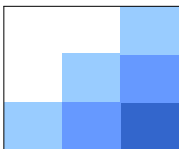




# ***Design patterns***

**Giảng viên: Huỳnh Tuấn Anh**  
**Khoa CNTT - Đại học Nha Trang**



**9/12/2021**

- ❖ Tổng quan về design patterns
- ❖ Nhóm: Creational patterns
- ❖ Nhóm Structural patterns
- ❖ Nhóm Behavioral patterns

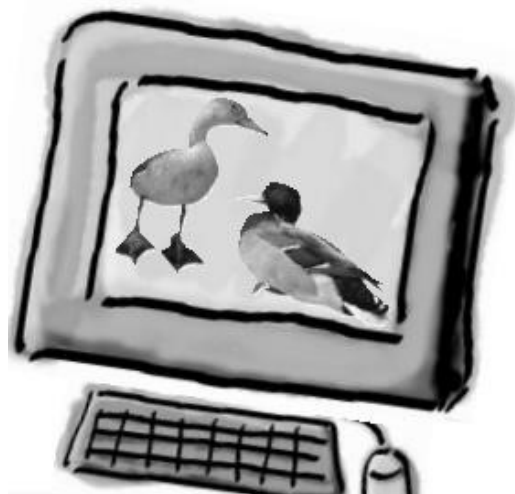
# Tổng quan về design patterns

9/12/2021

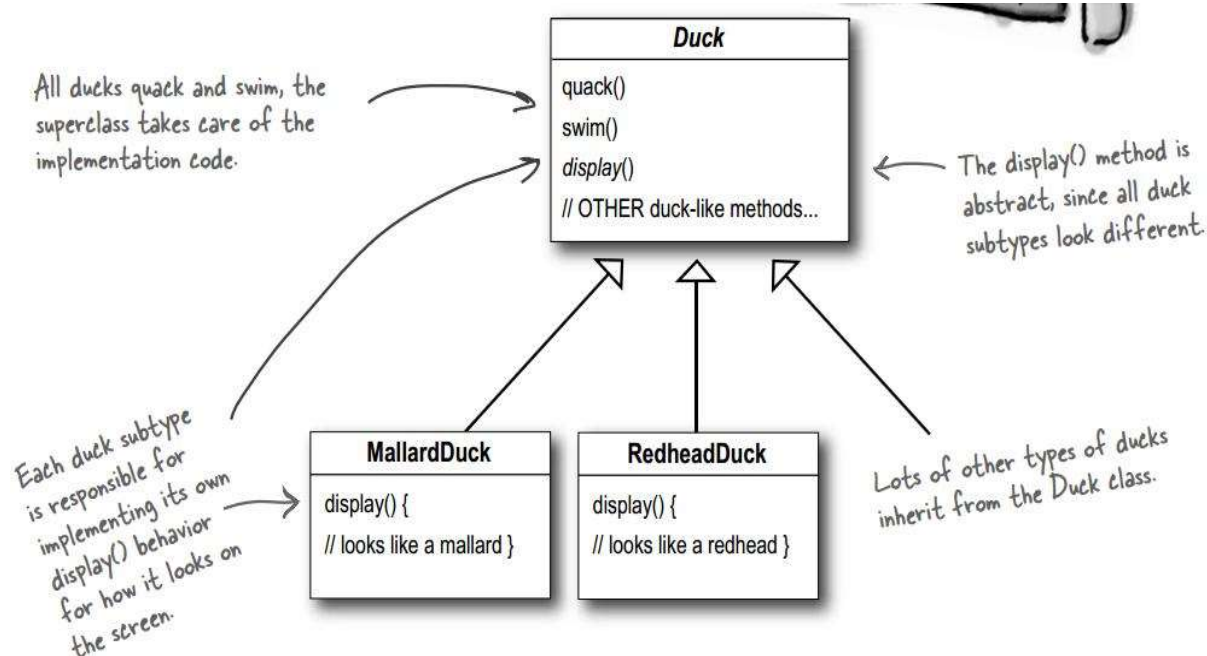
## Vấn đề mở đầu:

