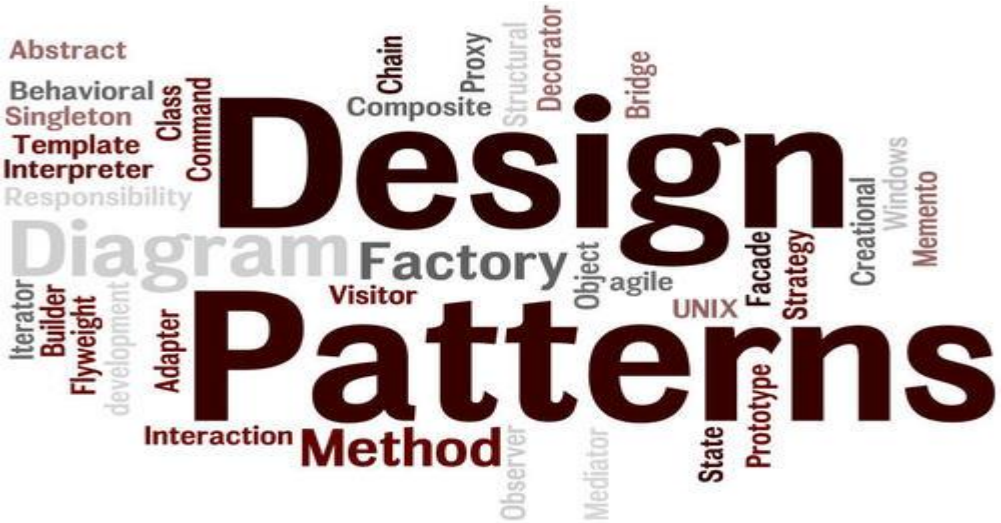


Mẫu thiết kế (Design Patterns)

ThS. Trần Anh Dũng

dungta@uit.edu.vn

Nội dung

- Thông tin chung
- Mục tiêu môn học
- Giới thiệu
- Tổng quan về mẫu thiết kế GoF
- Cấu trúc chung để mô tả mỗi mẫu GoF
- Danh sách 23 mẫu GoF

Giới thiệu chung

- Mẫu thiết kế - SE401
  - Số tín chỉ: 3 (45 tiết)
- Thông tin giảng viên:
  - ThS. Trần Anh Dũng
  - Email: dungta@uit.edu.vn
- Website môn học:
  - https://courses.uit.edu.vn/course/view.php?id=8136

Hình thức kiểm tra & đánh giá

- Seminar + bài tập : 50%
- Thi vấn đáp cuối kỳ : 50%

Mục tiêu môn học

- Môn học trang bị cho sinh viên các kiến thức về phát triển phần mềm chuyên sâu dựa trên các mẫu thiết kế.
- Qua đó, sinh viên sẽ tìm hiểu và nắm được các mẫu thiết kế kinh điển và các trường hợp áp dụng chúng trong thực tế.

Giới thiệu

- Thiết kế là một giai đoạn quan trọng trong qui trình phát triển phần mềm.
- Các lỗi thiết kế thường phải trả giá cao do ảnh hưởng đến nhiều giai đoạn sau (cài đặt, kiểm tra).
- Phương pháp LTHĐT cung cấp cơ chế để có thể xây dựng được phần mềm dễ nâng cấp, dễ thay đổi→ Phụ thuộc nhiều vào khả năng của người thiết kế.
- Mục tiêu của thiết kế: Không chỉ thiết kế những phần mềm đúng mà còn có thể hỗ trợ tái thiết kế trong tương lai.

Giới thiệu

- Chỉ dùng các phương pháp phân tích thiết kế vẫn chưa đủ.
- "Design patterns" [Gam95], một cách tiếp cận dựa mẫu (pattern-based approach) có tác dụng hỗ trợ cho pha thiết kế phần mềm.
- Liên hệ đến các cách tiếp cận tương tự: "analysis patterns", "design heuristics", "process patterns",...

