Mẫu thiết kế - 1
(Design Patterns)

Nội dung

- Giới thiệu về Design Patterns
- Singleton
- Prototype
- Factory method
- Composite
- Proxy

Tổng quan

- MTK là những giải pháp đã được sử dụng trong các phần mềm khác nhau và xem là tốt (best practice): thiết kế tốt nhất, *i.e., số lớp, quan hệ giữa các lớp… khả năng tương tác, thay đổi, mở rộng, tái sử dụng,…*
- Mốc quan trọng: 1995, Gang of Four (GoF) Gamma, Helm, Johnson, và Vlissides; *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley, 1995.

Tổng quan

MTK giải quyết các vấn đề phi chức năng

- Khả năng thay đổi
- Khả năng tương tác
- Khả năng tái sử dụng
- Độ tin cậy

Tổng quan

Có 3 loại MTK

- **Khởi tạo (creational):** liên quan đến khởi tạo đối tượng
- **Cấu trúc (structural):** liên quan đến tổ chức lớp và đối tượng
- **Hành vi (behavioral):** liên quan đến việc gán các chức năng cho lớp

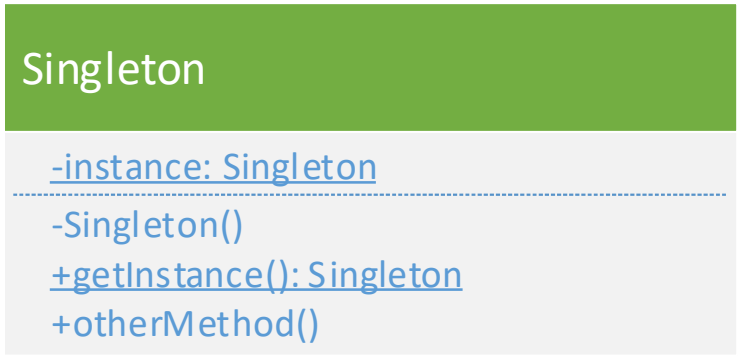
Nội dung chính của một MTK

- Tên: tên của MTK, thường có nghĩa để người dùng dễ hình dung; Ví dụ: Bridge, Mediator, Flyweight
- Ngữ cảnh: ngữ cảnh để áp dụng MTK, ví dụ
- Vấn đề giải quyết: dự định của MTK, mục tiêu (trong điều kiện ràng buộc)
- Giải pháp: các lớp, đối tượng và mối quan hệ giữa các phần tử được đề xuất
- Kết quả: thảo luận về kết quả mang lại của MTK

Singleton

- Ngữ cảnh: Trong một số ứng dụng, việc chỉ có duy nhất một đối tượng (của một lớp đặc biệt nào đó) được tạo ra là rất quan trọng. Ví dụ: kết nối DB, Window manager, file system,…
- Vấn đề: Làm thế nào để chúng ta có thể đảm bảo chỉ duy nhất 1 đối tượng thuộc một lớp nào đó được tạo ra?
- Giải pháp: tạo một lớp với phương thức khởi tạo private, phương thức getInstance() được sử dụng để tạo đối tượng; trong lần gọi đầu tiên, phương thức này sẽ tạo ra một đối tượng; trong những lần gọi tiếp theo, đối tượng đó sẽ được trả về cho client (không có đối tượng mới được tạo ra)