### ❖ Game: SimUDuck

- Trò chơi có các nhân vật là các con vịt có khả năng bơi và kêu “quack quack”



## Thiết kế trò chơi bằng phương pháp hướng đối tượng

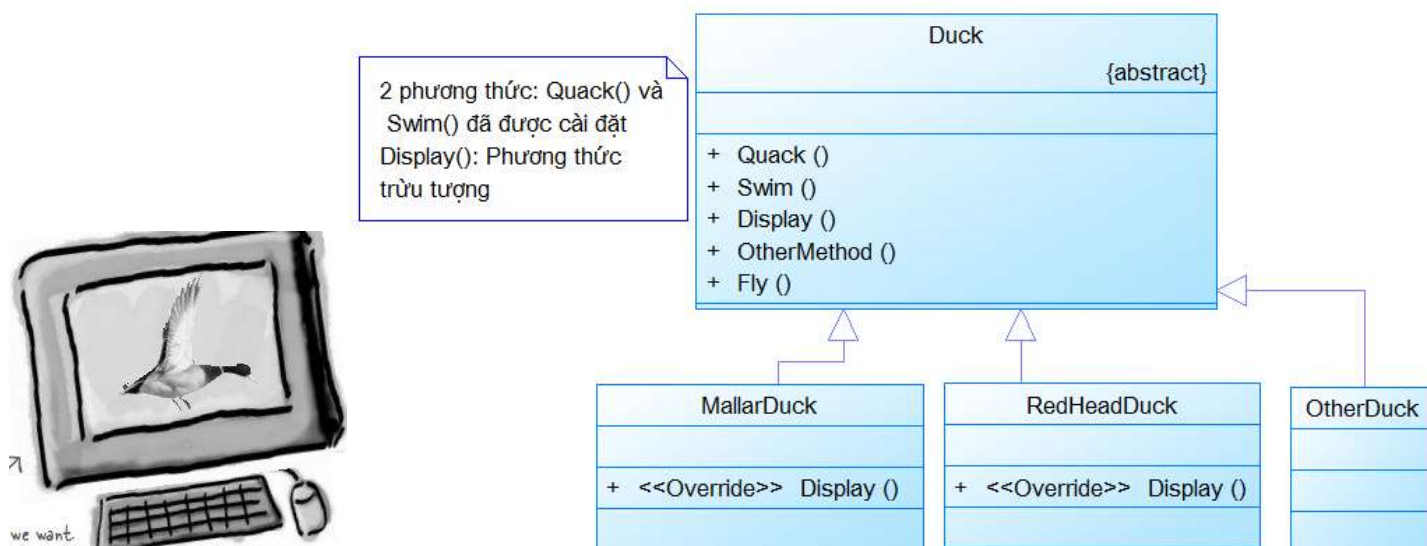


Huỳnh Tuấn Anh - ĐHNT

5

## Vấn đề mở đầu:

Vấn đề: Mở rộng trò chơi để cạnh tranh:  
Vịt có thể biết bay!!!