Giới thiệu

- Xu hướng sử dụng mẫu trong công việc chuyên môn, trong học tập:
  - Các mẫu, khuôn được dùng trong các ngành công nghiệp khác nhau (in ấn, đúc,...)
  - Các bài tập mẫu cho học sinh, sinh viên
  - Các mẫu chương trình
  - Các mẫu hướng dẫn thiết kế giao diện với người dùng

Giới thiệu

- Thuật ngữ:
  - Các từ: sample, pattern, template,...
  - "Design patterns": Patterns in Object-oriented software design
  - **"a design pattern"**: an elegant solution to a specific problem in OO software design
  - Thuật ngữ tiếng Việt...

Giới thiệu

- Design pattern?
- Một biện pháp được đề xuất để có những bản thiết kế tốt: Sử dụng lại những mẫu thiết kế của những chuyên gia đã qua kiểm nghiệm thực tế.
- Mẫu thiết kế thường có đặc điểm:
  - Là những thiết kế đã được sử dụng và được đánh giá tốt.
  - Giúp giải quyết những vấn đề thiết kế thường gặp.
  - Chú trọng việc giúp cho bản thiết kế có tính uyển chuyển, dễ nâng cấp, thay đổi.

Giới thiệu

- Vai trò của mẫu thiết kế:
  - Cung cấp phương pháp giải quyết những vấn đề thực tế thường gặp đã được đánh giá, kiểm nghiệm.
  - Là biện pháp tái sử dụng tri thức các chuyên gia phần mềm.
  - Hình thành kho tri thức, ngữ vựng trong giao tiếp giữa những người làm phần mềm.
  - Giúp tìm hiểu, nắm vững hơn đặc điểm ngôn ngữ lập trình, nhất là lập trình hướng đối tượng.

→Tăng độ tin cậy, tiết kiệm nguồn lực,…

Giới thiệu

- Định nghĩa một mẫu (pattern) nói chung:
  - Một mẫu là một cặp (vấn đề, lời giải) có thể áp dụng trong nhiều tình huống, ngữ cảnh khác nhau. Mỗi mẫu thường bao gồm các bộ phận sau:
    - Tên
    - Nội dung vấn đề
    - Lời giải (phải đủ tổng quát để có thể dùng trong nhiều tình huống)
    - Các hệ quả mang lại và ví dụ áp dụng

Tổng quan về mẫu thiết kế GoF

- Nguồn gốc lịch sử và Tác giả
  - Gồm 23 mẫu thiết kế của 4 tác giả: Erich Gamma, Richard Helm, Ralph Johnson, và John Vlissides;
  - Các mẫu này còn được gọi là mẫu GoF (Gang of Four)
  - “Finding patterns is much easier than describing them”

Tổng quan về mẫu thiết kế GoF

- Khoảng ½ của bộ mẫu này có nguồn gốc từ luận án tiến sĩ của Erich Gamma. Các tác giả gặp nhau tại 2 hội nghị OOPSLA'91 và OOPSLA'92 (Object-Oriented Programming Systems, Languages, and Applications Conference). Sau đó cùng làm việc để soạn lại một bộ gồm 23 mẫu và trình bày tại hội nghị ECOOP'93 (European Conference on Object Oriented Programming).

Cấu trúc chung để mô tả mỗi mẫu GoF

- Tên và Phân loại
- Mục đích, ý định: mô tả ngắn gọn về mẫu
- Bí danh
- Motivation: trình bày một tình huống cụ thể trong thiết kế phần mềm dẫn đến việc sử dụng mẫu này để giải quyết vấn đề.

Cấu trúc chung để mô tả mỗi mẫu GoF

- Khả năng ứng dụng: gợi ý các tình huống trong thiết kế mà có thể ứng dụng mẫu này
- Cấu trúc: mô tả mẫu bằng các ký hiệu đồ hình thường dùng trong các phương pháp phân tích, thiết kế (ký hiệu OMT, UML,...)
- Các thành viên: trình bày ý nghĩa của các lớp/đối tượng tham gia vào mẫu thiết kế và trách nhiệm của chúng

Cấu trúc chung để mô tả mỗi mẫu GoF

- Sự cộng tác: các thành viên (lớp/đối tượng) của mẫu cộng tác như thế nào để thực hiện trách nhiệm của chúng
- Các hệ quả mang lại
- Chú ý liên quan đến việc cài đặt
- Mã nguồn minh họa

Cấu trúc chung để mô tả mỗi mẫu GoF

- Nêu ra những ví dụ về các hệ thống thực tế (đã được phát triển và đang chạy) mà có sử dụng mẫu này
- Các mẫu liên quan: những mẫu nào có liên hệ với mẫu này, những điểm quan trọng cần phải phân biệt; mẫu này có thể dùng phối hợp với những mẫu nào.

