Design Patterns

Strategy, Template

COMP3607
Object Oriented Programming II

08-Oct-2018

Outline

- Design Patterns
  - Strategy
  - Template

3 Strategy Design Pattern

The Strategy design pattern is an object behavioural pattern.

It defines a family of algorithms, encapsulates each one, and make them interchangeable.

Strategy lets the algorithm vary independently from the clients that use it.

This is achieved by capturing the abstraction in an interface, and burying implementation details in derived classes.

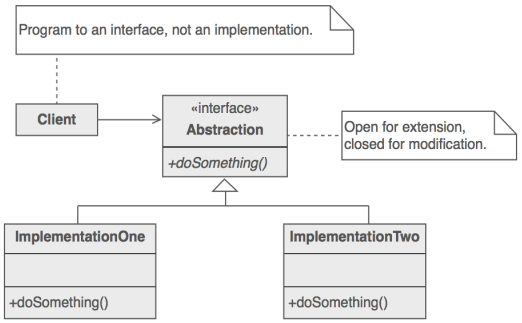
Abstract Coupling

One of the dominant object-oriented design principles is the "Open-Closed principle".

This is routinely achieved by encapsulating interface details in a base class, and putting implementation details in derived classes.

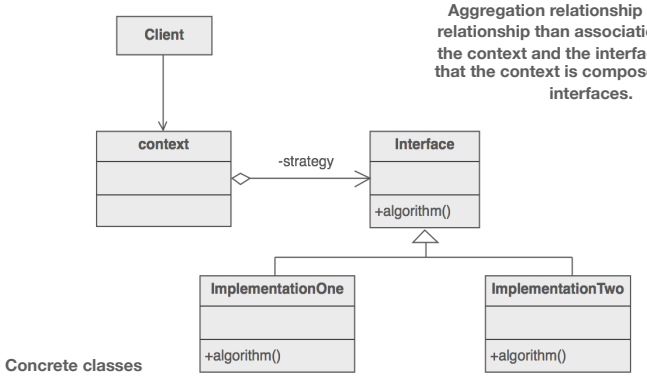
Clients can then couple themselves to an interface, and not have to experience the upheaval associated with change: no impact when the number of derived classes changes, and no impact when the implementation of a derived class changes.

Maximise Cohesion, Minimise Coupling



https://sourcemaking.com/design_patterns/strategy

Strategy (UML) Structure



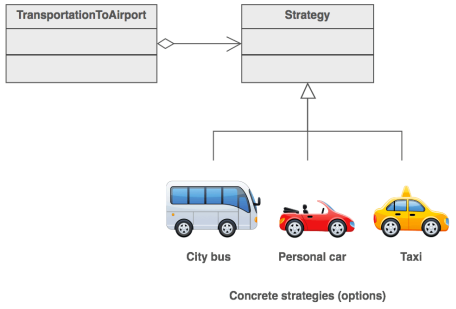
https://sourcemaking.com/design_patterns/strategy

Strategy Design Pattern -Steps-

1. Define the interface of an interchangeable family of algorithms
2. Bury algorithm implementation details in derived classes
3. Clients of the algorithm couple themselves strictly to the interface

Example

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
- a limousine service.
- subways (some airports)
- helicopters (some airports)

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 trade-offs between cost, convenience, and time.

3

Other Examples

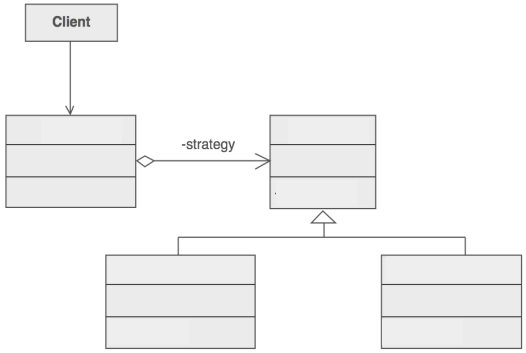
Other examples that use the Strategy pattern would be:

- saving files in different formats
- running various sorting algorithms
- different algorithms for file compression

Exercise

5 mins

Exercise: Use the Strategy Design Pattern to fill in the diagram for a file compression application where we can create either zip or rar files



https://dzone.com/articles/design-patterns-strategy

3

Applicability

Use the Strategy pattern when:

- many related classes differ only in their behaviours
- you need different variants of an algorithm
- an algorithm uses data that clients shouldn't know about
- a class defines many behaviours and these appear to be multiple conditional statements in its operations.

3

Consequences: Strategy

Benefits and Liabilities:

- Families of related algorithms
- An alternative to subclassing
- Strategies eliminate conditional statements
- A choice of implementations
- Clients must be aware of different strategies
- Communication overhead between Strategy and Content
- Increased number of objects

4 Template Design Pattern

- Define the skeleton of an algorithm in an operation, deferring some steps to client subclasses.
- Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.
- Base class declares algorithm 'placeholders', and derived classes implement the placeholders.