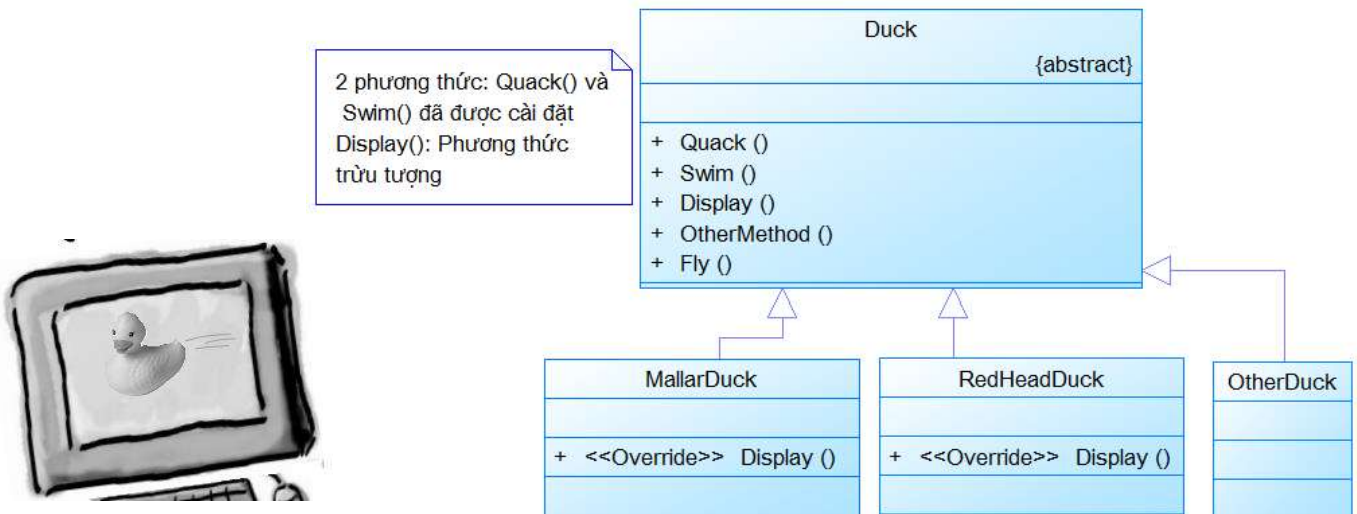


Huỳnh Tuấn Anh - ĐHNT

6

## Các ý tưởng giải quyết vấn đề

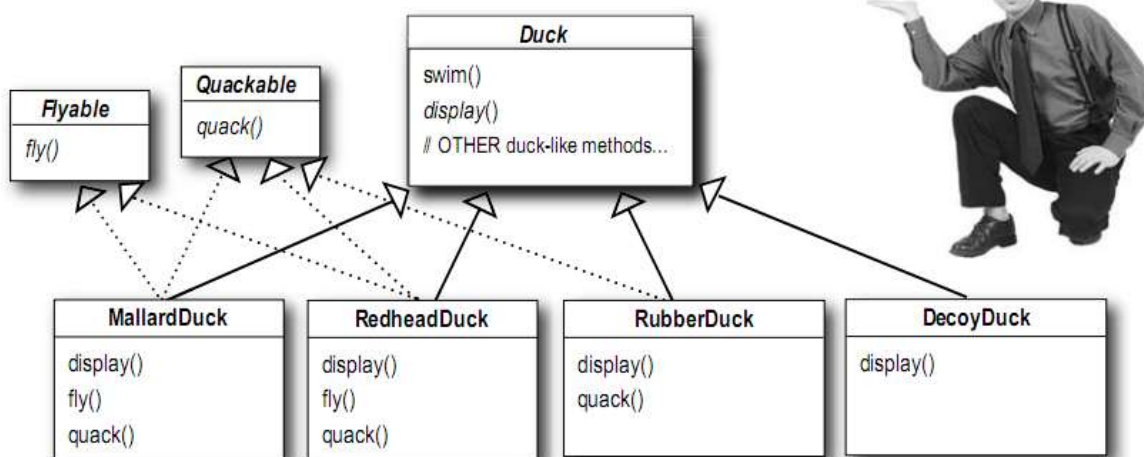
❖ 1<sup>st</sup> idea: Cài đặt thêm phương thức Fly() vào lớp Duck



OtherDuck=Rubber Duck ???

