

# Design Pattern

Milad Abolhassani

tuxgeek.ir  
milad@tuxgeek.ir

Slides:

<http://slideshare.com/miladas>

<http://miladas.github.io/slides>



- Class
- Property
- Method
- Inheritance
- Polymorphism
- Interface
- Abstract class
- Encapsulation
- Public/protected/private
- Final

**SOLID**

4

SOLID

# SOLID

5

- Single Responsibility
- Open/Closed
- Liskov substitution
- Interface Segregation
- Dependency Injection

# Single Responsibility

6



# Open/Closed

7

- Meyer's open/closed
  - Once class completed, never should modified
- Polymorphic open/closed principle
  - Interface
  - The interface is open for extension but close for modification

# Liskov substitution

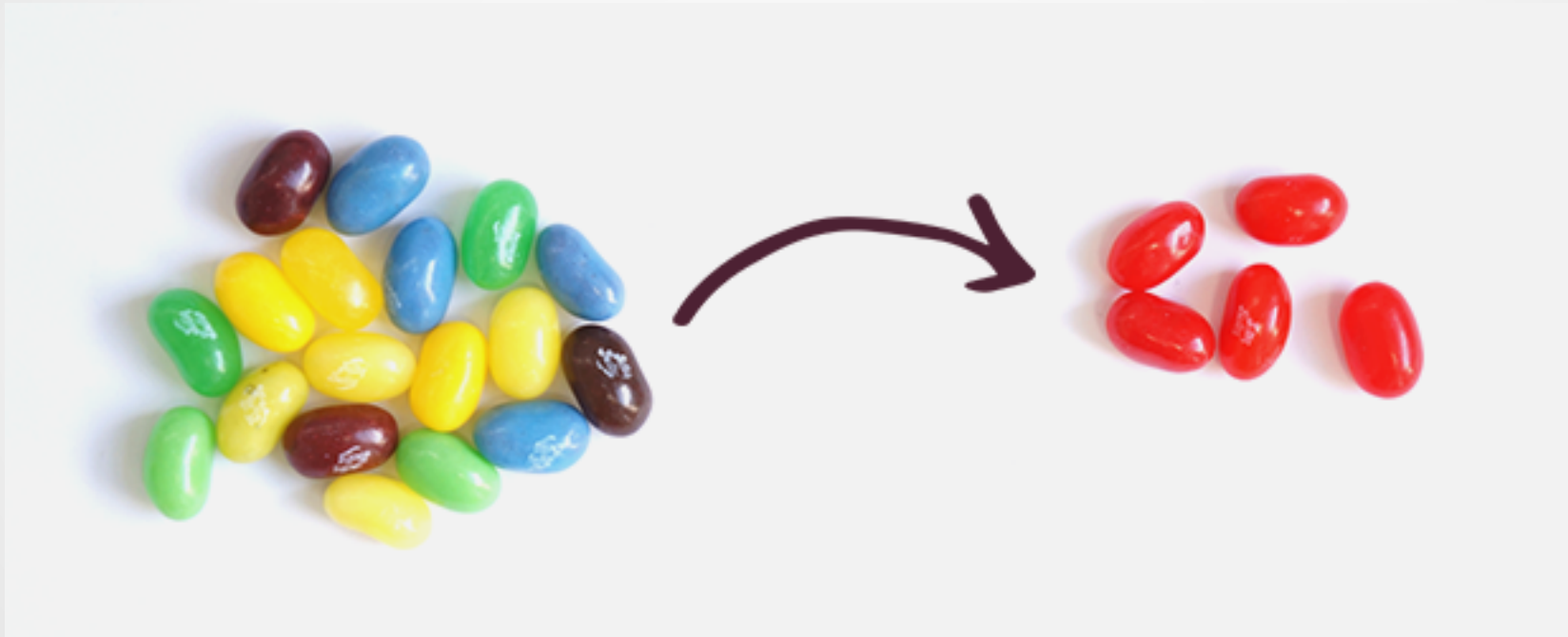
8

- Program to an interface rather than of it's implementation
- Design by contract
- Do not extend the interface



# Interface Segregation

9



# Dependency injection

10



# What is design pattern?

11

- In software engineering, a design pattern is a general reusable solution to a commonly occurring problem.