Danh sách 23 mẫu GoF

CÁC MẪU VỀ TẠO LẬP LỚP hay ĐỐI TƯỢNG	
Tên	**Mục đích**
Abstract Factory (Fabrique Abstraite)	Cung cấp một interface cho việc tạo lập các đối tượng (có liên hệ với nhau) mà không cần qui định lớp khi tạo mỗi đối tượng
Builder (Monter)	Tách rời việc xây dựng (construction) một đối tượng phức tạp khỏi biểu diễn của nó sao cho cùng một tiến trình xây dựng có thể tạo được các biểu diễn khác nhau
Factory Method (Fabrication)	Định nghĩa một interface để tạo lập đối tượng nhưng ủy nhiệm việc instanciation cho các lớp con (các lớp kế thừa)
Prototype	Qui định loại của các đối tượng cần tạo bằng cách dùng một đối tượng mẫu, tạo mới nhờ vào sao chép đối tượng mẫu nầy.
Singleton	Cài đặt lớp mà chỉ có một thể hiện (đối tượng) duy nhất

Danh sách 23 mẫu GoF

CÁC MẪU VỀ CẤU TRÚC LỚP hay ĐỐI TƯỢNG	
Tên	**Mục đích**
Adapter (adapteur)	Do vấn đề tương thích, thay đổi interface của một lớp thành một interface khác phù hợp với yêu cầu người sử dụng lớp.
Bridge (Pont)	Tách rời ngữ nghĩa của một vấn đề khỏi việc cài đặt ; mục đích để cả hai bộ phận (ngữ nghĩa và cài đặt) có thể thay đổi độc lập nhau.
Composite	Tổ chức các đối tượng theo cấu trúc phân cấp dạng cây; Tất cả các đối tượng trong cấu trúc được thao tác theo một cách thuần nhất như nhau.
Decorator (Décorateur)	Gán thêm trách nhiệm cho đối tượng (mở rộng chức năng) vào lúc chạy (dynamically).
Facade (Façade)	Cung cấp một interface thuần nhất cho một tập hợp các interface trong một "hệ thống con" (subsystem).
Flyweight (Poids mouche)	Sử dụng việc chia sẻ để thao tác hiệu quả trên một số lượng lớn đối tượng "cở nhỏ" (chẳng hạn paragraph, dòng, cột, ký tự...)
Proxy (Procuration)	Điều khiển một cách gián tiếp việc truy xuất một đối tượng thông qua một đối tượng được ủy nhiệm.