## Các ý tưởng giải quyết vấn đề

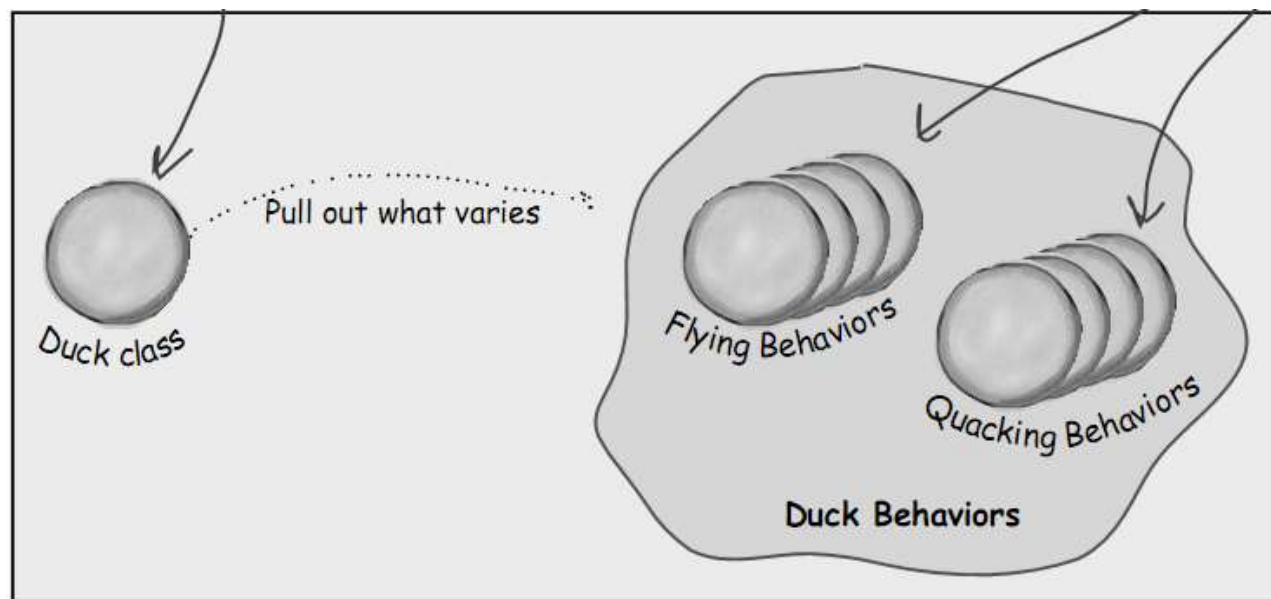
❖ 2<sup>nd</sup> idea: interface?



Nếu có nhiều loại vịt bay hoặc kêu giống nhau???