4 Variant vs Invariant Steps

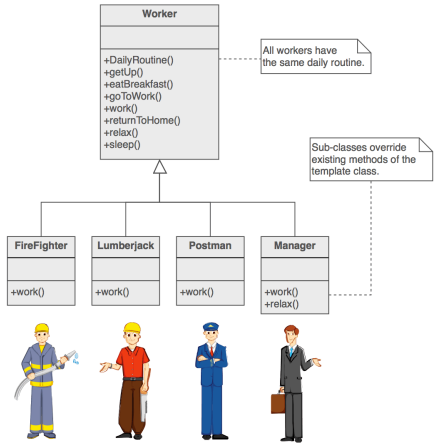
The component designer decides which steps of an algorithm are **invariant** (or standard), and which are **variant** (or customizable).

The invariant steps are implemented in an abstract base class.

The variant steps are either given a default implementation, or no implementation at all.

The variant steps represent "**hooks**", or "placeholders", that can, or must, be supplied by the component's client in a concrete derived class.

4 Example



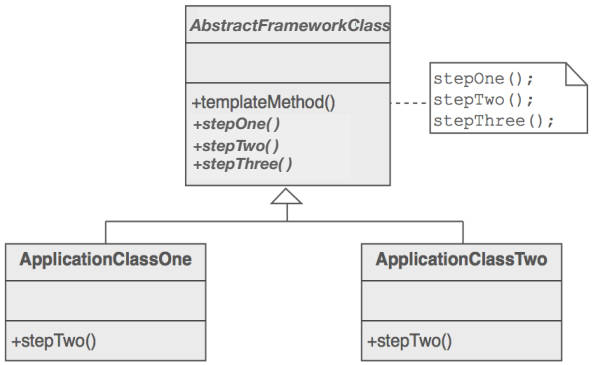
https://sourcemaking.com/design_patterns/template_method

4 Template Method Operations

- **Concrete methods:** Standard complete methods that are useful to the subclasses. These methods are usually utility methods.
- **Abstract methods**: Methods containing no implementation that must be implemented in subclasses.
- **Hook methods:** Methods containing a default implementation that may be overidden in some classes. Hook methods are intended to be overridden, concrete methods are not.
- **Template methods:** A method that calls any of the methods listed above in order to describe the algorithm without needing to implement the details.

4

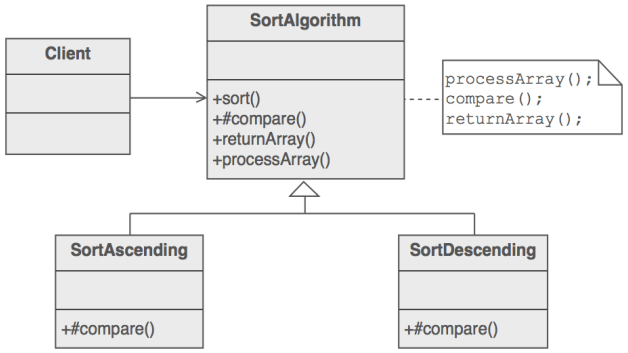
Template (UML)



https://sourcemaking.com/design_patterns/template_method

4

Example

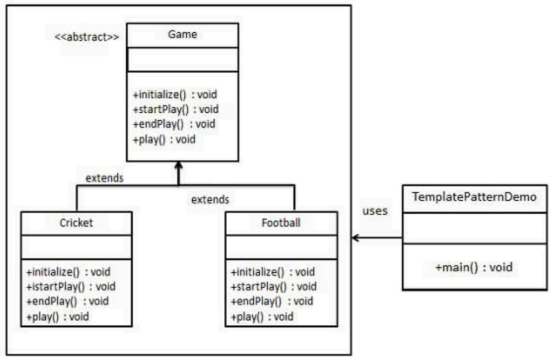


https://sourcemaking.com/design_patterns/template_method

Exercise

10 mins

Exercise: Use the Template Design Pattern to write code diagram to create a *Game* abstract class defining operations with a template method set to be final so that it cannot be overridden. *Cricket* and *Football* are concrete classes that extend *Game* and override its methods



https://www.tutorialspoint.com/design_pattern/template_pattern.htm

4

Design Pattern Comparisons -Template vs Strategy-

- Strategy is like Template Method except in its granularity.
- Template Method uses inheritance to vary part of an algorithm. Strategy uses delegation to vary the entire algorithm.
- Strategy modifies the logic of individual objects. Template Method modifies the logic of an entire class.

4 Applicability: Template

Template design pattern is used:

- to implement the invariant parts of an algorithm once and leave the varying behaviour implementation details up to the subclasses
- when common behaviour among subclasses should be factored and localised in a common class to avoid code duplication.
- to control subclasses extensions.

4 Consequences: Template

- Template methods lead to an inverted control structure: “the Hollywood principle” - “Don’t call us, we’ll call you”
  - A parent class calls the operations of a subclass and not the other way around.
- Fundamental technique for code reuse especially for class libraries (factoring out common behaviour).
- Hook operations: provide default behaviour that subclasses extend by default.
  - Template methods must specify which operations are hooks (may be overridden) and which are abstract (must be overridden).

References

- Design Patterns: online reading resources
  - https://sourcemaking.com/design_patterns/strategy
  - https://sourcemaking.com/design_patterns/template_method
  - https://dzone.com/articles/design-patterns-template-method
  - https://dzone.com/articles/design-patterns-strategy