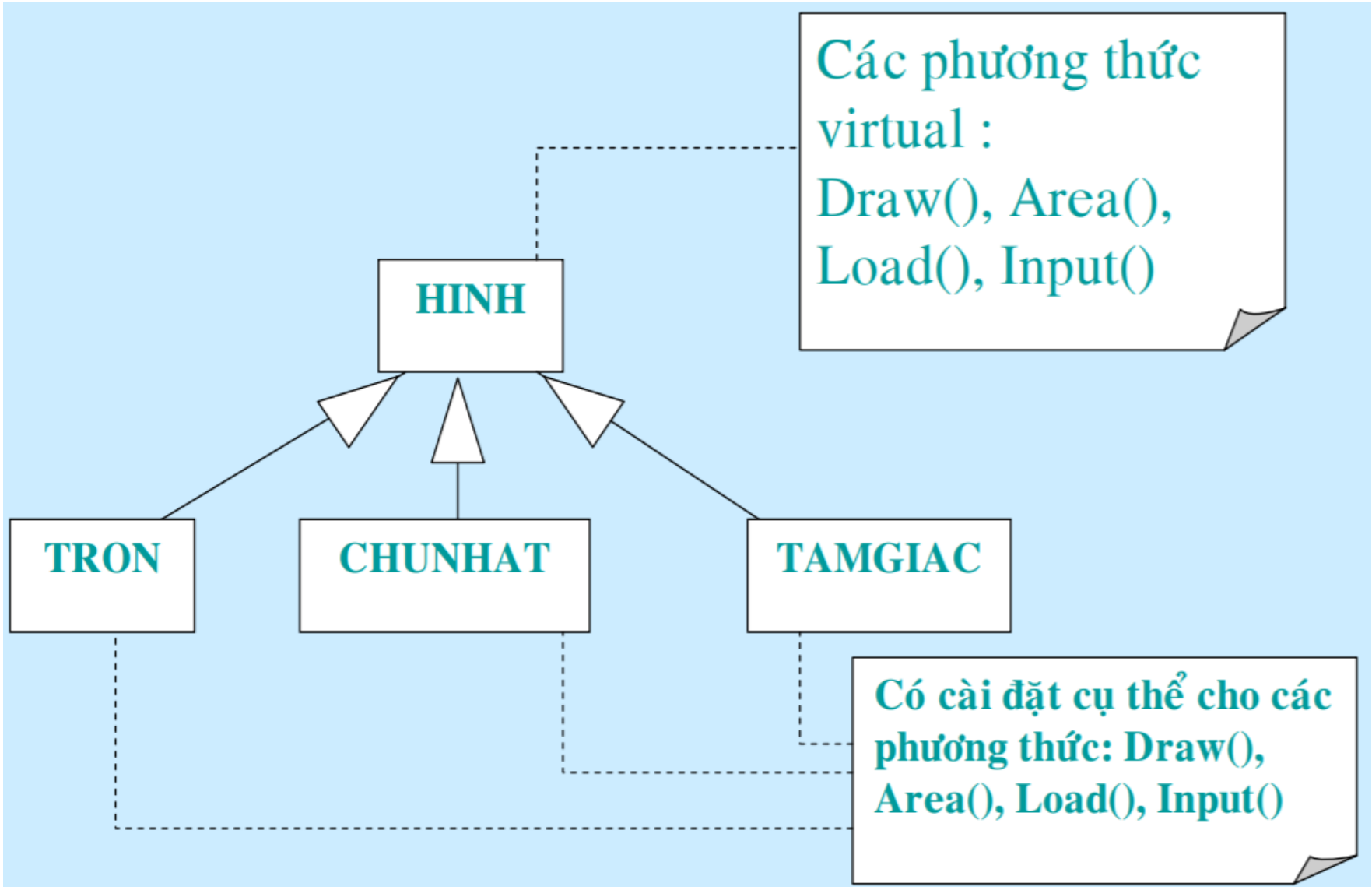
Danh sách 23 mẫu GoF

CÁC MẪU VỀ ỨNG XỬ CỦA LỚP hay ĐỐI TƯỢNG (1)	
Tên	**Mục đích**
Chain of Responsibility (Chaîne de responsabilités)	Khắc phục việc ghép cặp giữa bộ gởi và bộ nhận thông điệp; Các đối tượng nhận thông điệp được kết nối thành một chuỗi và thông điệp được chuyển dọc theo chuỗi nầy đến khi gặp được đối tượng xử lý nó.
Command (Commande)	Mỗi yêu cầu (thực hiện một thao tác nào đó) được bao bọc thành một đối tượng. Các yêu cầu sẽ được lưu trữ và gởi đi như các đối tượng.
Interpreter (Interpreteur)	Hỗ trợ việc định nghĩa biểu diễn văn phạm và bộ thông dịch cho một ngôn ngữ.
Iterator (Itérateur)	Truy xuất các phần tử của đối tượng dạng tập hợp tuần tự (list, array, ...) mà không phụ thuộc vào biểu diễn bên trong của các phần tử.
Mediator (Médiateur)	Định nghĩa một đối tượng để bao bọc việc giao tiếp giữa một số đối tượng với nhau.
Memento	Hiệu chỉnh và trả lại như cũ trạng thái bên trong của đối tượng mà vẫn không vi phạm việc bao bọc dữ liệu.

