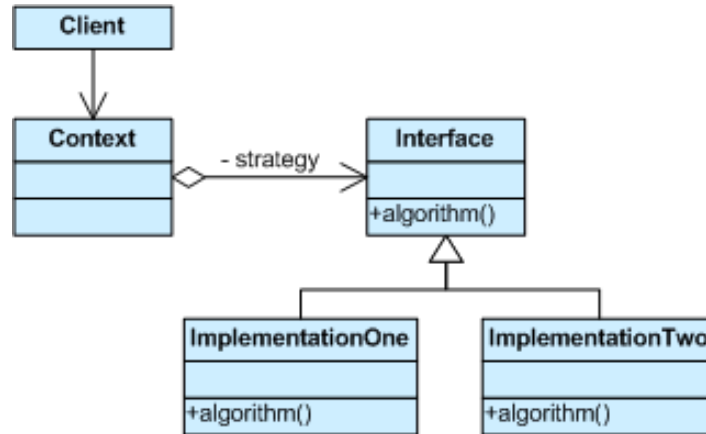
Example

A Strategy defines a set of algorithms that can be used interchangeably. Modes of transportation to an airport is an example of a Strategy. Several options exist such as driving one's own car, taking a taxi, an airport shuttle, a city bus, or a limousine service. For some airports, subways and helicopters are also available as a mode of transportation to the airport. Any of these modes of transportation will get a traveler to the airport, and they can be used interchangeably. The traveler must chose the Strategy based on tradeoffs between cost, convenience, and time.

Personal Car   Taxi   City Bus
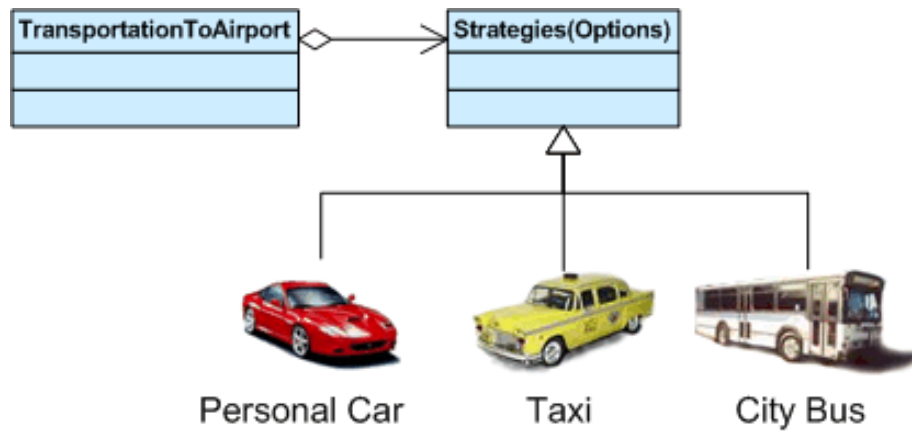
Check list

1. Identify an algorithm (i.e. a behavior) that the client would prefer to access through a "flex point".

2. Specify the signature for that algorithm in an interface.

3. Bury the alternative implementation details in derived classes.

4. Clients of the algorithm couple themselves to the interface.

Rules of thumb

- Strategy is like Template Method except in its granularity.

- State is like Strategy except in its intent.

- Strategy lets you change the guts of an object. Decorator lets you change the skin.

- State, Strategy, Bridge (and to some degree Adapter) have similar solution structures. They all share elements of the 'handle/body' idiom. They differ in intent - that is, they solve different problems.

- Strategy has 2 different implementations, the first is similar to State. The difference is in binding times (Strategy is a bind-once pattern, whereas State is more dynamic).

- Strategy objects often make good Flyweights.

Read next
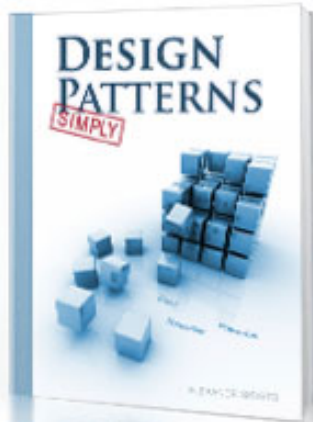
This article is taken from the book **Design Patterns - Simply**.

ALL design patterns are compiled there. The book is written in a clear and simple language that makes it easy to read and understand (just like this article).

It is a part of our Design Patterns Course. We distribute it in PDF format, so it will be available for downloading in 10 seconds!

Learn more…

Strategy code examples

C# examples

- Strategy in C#

C++ examples

- Strategy in C++

Are you a developer? Try out the HTML to PDF API

Delphi examples

- Strategy in Delphi

Java examples

- Strategy in Java

PHP examples

- Strategy in PHP

‹ State Design Pattern   Template Method Design Pattern ›