- Patterns originated as an architectural concept by Christopher Alexander (1977/79).
- Design patterns gained popularity in computer science after the book *Design Patterns: Elements of Reusable Object-Oriented Software* was published in 1994.
  - Gang of Four
  - Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John

# Gang of four

13



# Why design pattern?

14

- Clarity
- Correctness
- Same vocabularies
  - Avoid miss communications
- Reuse
- Save time, trial and error
- Can speed up the development process
  - by providing tested, proven development paradigms
- We are not reusing code, we are reusing experience

# Category

15

- Creational design patterns
- Structural design patterns
- Behavioral design patterns

# Creational design patterns

16

- These design patterns provide a way to create objects while hiding the creation logic.



# Creational design patterns

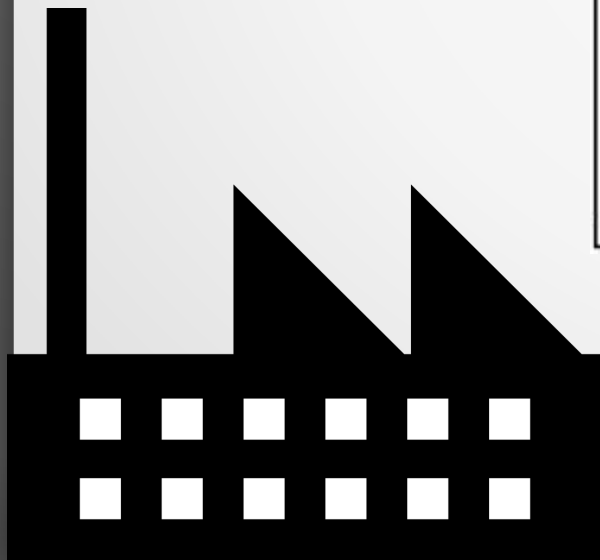
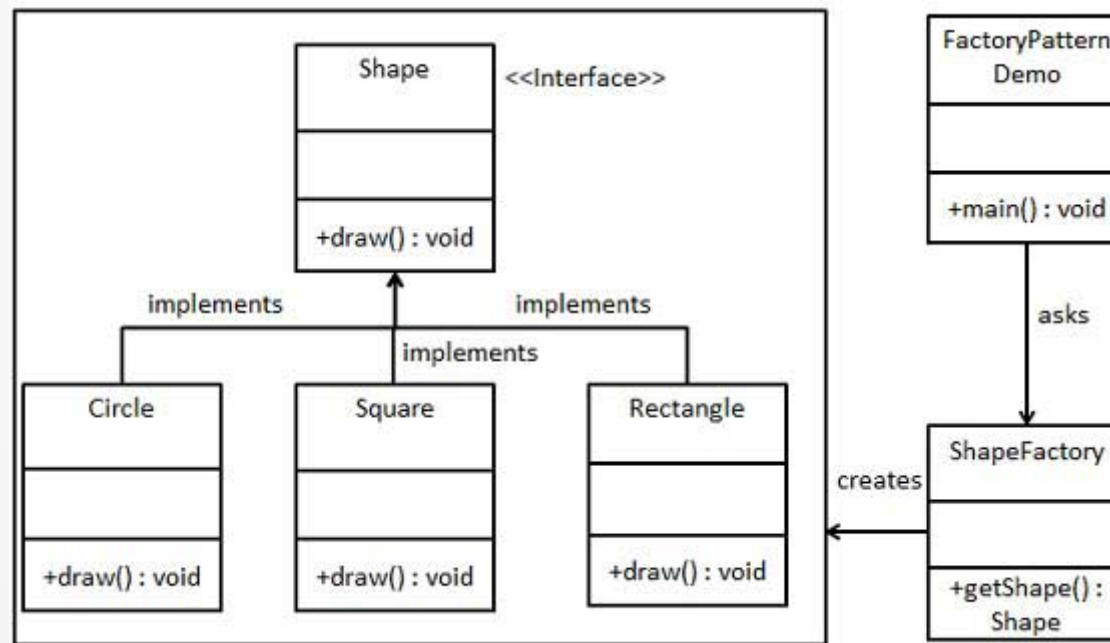
17

