VISSOFT

Bài 23
Behavioral Design Pattern

Module: ADVANCED PROGRAMMING WITH JAVA

Mụctiêu

- Trình bày *được ý* nghĩa *của* các Behavioral Design Pattern
- Triển khai *được* Template Method Design Pattern
- Triển khai *được* Command Design Pattern
- Triển khai *được* Observer Design Pattern
- Triển khai *được* Strategy Design Pattern

Behavioral Design Pattern

VISSOFT

- Là nhóm các Design Pattern được sử dụng để giải quyết các vấn đề phổ biến trong giao tiếp giữa các đối tượng.
- Một số Pattern thông dụng trong nhóm:
  - Command
  - Observer
  - Strategy
  - TemplateMethod

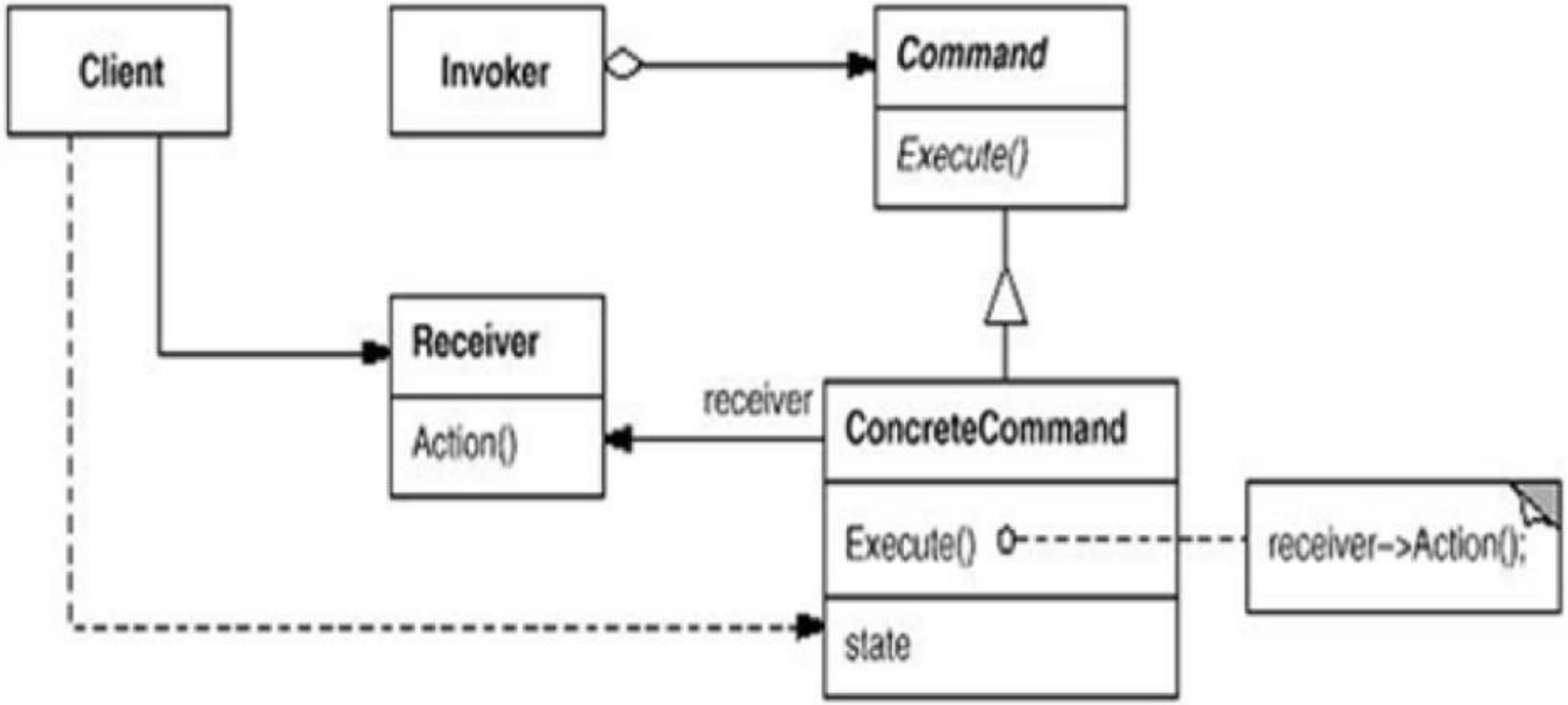
CommandPattern

VISSOFT

- Vấn đề\bài toán: Cần đưa ra các yêu cầu cho đối tượng mà không biết bất cứ điều gì về hoạt động được yêu cầu hoặc bên nhận yêu cầu.
- Giải pháp:
  - Đóng gói yêu cầu thành một đối tượng do đó cho phép bạn sử dụng request như một tham số tới các client khác nhau.
  - Chuyển "triệu gọi phương thức trên một đối tượng" sang trạng thái đối tượng đầy đủ.
- Nếu request là đối tượng nghĩa là có thể cho nó vào hàng chờ, ghi lại lịch sử yêu cầu hoặc có thể khôi phục các thao tác yêu cầu cũ…

Cácthànhphầntrong CommandPattern

VISSOFT



Cácthànhphầntrong CommandPattern

VISSOFT

- **Command**: *Định* nghĩa một interface *để* thực hiện một hành *động* (Execute). Là *đối* tượng lưu giữ Request và State của một *đối* tượng tại một thời *điểm*.
- **ConcreteCommand**: Cài *đặt* Excute bằng cách khai báo các hoạt *động* tương *ứng* trên Receiver.
- **Client**: tạo ra một *đối* tượng **ConcreteCommand** và thiết lập Receiver của nó.
- **Invoker**: Là nơi *lưu trữ* và *phát sinh* mỗi Request dưới dạng Command Object. Quyết *định* khi nào thực hiện nó.
- **Receiver**: Là *đối* tượng thực hiện lệnh trên mỗi yêu cầu.