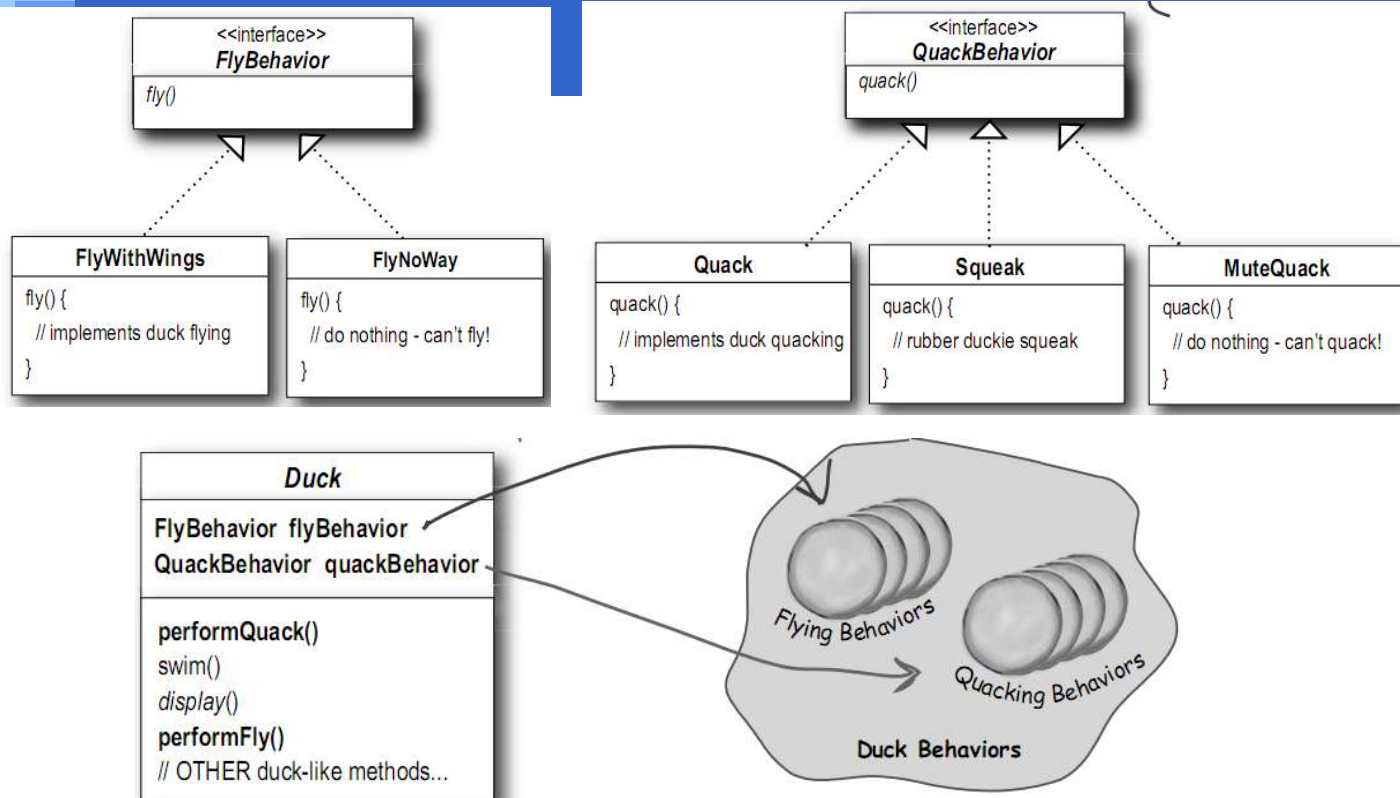
## Các ý tưởng giải quyết vấn đề

❖ 3<sup>rd</sup> idea: Tách rời Duck và các hành vi của chúng!