- Factory method
- Singleton
- Prototype
- Builder

# Factory Method

18

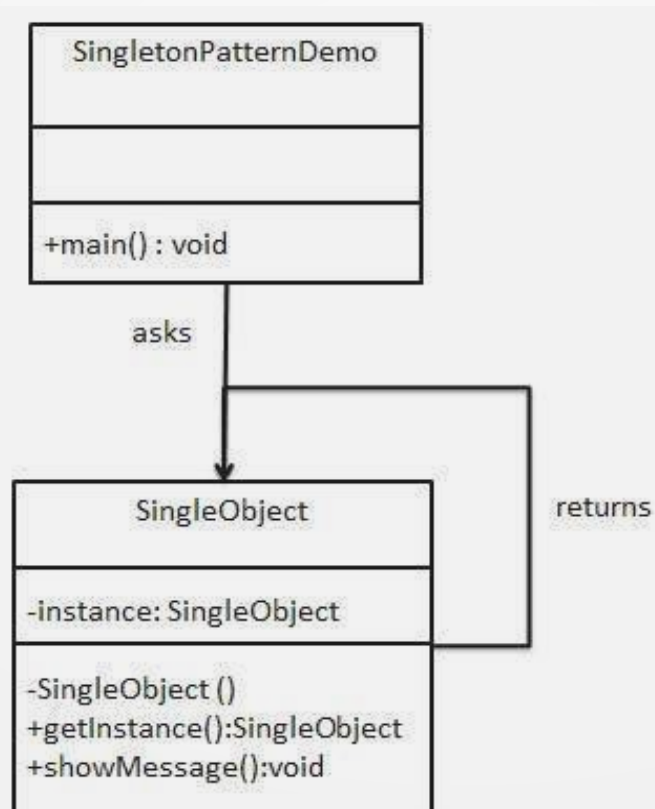
- Factory Method lets a class defer instantiation to subclasses.



# Singleton

19

- This pattern involves a single class which is responsible to create an object while making sure that only single object gets created.



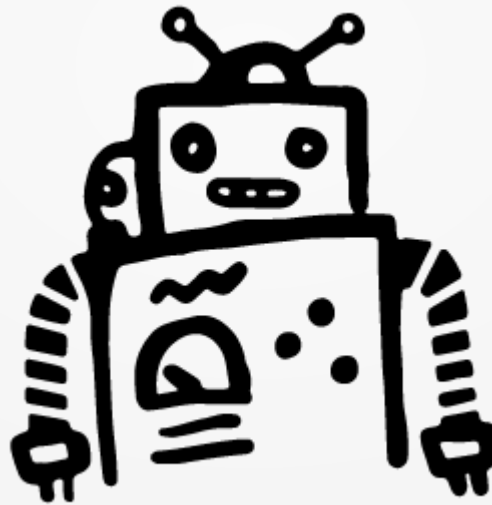
# Prototype

20

- Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.



- Separate the construction of a complex object from its representation so that the same construction process can create different representations.





# Structural design patterns

23

- These design patterns are all about Class and Object composition. Concept of inheritance is used to compose interfaces and define ways to compose objects to obtain new functionalities.