TriểnkhaiCommandPattern

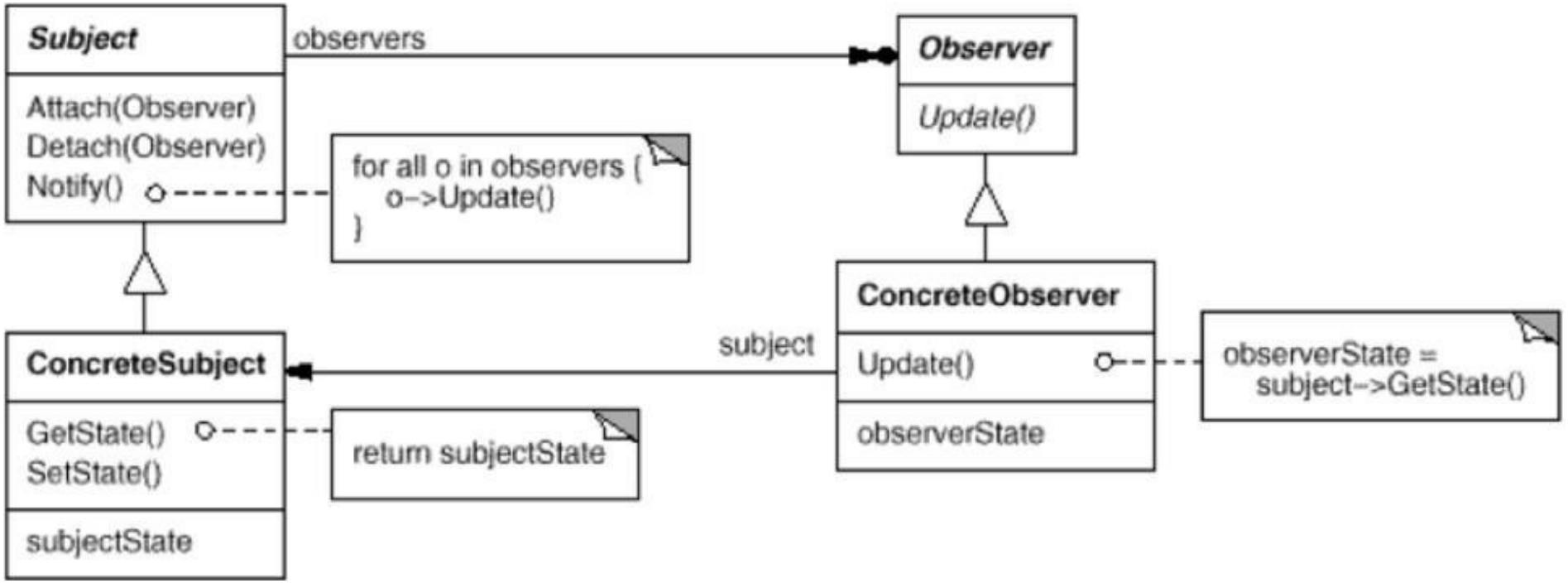
ObserverPattern

VISSOFT

- Observer thuộc nhóm Behavioral là một mẫu thiết kế dành cho việc một đối tượng khi thay đổi trạng thái của bản thân nó thì các đối tượng đính kèm theo cũng sẽ được thông báo.
- Observer gồm một đối tượng gọi là **subject**, đối tượng này duy trì một danh sách các thành phần phụ thuộc nó gọi là **observer**, và thông báo tới chúng một cách tự động về bất cứ thay đổi nào, thường thì bằng cách gọi một phương thức của chúng.

Các thành phần trong Observer Pattern

VISSOFT



Các thành phần trong Observer Pattern

VISSOFT

- **Subject**: Ở đây Subject được hiểu như "ông chủ", là một interface với các phương thức
  - attach **điều thêm một nhân viên đến làm việc** (attach observer object)
  - detach **rút một nhân viên đi nơi khác** (detach observer object)
  - notify **thông báo cho toàn bộ nhân viên làm một việc gì đó** (notify observer update)
- **ConcreteSubject**: Nhiệm vụ là lưu trữ trạng thái của các ConcreteObserver objects và từ các trạng thái này sẽ gửi đi thông báo mỗi khi trạng thái bị thay đổi
  - setState optional – thiết đặt trạng thái cho Subject
  - getState optional - lấy trạng thái hiện tại của Subject

Các thành phần trong Observer Pattern

VISSOFT

- **Observer**
  - Là một interface được định nghĩa với method update, phương thức này sẽ nhận thay đổi mỗi khi có thông báo từ Subject.
- **ConcreteObserver**
  - Luôn duy trì một tham chiếu đến một ConcreteSubject
  - Cần phải lưu trữ trạng thái phù hợp với trạng thái của Subject
  - Thực thi phương thức update của Observer để trạng thái luôn đồng nhất với trạng thái của Subject (sử dụng mỗi khi có notify)