Singleton

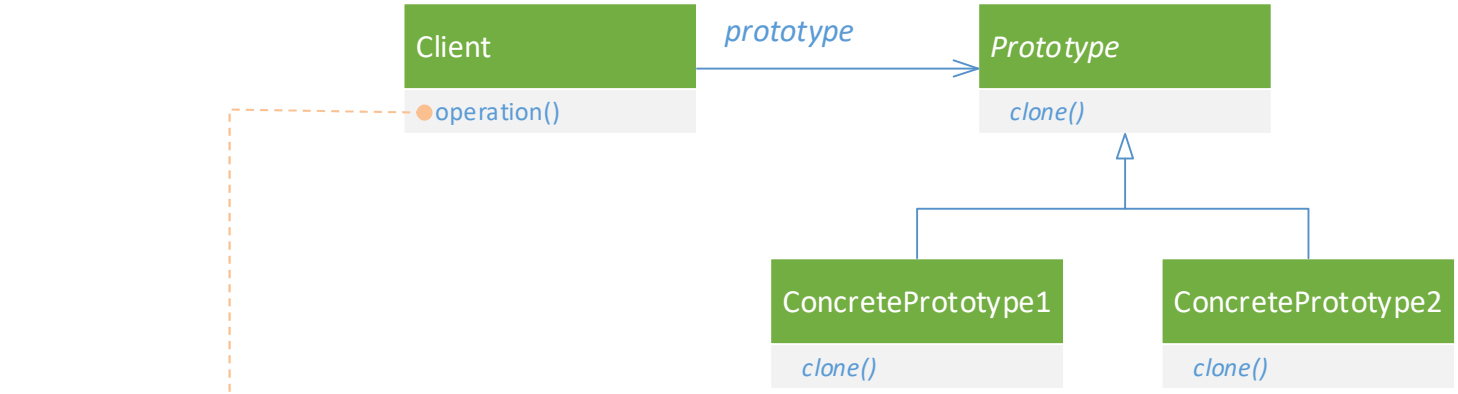



```
class Singleton {
        private static Singleton instance = null;
        private Singleton( ) {}
        public static Singleton getInstance( ) {
                if (instance == null)
                        instance = new Singleton();
                return instance;
        }
}
```

Singleton

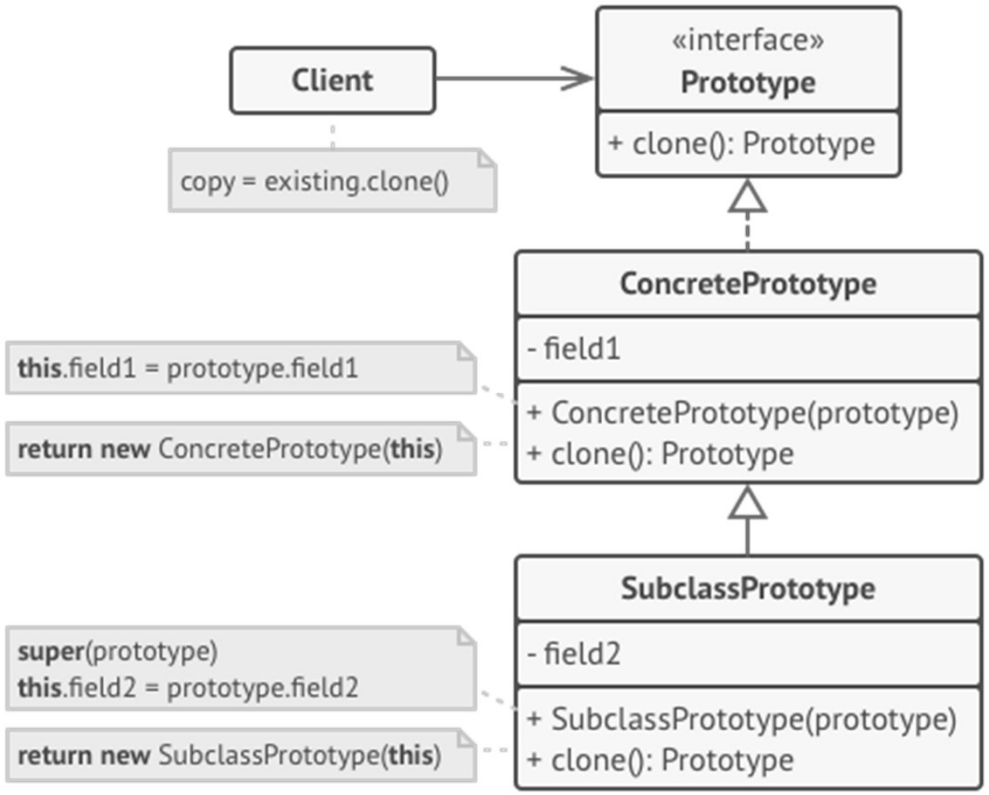
- Thảo luận
  - Mở rộng thành n đối tượng (thay vì một)
  - Nếu một lớp A kế thừa lớp Singleton
  - Dùng thuộc tính và phương thức lớp (static) thay vì dùng Singleton?
  - Vấn đề đa luồng có thể được giải quyết như thế nào?