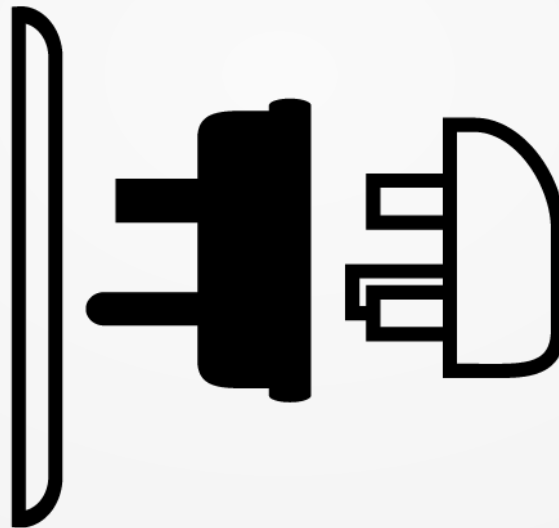
# Structural design patterns

24

- Adapter
- Decorator
- Facade
- Flyweight
- Proxy



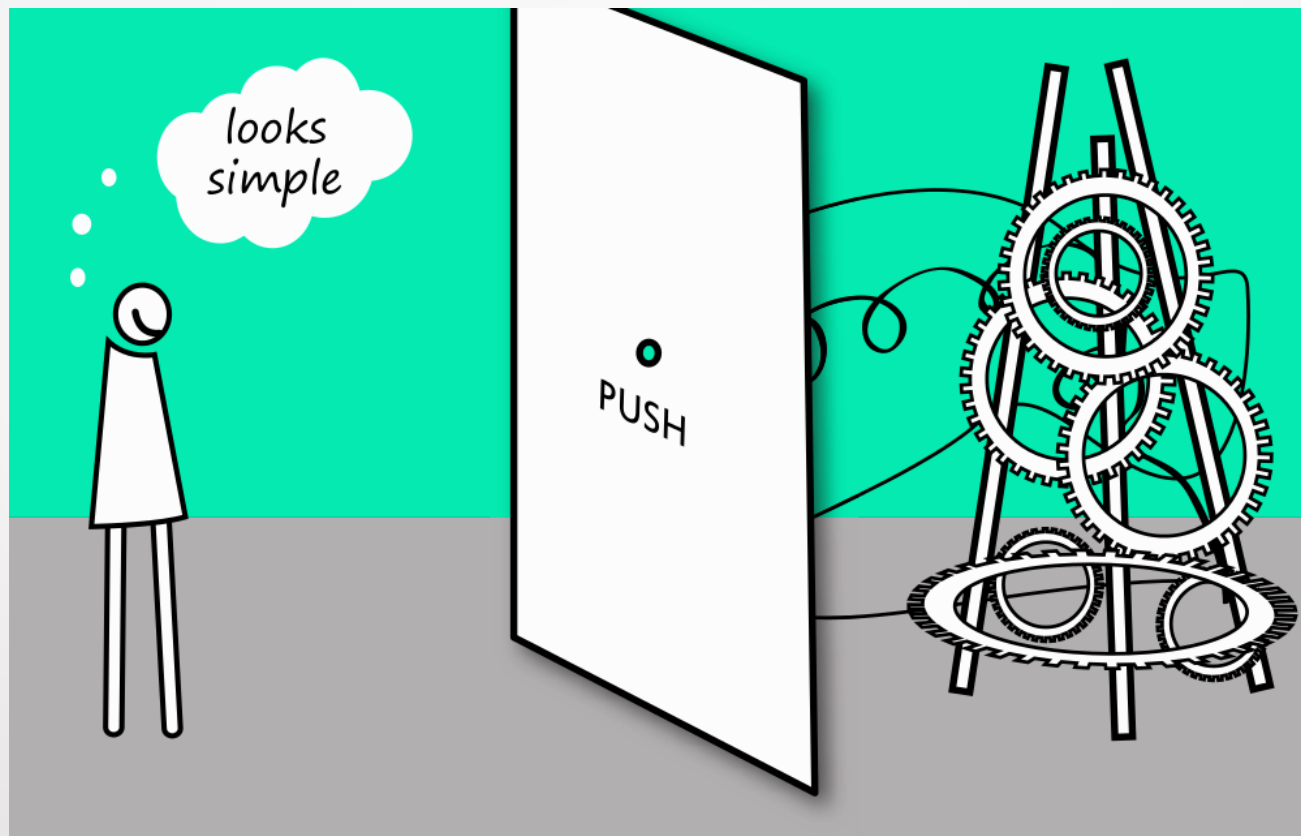
- Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.