Danh sách 23 mẫu GoF

CÁC MẪU VỀ ỨNG XỬ CỦA LỚP hay ĐỐI TƯỢNG (2)	
Tên	**Mục đích**
Observer (Observateur)	Định nghĩa sự phụ thuộc *một-nhiều* giữa các đối tượng sao cho khi một đối tượng thay đổi trạng thái thì tất cả các đối tượng phụ thuộc nó cũng thay đổi theo.
State (Etat)	Cho phép thay đổi ứng xử của đối tượng tùy theo sự thay đổi trạng thái bên trong của nó.
Strategy (Stratégie)	Bao bọc một họ các thuật toán bằng các lớp đối tượng để thuật toán có thể thay đổi độc lập đối với chương trình sử dụng thuật toán.
Template method (Patron de méthode)	Định nghĩa phần khung của một thuật toán, tức là một thuật toán tổng quát gọi đến một số phương thức chưa được cài đặt trong lớp cơ sở; việc cài đặt các phương thức được ủy nhiệm cho các lớp kế thừa.
Visitor (Visiteur)	Cho phép định nghĩa thêm phép toán mới tác động lên các phần tử của một cấu trúc đối tượng mà không cần thay đổi các lớp định nghĩa cấu trúc đó.

Kế thừa, Đa xạ: cơ sở cho mẫu GoF



Nguyên lý thiết kế

- Để thiết kế tối ưu và lập trình hiệu quả, chúng ta phải nắm rõ những nguyên lý trong thiết kế hướng đối tượng.
  - The Open-Closed Principle: Nguyên lý mở
  - The Dependency Inversion Principle: Nguyên lý nghịch đảo phụ thuộc
  - The Liskov Substitution Principle: Nguyên lý thay thế Liskov
  - The Interface Segregation: Nguyên lý phân tách interface

Nguyên lý mở

- Phát biểu
  - Các thực thể của phần mềm (class, module,..) nên được mở cho việc mở rộng nhưng nên được đóng với việc thay đổi nó.
- Các thực thể trong một phần mềm không đứng riêng lẻ mà có sự gắn kết chặt chẽ với nhau.
- Để quá trình bảo trì, nâng cấp, mở rộng phần mềm diễn ra dễ dàng và hiệu quả hơn, các thực thể phần mềm nên được xây dựng tuân theo nguyên lý Open-Closed

Nguyên lý nghịch đảo phụ thuộc

- Phát biểu
  - Các thành phần trong phần mềm không nên phụ thuộc vào những cái riêng, cụ thể (details) mà ngược lại nên phụ thuộc vào những cái chung, tổng quát (abstractions) của những cái riêng, cụ thể đó.
- Cái trừu tượng không phụ thuộc vào cái cụ thể bởi cái trừu tượng có tính ổn định hơn.
- Phụ thuộc vào cái trừu tượng thì sẽ giúp cho hệ thống trở nên linh động hơn, thích ứng tốt với những sự thay đổi thường xuyên của cái riêng, cái cụ thể.

Nguyên lý Thay thế Liskov

- Phát biểu
  - Một lớp dẫn xuất phải có thể thay thế cho những lớp cơ sở của chúng.
- Các đối tượng của lớp dẫn xuất hoàn toàn có thể thay thế các đối tượng của lớp cơ sở trong những hàm thao tác trên các đối tượng của lớp cơ sở.
- Một lớp dẫn xuất phải có khả năng cư xử như các đối tượng của lớp cơ sở. Các ràng buộc của lớp cơ sở phải được kế thừa bởi lớp dẫn xuất.

Nguyên lý Phân tách interface

- Phát biểu
  - Không nên buộc các thực thể phần mềm phụ thuộc vào những interface mà chúng không sử dụng đến.
- Những lớp đối tượng có interface bị "ô nhiễm" (fat interface or polluted interface)
  - Trở nên cồng kềnh
  - Làm giảm hiệu năng của phần mềm
  - Tăng sự kết dính giữa các thực thể phần mềm
- Điều này dẫn đến sự dư thừa không cần thiết trong các thực thể phần mềm.

Q & A