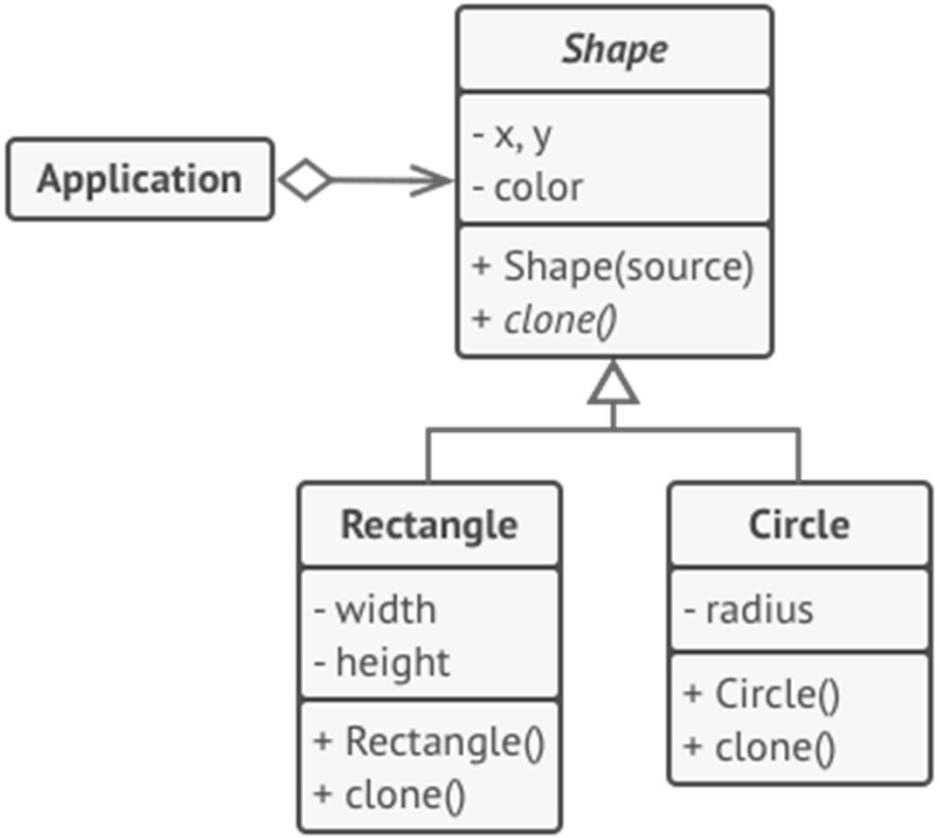
Prototype




```
//operation()
p=prototype.clone();
```

Prototype – Case Study



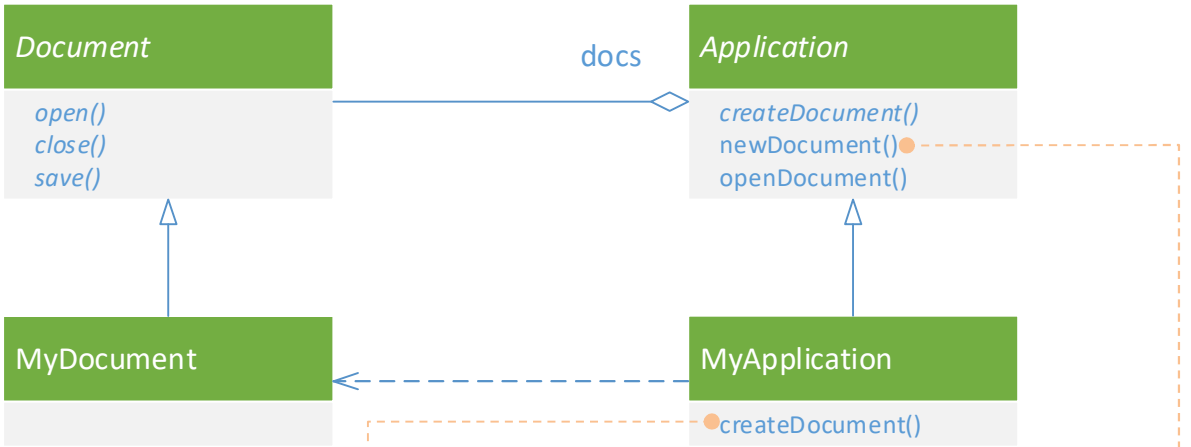


Factory Method

- Ngữ cảnh: Một framework cho ứng dụng đa cửa sổ. Hai phần tử chính trong ứng dụng này là Application và Document (đều là trừu tượng). Người dùng framework phải kế thừa hai lớp này để viết ứng dụng. Application tạo và quản lý Document. Vì Document được kế thừa về sau nên Application không biết được lớp con của Document. Thách thức: Application tạo và quản lý lớp con (chưa biết) của Document