Triển khai Observer Pattern

VISSOFT

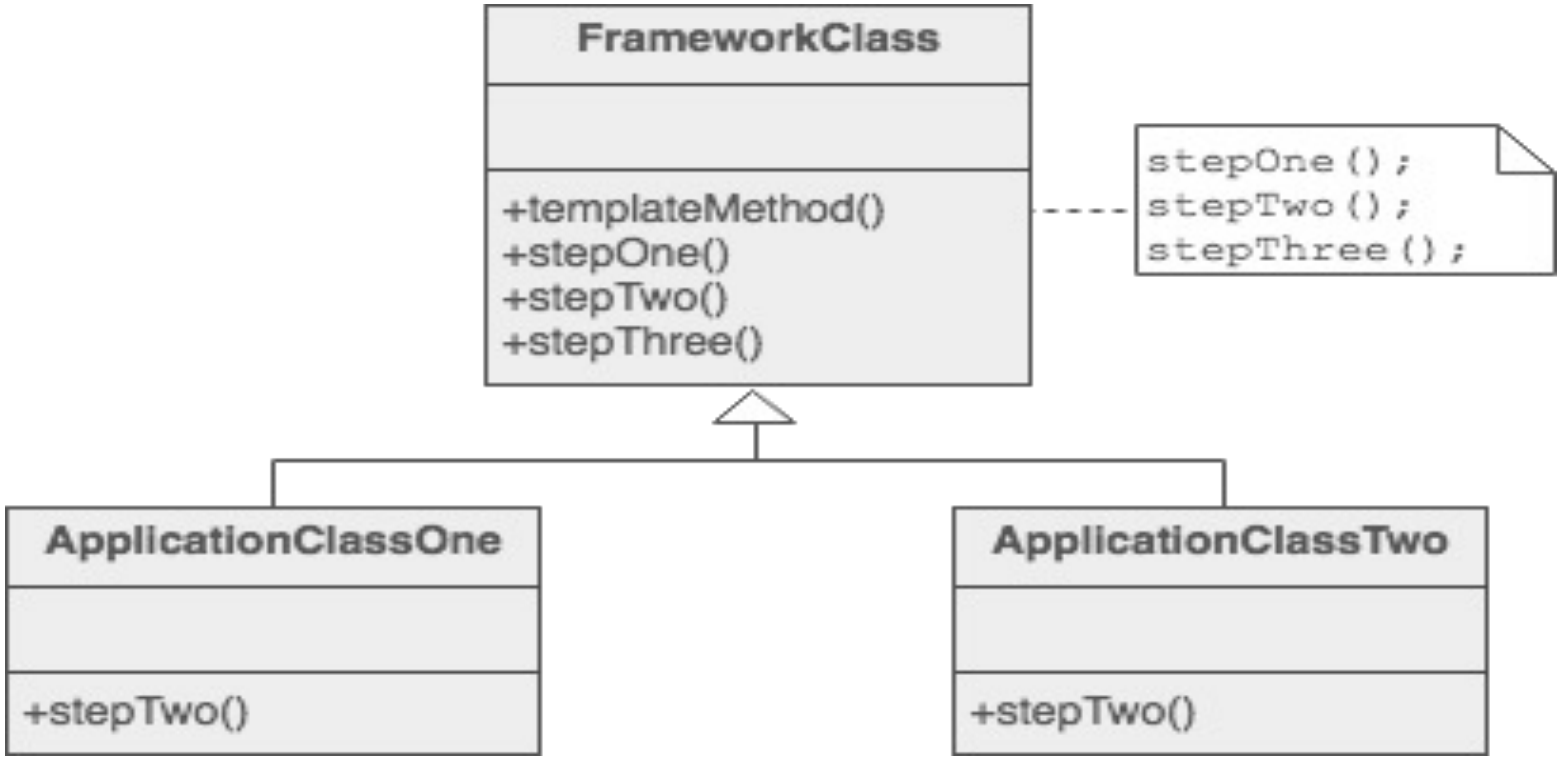
Template Method Pattern

VISSOFT

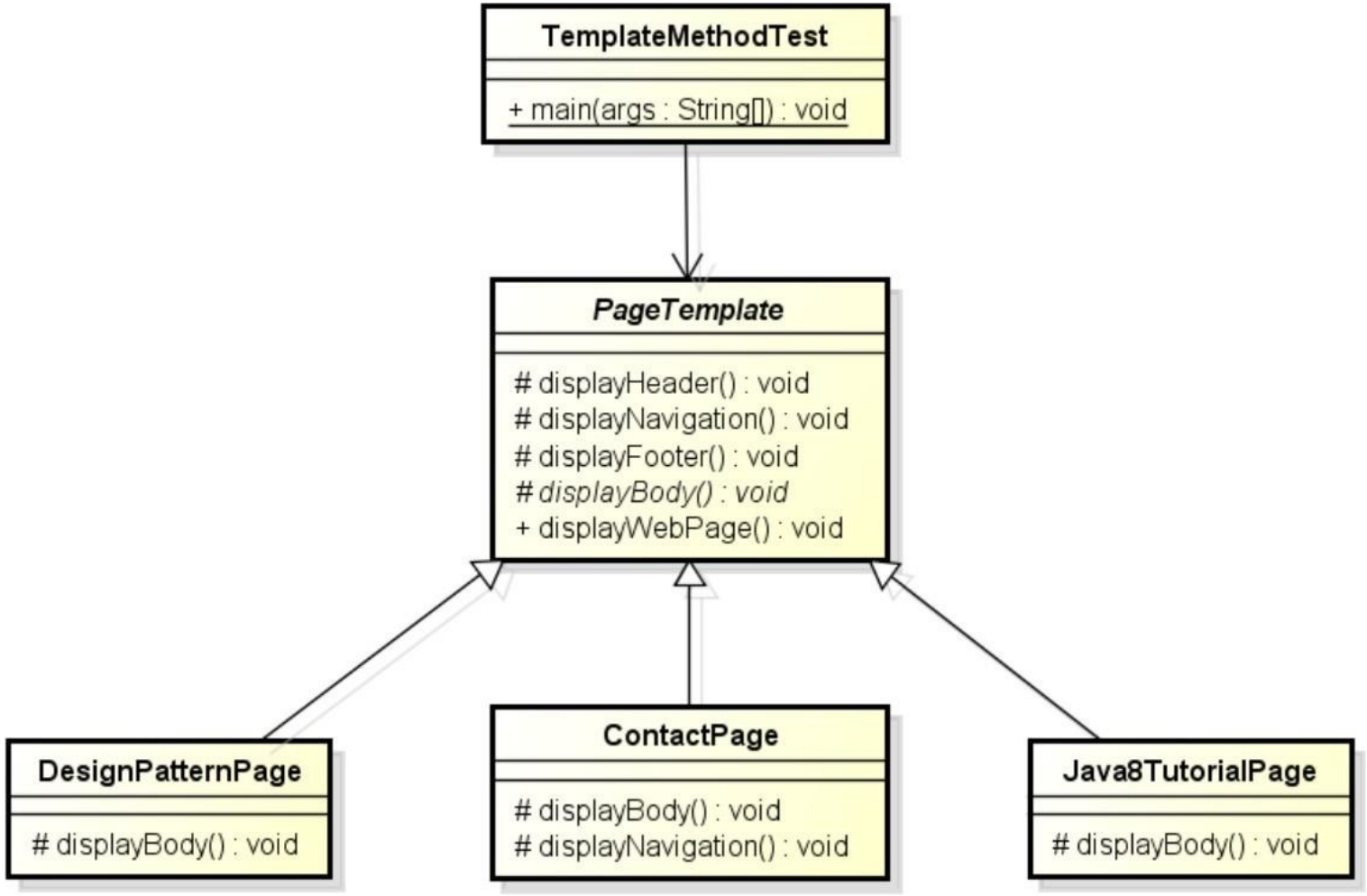
- Mẫu Template Method định nghĩa bộ khung của một thuật toán trong một chức năng chuyển giao việc thực hiện nó cho các lớp con.
- Mẫu Template Method cho phép lớp con định nghĩa lại cách thực hiện của một thuật toán, mà không phải thay đổi cấu trúc thuật toán.
- Ví dụ: Khi chúng ta có một template đã có sẵn header, footer, navigation cho trang web, chỉ riêng phần boby là để trống và sẽ hiển thị nội dung theo từng page. Nghĩa là chỉ có phần body là sẽ thay đổi theo từng page, còn các phần khác sẽ giống nhau cho tất cả các page. Trừ một số page đặc biệt mà nội dung của header, footer, navigation sẽ thay đổi.

Cácthànhphầntrong TemplateMethodPattern

- Trongmẫunày,gồmlớpcơsở(lớpcha)*được*khaibáosẵnthuật toánvàlớpdẫnxuất(lớpcon)thựcthinó.