Huỳnh Tuấn Anh - ĐHNT

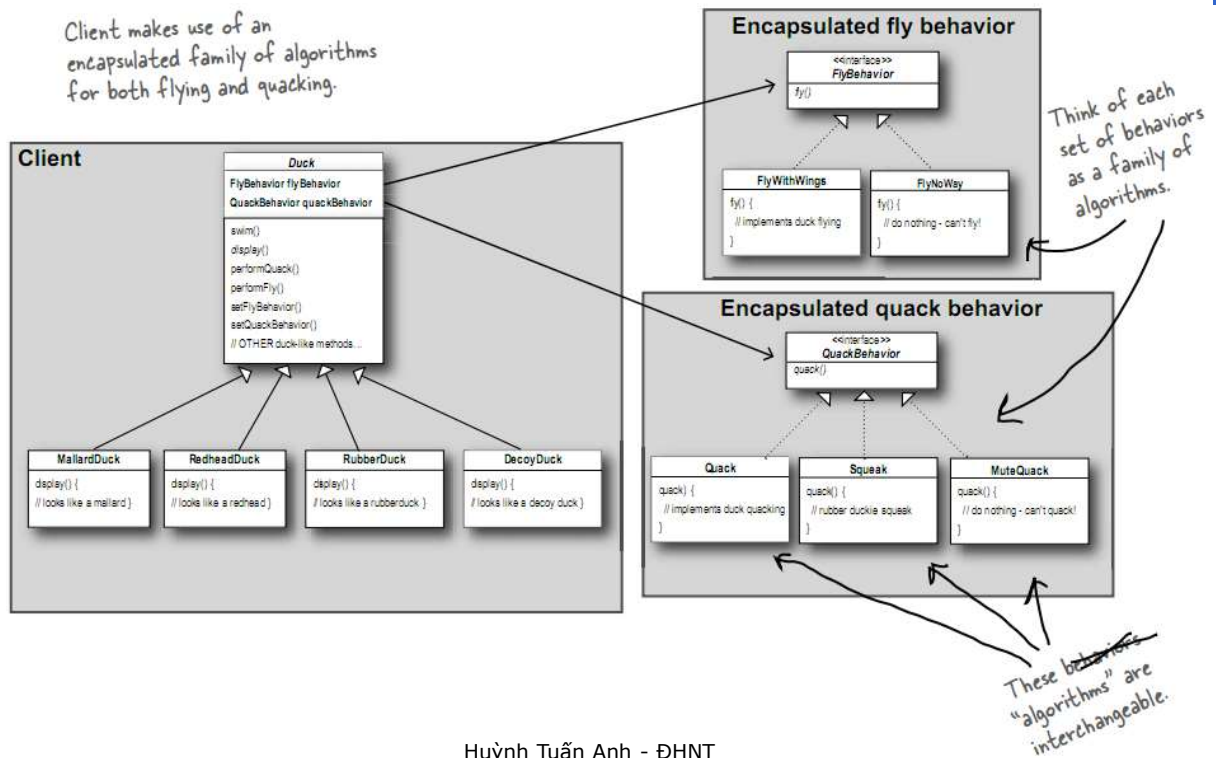
9



Huỳnh Tuấn Anh - ĐHNT

10

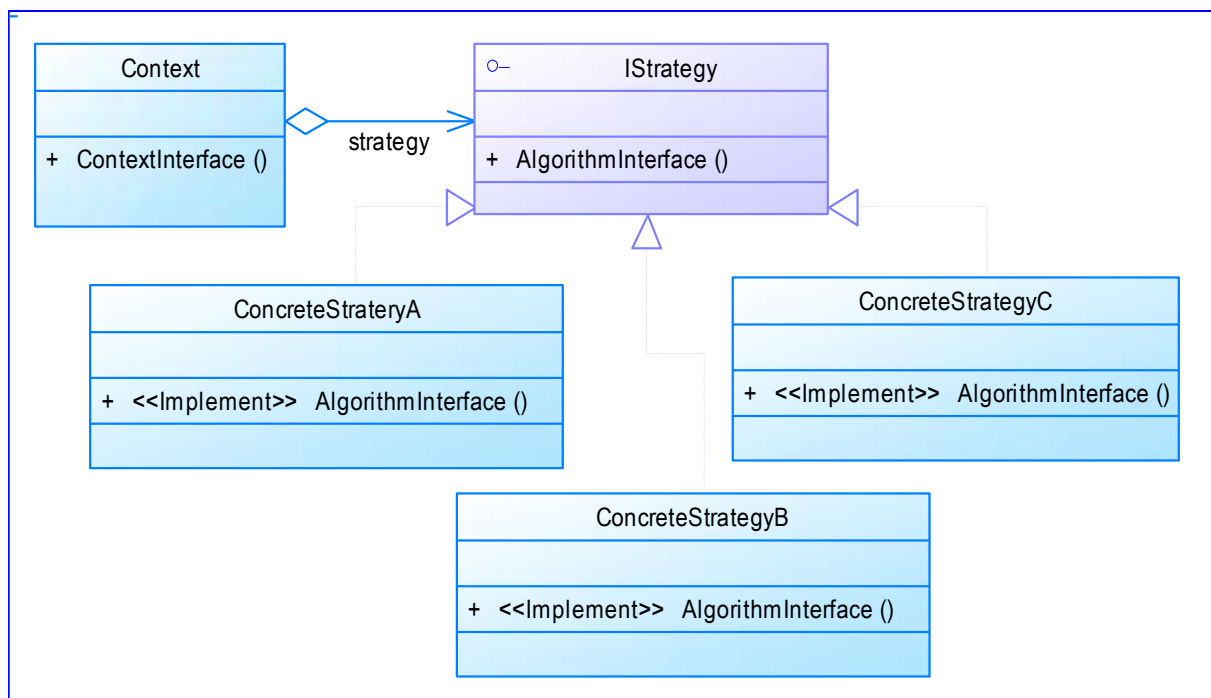
# Strategy pattern



Huỳnh Tuấn Anh - ĐHNT

11

# Strategy pattern



Huỳnh Tuấn Anh - ĐHNT

12

## Mục đích

- ❖ Định nghĩa một họ các thuật toán, đóng gói mỗi thuật toán và làm cho chúng có thể hoán đổi nhau.
- ❖ Strategy cho phép thuật toán thay đổi không phụ thuộc vào client sử dụng nó.

## Nguyên tắc đầu tiên trong thiết kế

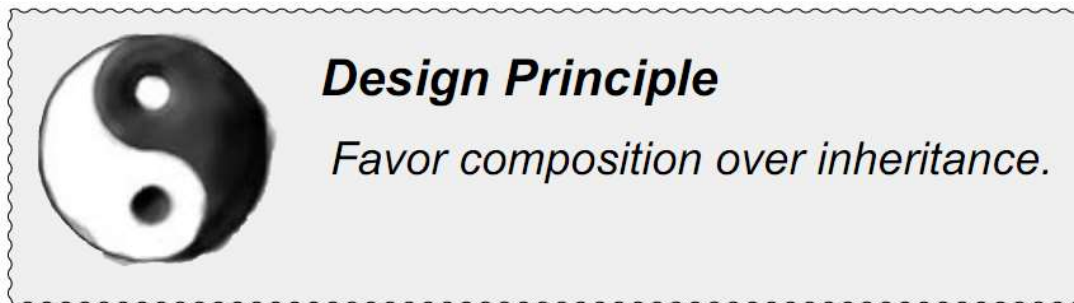


### ***Design Principle***

*Identify the aspects of your application that vary and separate them from what stays the same.*



- ❖ HAS-A can be better than IS-A



## Khái niệm design patterns

- ❖ Bắt nguồn từ thuật ngữ ngành xây dựng của Christopher Alexander, dùng để :
  - Mô tả các vấn đề thường xuyên gặp phải, lặp đi lặp lại trong quá trình xây dựng các công trình.
- ❖ Đưa ra một giải pháp cơ bản để giải quyết các vấn đề đó.
  - Áp dụng giải pháp đó hàng ngàn lần trong xây dựng mà không lần nào giống lần nào !
- ❖ Định nghĩa của GoF : Gồm những mô tả cho việc giao tiếp giữa các đối tượng và lớp đối tượng mà sau đó được tùy biến, chỉnh sửa để giải quyết một vấn đề thiết kế chung chung trong một ngữ cảnh cụ thể.



## Khái niệm design patterns

### ❖ Định nghĩa của Ian Graham (OMG) :

- Là một quá trình xây dựng các kiến trúc thiết kế nhỏ có thể dùng lại.
- Các kiến trúc này cung cấp một giải pháp cơ bản, được dùng đi dùng lại trong các ngữ cảnh khác nhau nhưng đều tuân theo một “luật” nào đó.
- Các Pattern là trừu tượng và không thể dùng ngay được (unplugable), việc cài đặt, thực thi chúng sẽ thể hiện khác nhau với mỗi ứng dụng

## Khái niệm design patterns

### ❖ Dùng để xây dựng các phần mềm :

- Có thể sử dụng lại được.
- Tính linh động cao.
- Chuyên biệt hóa công việc các thành viên.
- Dễ kiểm thử, bảo trì, quản lý.