- Attach additional responsibilities to an object dynamically.



- Wrap a complicated subsystem with a simpler interface.

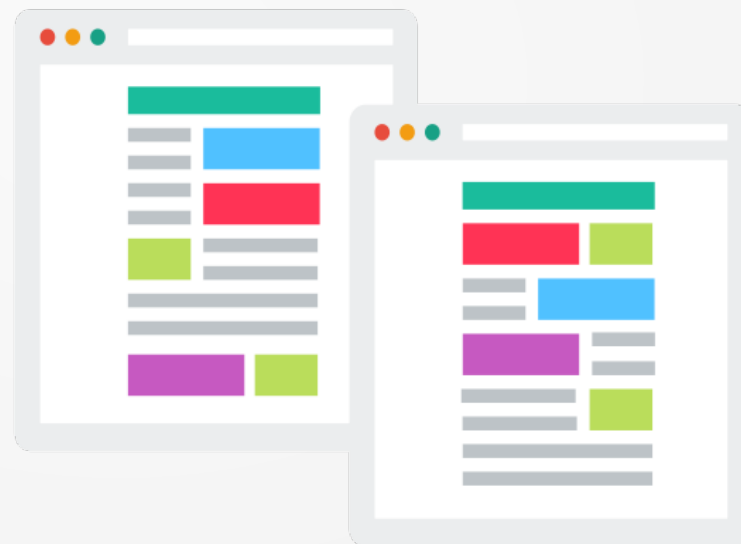


# Flyweight

28

- The Flyweight uses sharing to support large numbers of objects efficiently.

Browser loads images  
just once and then  
reuses them from pool:



- Use an extra level of indirection to support distributed, controlled, or intelligent access.



# Behavioral design patterns

30

- These design patterns are all about Class's objects communication. Behavioral patterns are those patterns that are most specifically concerned with communication between objects.