- Phương án: định nghĩa interface để tạo đối tượng nhưng để cho lớp con xác định lớp nào sẽ được sử dụng (để tạo đối tượng)

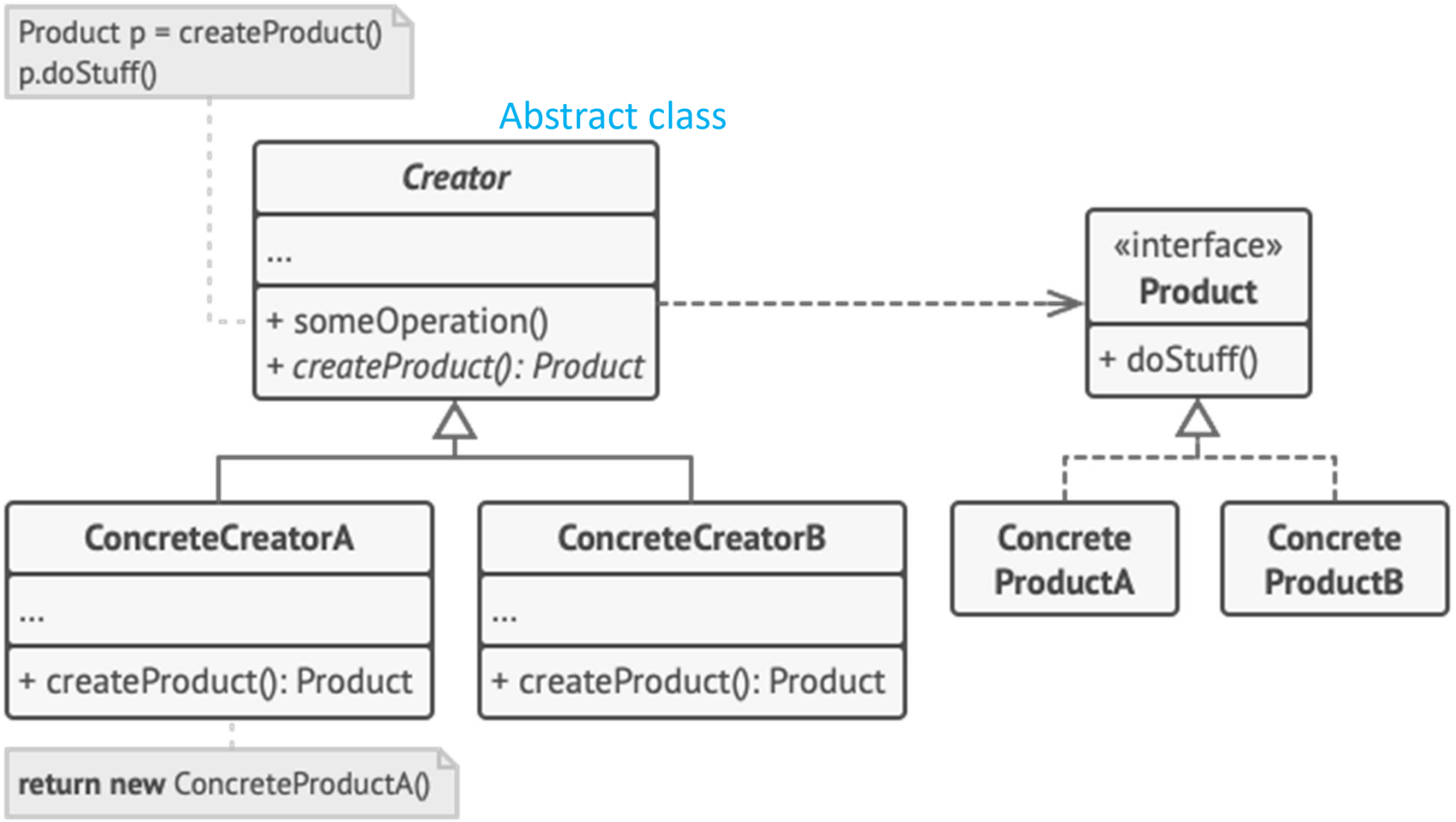
Factory Method