### ❖ Nhanh chóng tiếp cận và sử dụng lại hiệu quả các thành phần phần mềm được xây dựng với kỹ thuật Design Patterns :

- Framework : Struts, MVC, JSF, các Portal...
- Toolkit : JFC, BDK...
- Các thành phần điều khiển miễn phí

## Khái niệm design patterns

### ❖ Bản chất:

- Mục đích là chỉ ra giải pháp để tạo ra các thành phần dùng lại được và dễ dàng bảo trì, thậm chí là cho cả tài liệu của thành phần đó.
- Các phần mềm được tạo nhờ kỹ thuật Design Patterns phải được phân tách độc lập với nhau trên một số phương diện. Việc phân tách độc lập phần nào sẽ áp dụng Pattern phù hợp cho phần đó.
- Như vậy, việc sử dụng lại như thế nào chính là bản chất của kỹ thuật Design Patterns.

## Nguyên tắc thiết kế

- ❖ Design Principle: “Depend upon abstractions. Do not depend upon concrete classes.”

→ Program to an interface, not an implementation

## Phân loại các mẫu

- ❖ “Gang of Four” đã đưa ra 23 Pattern cơ bản. 23 Pattern này được phân loại theo mục đích và phạm vi áp dụng.
- ❖ Sau này các hãng phần mềm (Microsoft, Sun...) và các cá nhân đã “tạo ra” các Pattern mới, mỗi hãng lại có một “phong cách” khác nhau, ảnh hưởng tới ứng dụng của họ.
- ❖ Các Pattern mới có thể là sự kết hợp của một vài các Pattern cơ bản, các Pattern có được thừa nhận hay không là tùy thuộc và tính linh động, mở rộng và khả chuyển của nó.

## Phân loại các mẫu

- ❖ Theo mục đích :
  - Creational : Áp dụng trong quá trình khởi tạo, độc lập việc khai báo, khởi tạo một đối tượng mới.
  - Structural : Áp dụng trong việc tạo cấu trúc giữa các lớp hay đối tượng, mối liên quan giữa chúng.
  - Behavioral : Áp dụng trong việc độc lập các hành vi đối tượng, các thuật toán thực hiện hành vi đó.
- ❖ Theo phạm vi áp dụng :
  - Cho phạm vi đối tượng (objects).
  - Cho phạm vi lớp đối tượng (classes).

		Mục đích		
		Creational	Structural	Behavioral
Phạm vi áp dụng	Class	-Factory method	-Adapter	-Interpreter -Template method
	Object	-Abstract Factory -Builder -Prototype -Singleton	-Adapter -Bridge -Composite -Decorator -Facade -Proxy	-Chain of Responsibility -Command -Iterator -Mediator -Memento -Flyweight -Observer -State -Strategy -Visitor

Huỳnh Tuấn Anh - ĐHNT

23

## Phân loại mẫu

- ❖ Khó có thể tìm được ứng dụng chỉ áp dụng một Pattern và cũng không thể tìm được một ứng dụng dùng cả 23 Pattern.
- ❖ Một vài Pattern thường đi kèm với nhau như Prototype và Singleton.
- ❖ Một vài Pattern khác là thay thế lẫn nhau, chúng không thể dùng đồng thời như Prototype và Abstract Factory.
- ❖ Mục đích việc phân loại các Pattern:
  - Làm cho việc tìm hiểu và sử dụng từng Pattern nhanh hơn, đúng hơn.
  - Giúp tăng tốc việc tìm kiếm Pattern thích hợp (phù hợp với vấn đề cụ thể của ứng dụng đang phát triển)

Huỳnh Tuấn Anh - ĐHNT

24



## Tài liệu tham khảo

- ❖ Eric Freeman, Elisabeth Freeman. Head First Design pattern. O'Reilly 2006.
- ❖ Bài giảng Design pattern của PGS.TS Huỳnh Quyết Thắng, Đại học BKHN
- ❖ **Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.** Design Patterns Elements of Reusable Object-Oriented Software. Addison-Wesley 1995