# Behavioral design patterns

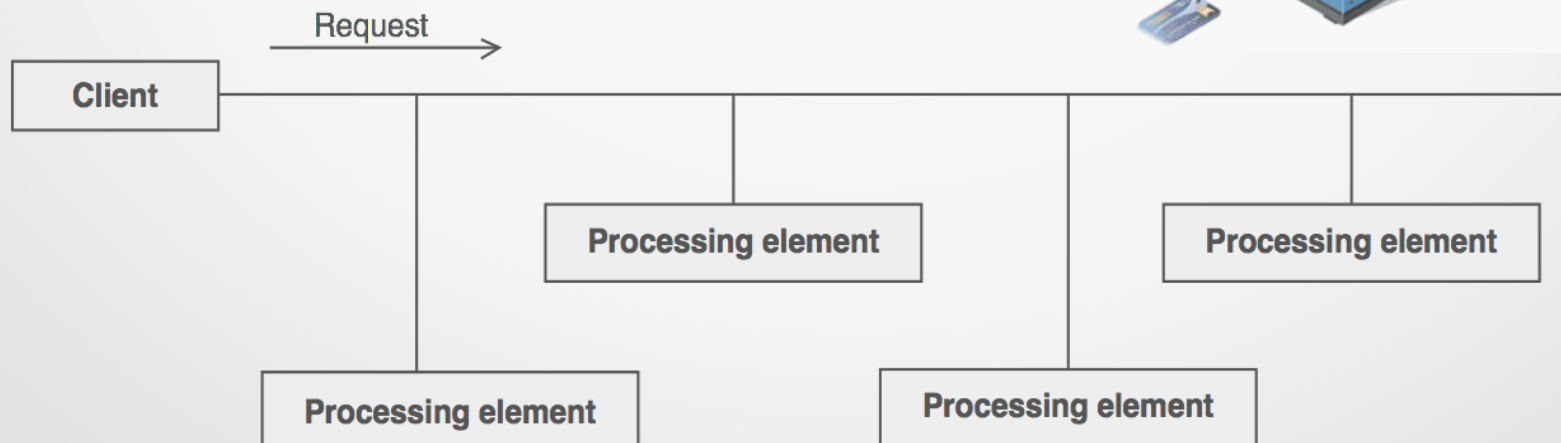
31

- Chain of responsibility
- Mediator
- Observer
- State
- Strategy
- Template method

# Chain of responsibility

32

- The Chain of Responsibility pattern avoids coupling the sender of a request to the receiver by giving more than one object a chance to handle the request.

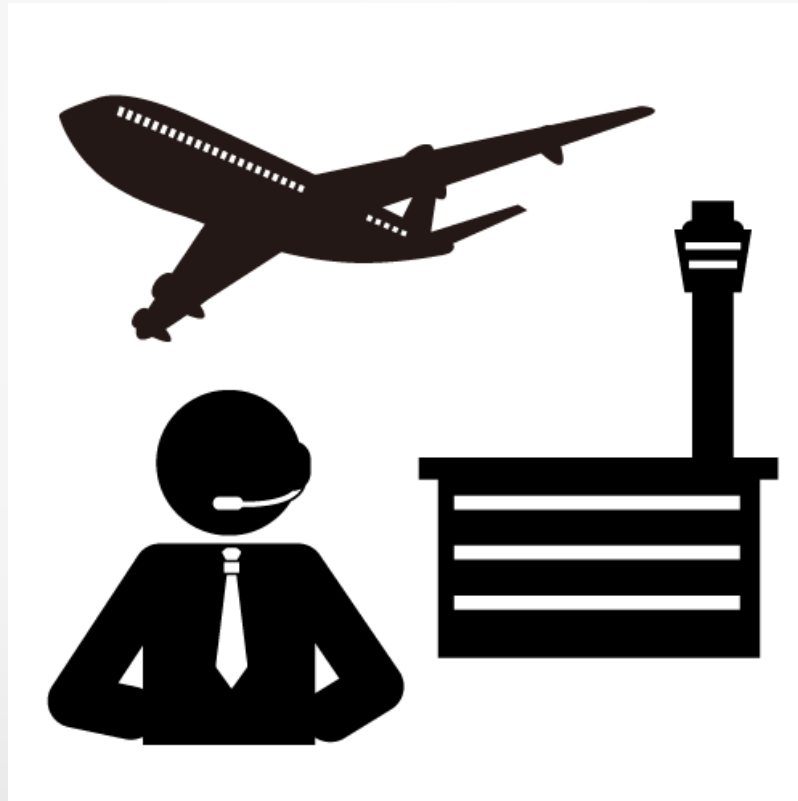




# Mediator

33

- The Mediator defines an object that controls how a set of objects interact. Loose coupling between colleague objects is achieved by having colleagues communicate with the Mediator, rather than with each other.



# Observer

34

- Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its dependent objects are to be notified automatically.