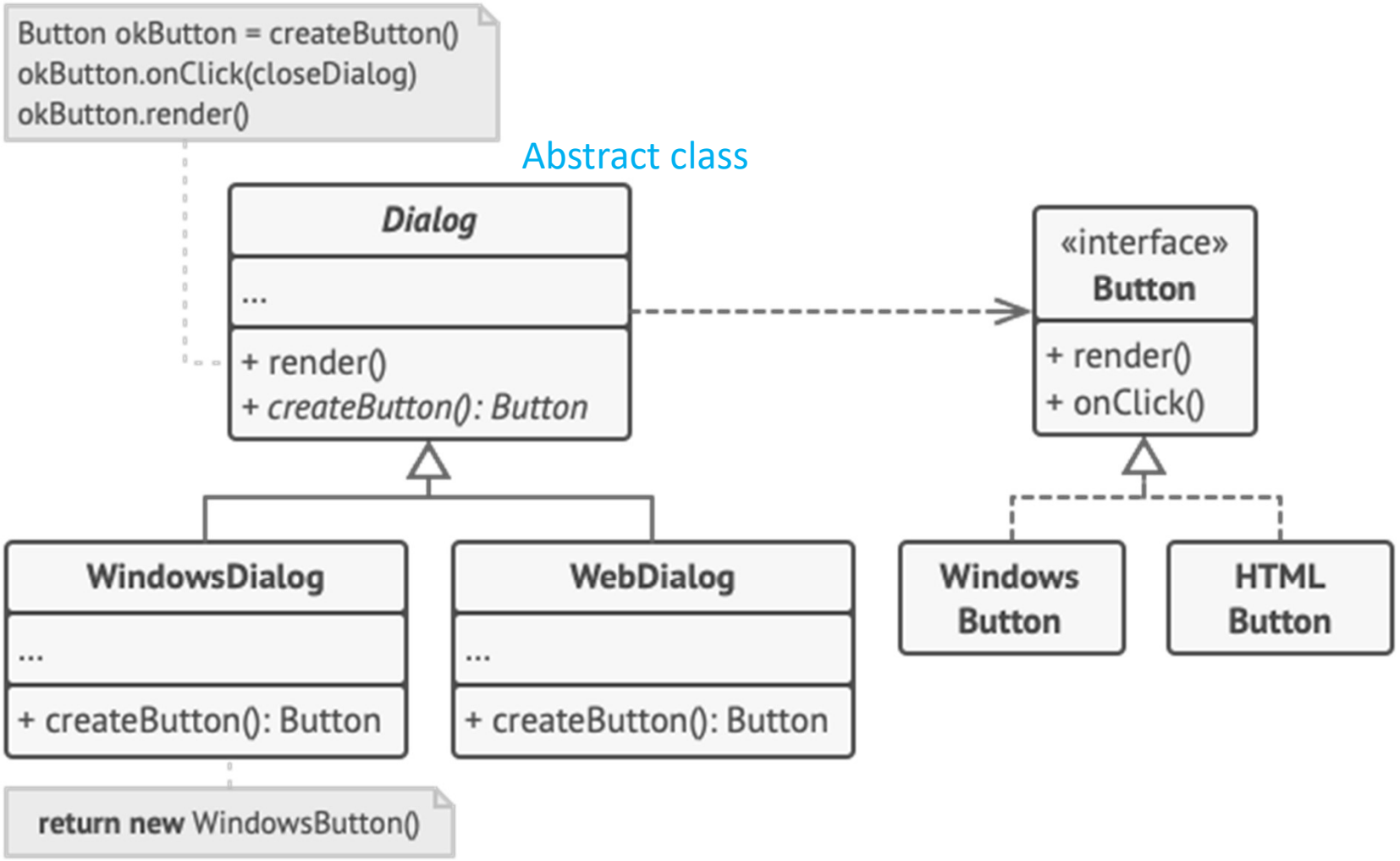

```
//createDocument()
return new MyDocument();
```

```
//newDocument()
Document doc=createDocument();
docs.add(doc);
doc.open();
```