Triển khai Template Method



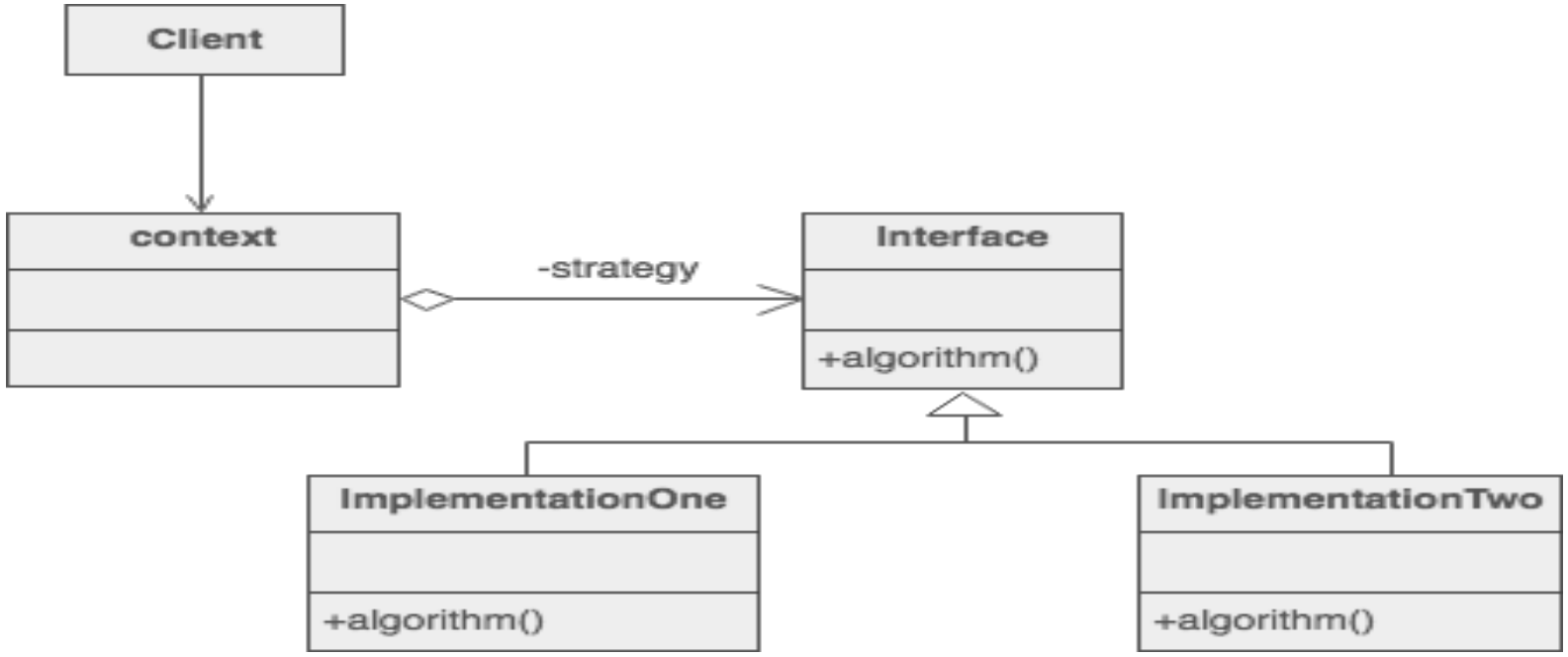
StrategyPattern

VISSOFT

- Strategy định nghĩa một họ các giải thuật khác nhau, mỗi giải thuật được triển khai bởi một lớp cụ thể và chúng có thể hoán đổi cho nhau tùy vào ngữ cảnh.
- Strategy giúp các giải thuật khác nhau độc lập với client sử dụng nó.
- Ví dụ, một lớp thực hiện nhiệm vụ so sánh dữ liệu đầu vào có thể sử dụng mẫu thiết kế Strategy để tự động lựa chọn giải thuật cho việc này dựa trên loại dữ liệu, nguồn gốc của chúng, lựa chọn của người dùng hay các yếu tố khác.

Các thành phần trong Strategy Pattern

- Strategy đơn giản là interface, vì thế chúng ta có thể hoán đổi các thuật toán ConreteStrategy mà không ảnh hưởng tới Context.
- Context có thể có bất cứ thứ gì khi nó yêu cầu thay đổi hành vi.



TriểnkhaiStrategyPattern

VISSOFT

[Thực hành] Triển khai Template Method

VISSOFT

[Thực hành] Triển khai Command

VISSOFT

[Thực hành] Triển khai Observer

VISSOFT

[Thực hành] Triển khai Strategy

VISSOFT

[Bài tập] Áp dụng Strategy Pattern

VISSOFT

Tổng kết

- Behavioral là nhóm các Design Pattern được sử dụng trong giao tiếp giữa các đối tượng.
- Trong Command các yêu cầu từ máy khách gửi lên sẽ được đóng gói thành đối tượng. Từ đó bạn có thể sử dụng request như một tham số tới các client khác nhau.
- Observer là một mẫu thiết kế dành cho việc một đối tượng khi thay đổi trạng thái của bản thân nó thì các đối tượng đính kèm theo cũng sẽ được thông báo.
- Mẫu Template Method định nghĩa một bộ khung của một thuật toán trong một chức năng chuyển giao việc thực hiện nó cho các lớp con.
- Strategy định nghĩa một họ các giải thuật khác nhau, mỗi giải thuật được triển khai bởi một lớp cụ thể và chúng có thể hoán đổi cho nhau tùy vào ngữ cảnh.