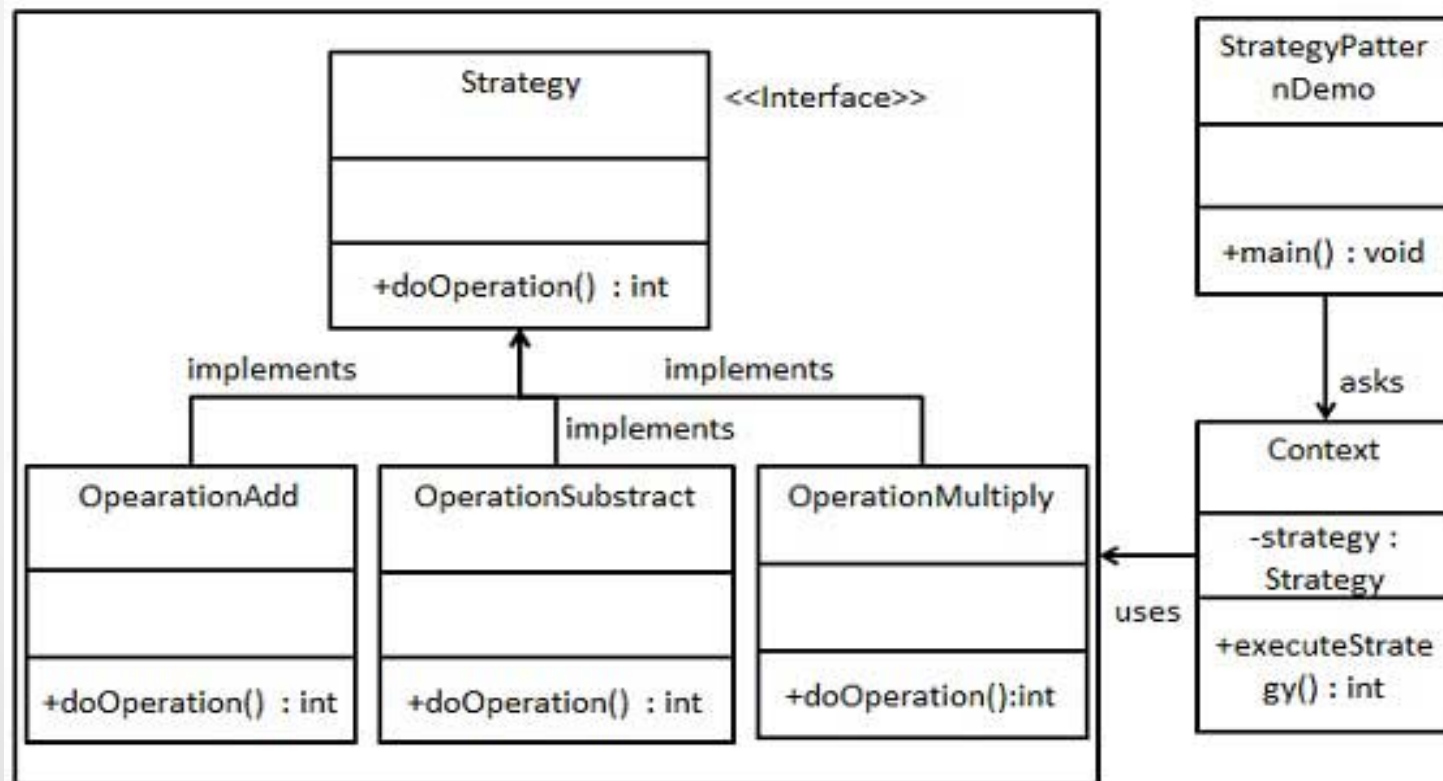
- The State pattern allows an object to change its behavior when its internal state changes.