Factory Method



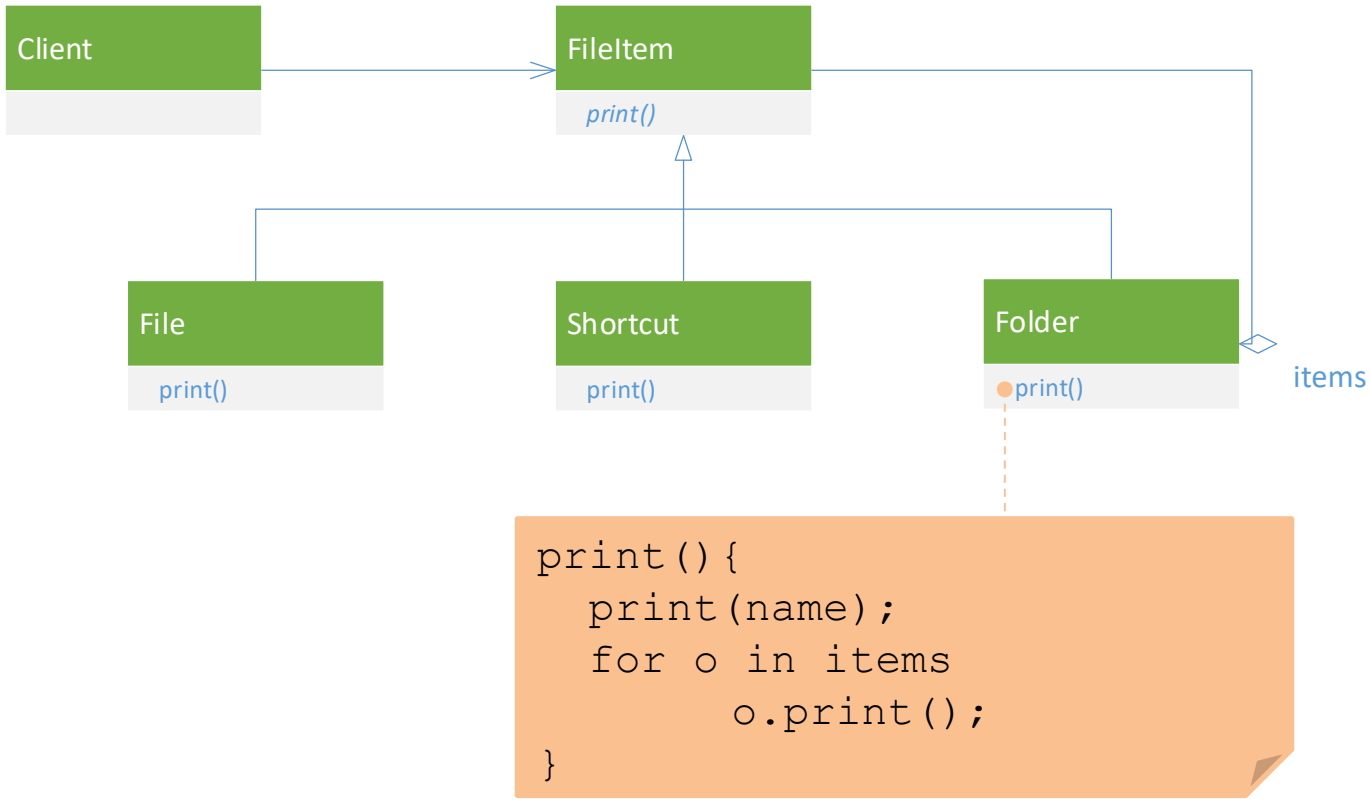
Factory method – Case study



Composite

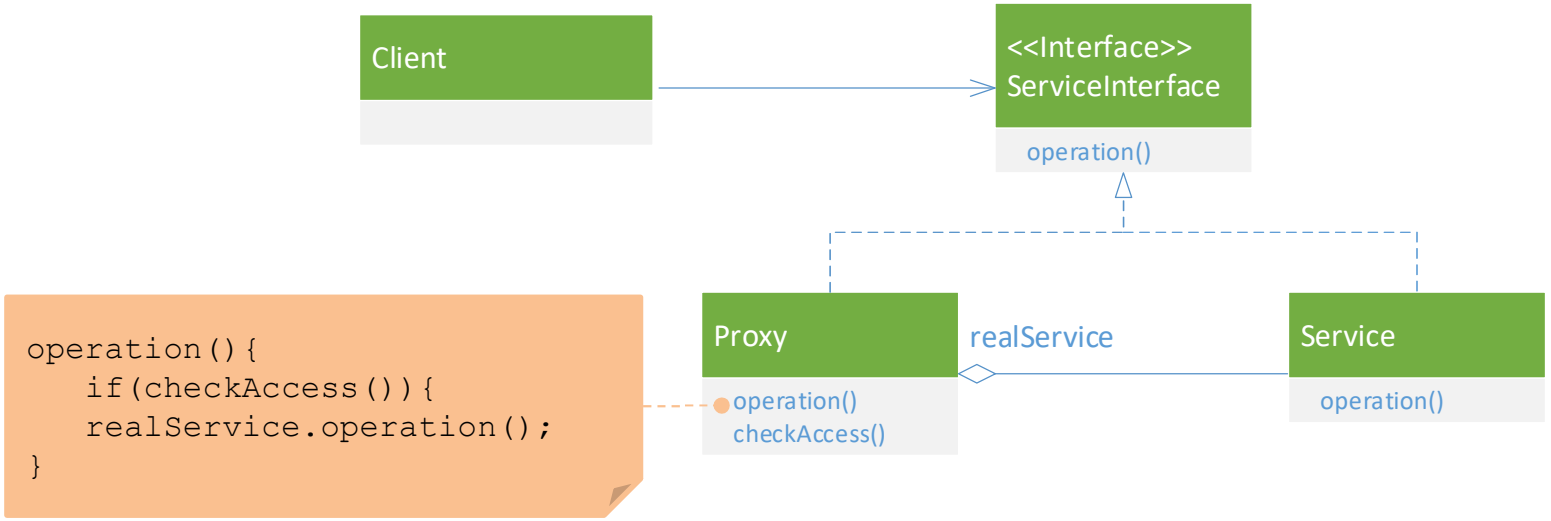
- Cần viết một công cụ quản lý hệ thống file. Các thành phần chính: file, shortcut, và folder. Folder có thể chứa folder, file, shortcut khác.
- Duyệt:
  - Duyệt file: in tên file, kích thước
  - Duyệt shortcut: in đường dẫn đến phần tử đích (phần tử mà shortcut làm đại diện)
  - Duyệt folder: in tên folder và duyệt tiếp nội dung bên trong folder

Composite

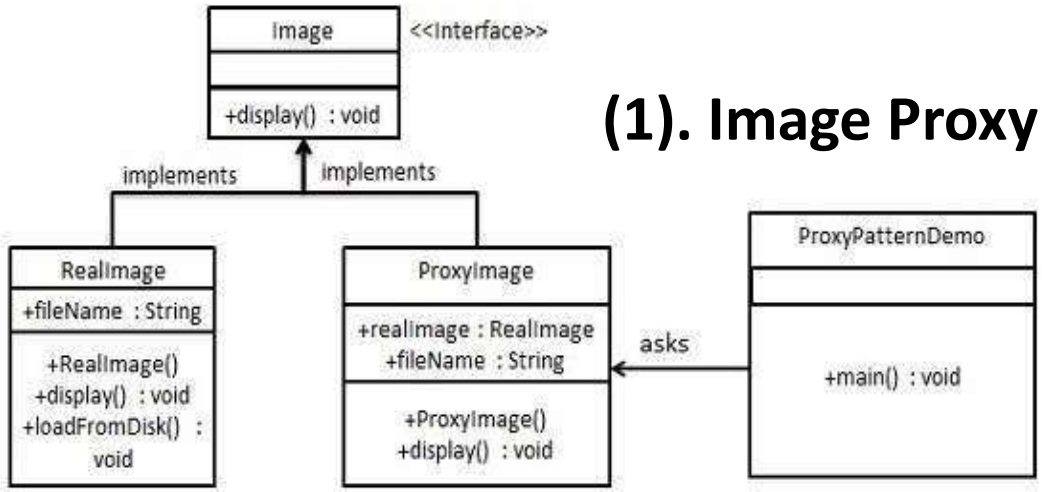


→ ***Recursive structures***

Proxy

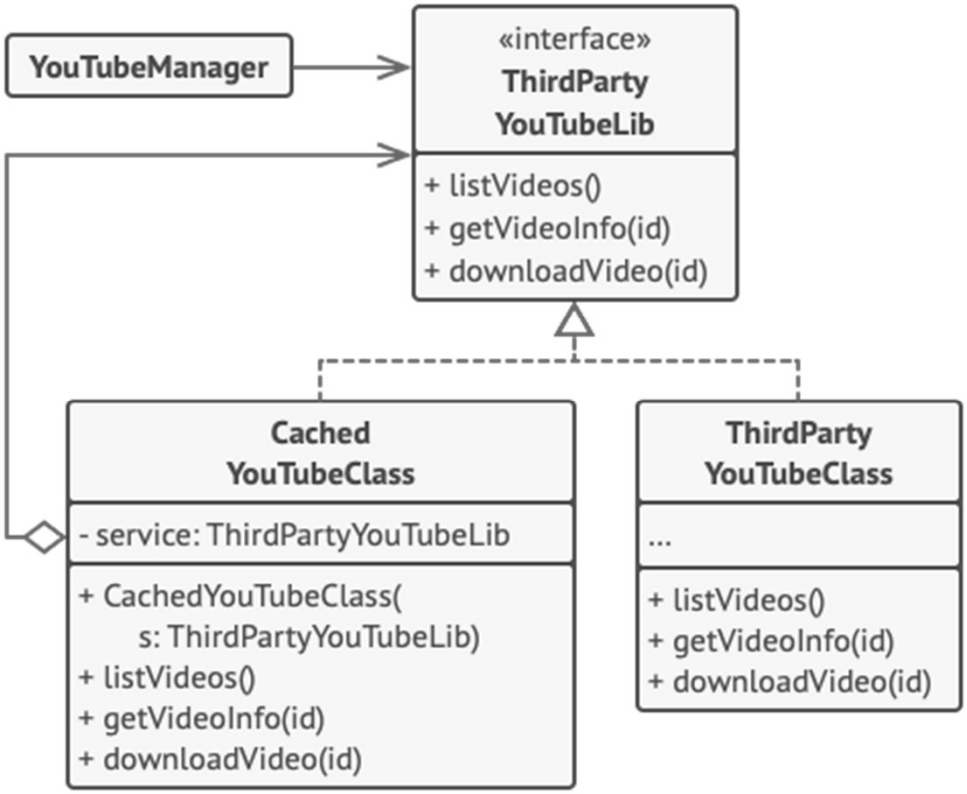

Client
<<Interface>>
ServiceInterface
operation()
operation(){
if(checkAccess()){
realService.operation();
}
Proxy
operation()
checkAccess()
realService
Service
operation()

Proxy patterns – case study



(1). Image Proxy

(2). Youtube cache



Questions

Câu 1. Biểu thức số học chứa các biến X, Y, Z và các phép toán +, -, *, /. Hãy áp dụng mẫu thiết kế nào đó (design patterns) để xây dựng Lớp Expression. Vẽ sơ đồ thiết kế lớp Expression và các lớp liên quan khi tạo thiết kế để xây dựng lớp Expression và tính giá trị một Expression với các giá trị cụ thể.

Câu 2. Một công thức lôgic mệnh đề bao gồm các phép toán AND và OR trên các biến lôgic mệnh đề VAR (A_1, A_2, B_1, B_2…), trên các hằng số TRUE (hoặc ký hiệu là 1), FALSE (hoặc ký hiệu là 0), và trên các công thức lôgic mệnh đề con khác.

a. Hãy xây dựng mô hình thiết kế lớp cho một công thức lôgic mệnh đề sử dụng mẫu thiết kế (có chỉ ra các thuộc tính cần thiết của các lớp nếu có)?
b. Hãy xây dựng mô hình thiết kế lớp cho công thức CNF (Conjunctive Normal Form). Công thức CNF được biểu diễn dưới dạng phép toán AND của các mệnh đề chỉ có phép toán OR trên các biến lôgic mệnh đề, thí dụ $F = (A_1 \vee A_2 \vee A_3 \vee \ldots) \wedge (B_1 \vee B_2 \vee \ldots) \wedge (\ldots)$