# Strategy

36

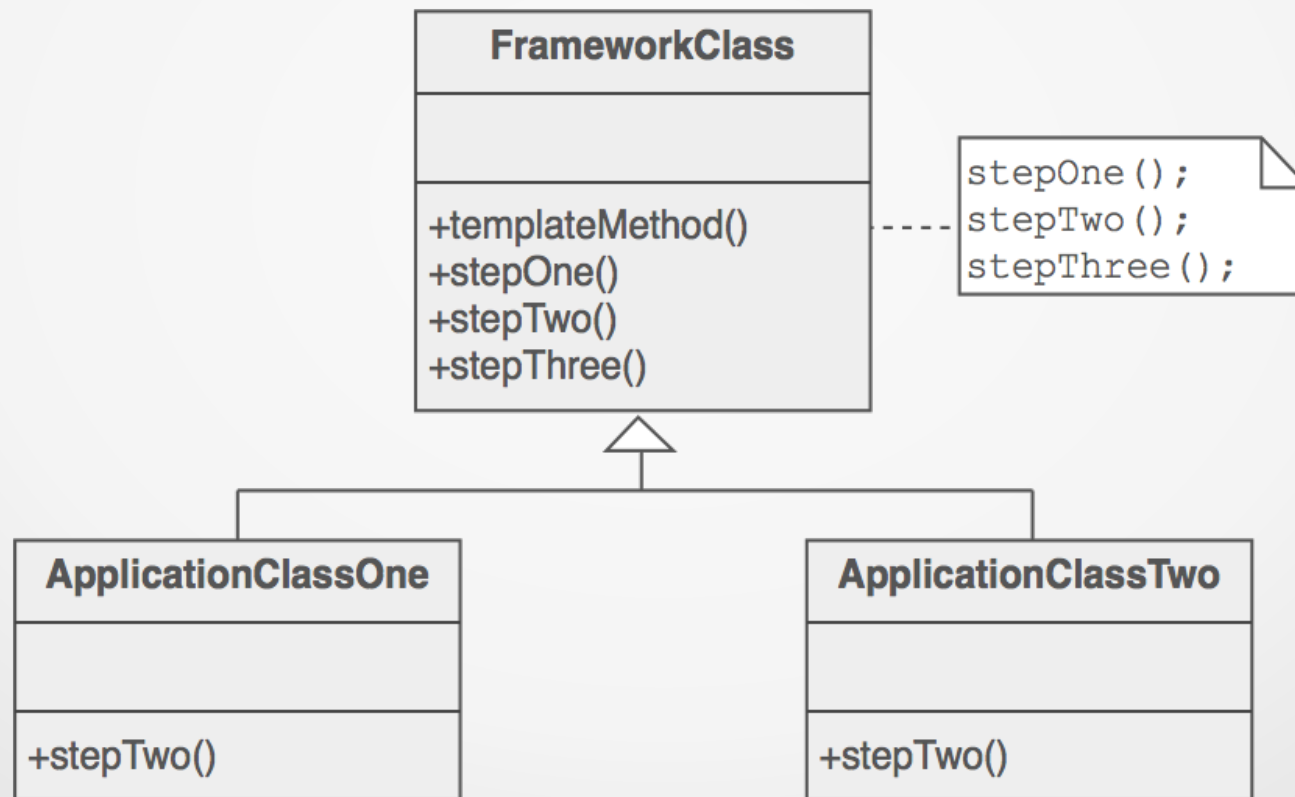
- In Strategy pattern, a class behavior or its algorithm can be changed at run time.



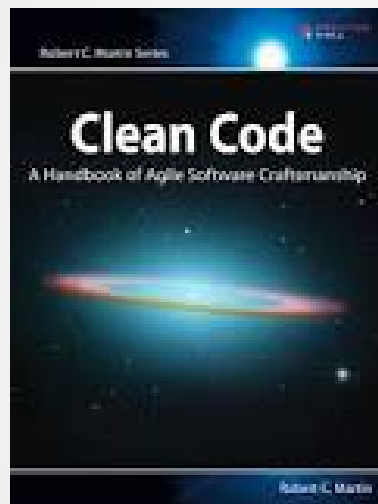
# Template method

37

- The Template Method defines a skeleton of an algorithm in an operation, and defers some steps to subclasses.



# Thank you



# References

- JavaWorld
- SourceMaking
- TutorialsPoint
- Informit
- geekswitblogs
- Wikipedia [1] [2]