**Design Pattern là gì?**

Design patterns là tập các giải pháp cho cho vấn đề phổ biến trong thiết kế các hệ thống máy tính. Đây là tập các giải pháp đã được công nhận là tài liệu có giá trị, những người phát triển có thể áp dụng giải pháp này để giải quyết các vấn đề tương tự.

Giống như với các yêu cầu của thiết kế và phân tích hướng đối tượng (nhằm đạt được khả năng sử dụng các thành phần và thư viện lớp), việc sử dụng các mẫu cũng cần phải đạt được khả năng tái sử dụng các giải pháp chuẩn đối với vấn đề thường xuyên xảy ra.

**Tại sao sử dụng Design Pattern?**

Design pattern cung cấp giải pháp ở dạng tổng quát, giúp tăng tốc độ phát triển phần mềm bằng cách đưa ra các mô hình test, mô hình phát triển đã qua kiểm nghiệm. Thiết kế phần mềm hiệu quả đòi hỏi phải cân nhắc các vấn đề sẽ nảy sinh trong quá trình hiện thực hóa (implementation). Dùng lại các design pattern giúp tránh được các vấn đề tiềm ẩn có thể gây ra những lỗi lớn, dễ dàng nâng cấp, bảo trì về sau.

Một lợi thế lớn để sử dụng một mẫu thiết kế là lập trình viên khác sẽ có thể dễ dàng nhận ra nó (đặc biệt là nếu bạn sử dụng quy ước đặt tên tốt).

**Khi nào nên sử dụng Design pattern?**

Đó là khi bạn muốn giữ cho chương trình của mình thực sự đơn giản. Việc sử dụng các design pattern sẽ giúp chúng ta giảm được thời gian và công sức suy nghĩ ra các cách giải quyết cho những vấn đề đã có lời giải. Bạn có thể đọc qua cuốn “Head First Design Patterns” để có cái nhìn tổng quát hơn về design pattern.

Hệ thống các mẫu design pattern hiện có **23** mẫu được định nghĩa trong cuốn “Design patterns Elements of Reusable Object Oriented Software”. Các tác giả của cuốn sách là Erich Gamma, Richard Helm, Ralph Johnson và John Vlissides, hay còn được biết đến với các tên “Gang of Four” hay đơn giản là “GoF”. Hệ thống các mẫu này có thể nói là đủ và tối ưu cho việc giải quyết hết các vấn đề của bài toán phân tích thiết kế và xây dựng phần mềm trong thời điểm hiện tại. Hệ thống các mẫu design pattern được chia thành 3 nhóm: nhóm Creational (5 mẫu), nhóm Structural (7 mẫu) và nhóm Behavioral (11 mẫu).

|  |  |  |
| --- | --- | --- |
| STT | Tên | Mục đích |
| **Nhóm Creational (nhóm kiến tạo)** | | |
| 1 | Abstract Factory | Cung cấp một interface cho việc tạo lập các đối tượng (có liên hệ với nhau) mà không cần  qui định lớp khi hay xác định lớp cụ thể (concrete) tạo mỗi đối tượng  Tần suất sử dụng:  cao |
| 2 | Builder | 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.  Tần suất sử dụng:  trung bình thấp |
| 3 | Factory Method | Định nghĩa Interface để sinh ra đối tượng nhưng để cho lớp con quyết định lớp  nào được dùng để sinh ra đối tượng Factory method cho phép một lớp chuyển quá trình  khởi tạo đối tượng cho lớp con.  Tần suất sử dụng:  cao |
| 4 | Prototype | Qui định loại của các đối tượng cần tạo bằng cách dùng một đối tượng mẫu, tạo mới n  hờ vào sao chép đối tượng mẫu này.  Tần suất sử dụng: trung bình |
| 5 | Singleton | Đảm bảo 1 class chỉ có 1 instance và cung cấp 1 điểm truy xuất toàn cục đến nó.  Tần suất sử dụng:  cao trung bình |
| **Nhóm Structural (nhóm cấu trúc)** | | |
| 6 | Adapter | Do vấn đề tương thích, thay đổi interface của một lớp thành một interface khác phù hợp  với yêu cầu người sử dụng lớp.  Tần suất sử dụng:  cao trung bình |
| 7 | Bridge | 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.  Tần suất sử dụng:  trung bình |
| 8 | 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.  Tạo quan hệ thứ bậc bao gộp giữa các đối tượng. Client có thể xem đối tượng bao gộp và bị  bao gộp như nhau -> khả năng tổng quát hoá trong code của client -> dễ phát triển, nâng cấp  , bảo trì.  Tần suất sử dụng:  cao trung bình |
| 9 | Decorator | Gán thêm trách nhiệm cho đối tượng (mở rộng chức năng) vào lúc chạy (dynamically).  Tần suất sử dụng:  trung bình |
| 10 | Facade | Cung cấp một interface thuần nhất cho một tập hợp các interface trong một “hệ thống con”  (subsystem). Nó định nghĩa 1 interface cao hơn các interface có sẵn để làm cho hệ thống  con dễ sử dụng hơn.  Tần suất sử dụng:   cao |
| 11 | Flyweight | 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…).  Tần suất sử dụng:  thấp |
| 12 | Proxy | Cung cấp đối tượng đại diện cho một đối tượng khác để hỗ trợ hoặc kiểm soát quá trình truy  xuất đối tượng đó. Đối tượng thay thế gọi là proxy.  Tần suất sử dụng:  cao trung bình |
| **Nhóm Behavioral (nhóm tương tác)** | | |
| 13 | Chain of Responsibility | 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.Tránh việc gắn kết cứng giữa phần tử gởi request với phần tử nhận  và xử lý request bằng cách cho phép hơn 1 đối tượng có có cơ hội xử lý request . Liên kết  các đối tượng nhận request thành 1 dây chuyền rồi “pass” request xuyên qua từng đối tượng  xử lý đến khi gặp đối tượng xử lý cụ thể.  Tần suất sử dụng:  trung bình thấp |
| 14 | Command | 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.Đóng gói request vào trong một Object , nhờ đó có thể nthông số hoá chương trình nhận request và thực hiện các thao tác trên request: sắp xếp, log, undo…  Tần suất sử dụng: cao trung bình |
| 15 | Interpreter | Hỗ trợ việc định nghĩa biểu diễn văn phạm và bộ thông dịch cho một ngôn ngữ.  Tần suất sử dụng:  thấp |
| 16 | Iterator | 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.  Tần suất sử dụng:   cao |
| 17 | Mediator | Định nghĩa một đối tượng để bao bọc việc giao tiếp giữa một số đối tượng với nhau.  Tần suất sử dụng:  trung bình thấp |
| 18 | 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.  Tần suất sử dụng:  thấp |
| 19 | Observer | 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.  Tần suất sử dụng:  cao |
| 20 | State | Cho phép một đối tượng thay đổi hành vi khi trạng thái bên trong của nó thay đổi , ta có cảm  giác như class của đối tượng bị thay đổi.  Tần suất sử dụng:  trung bình |
| 21 | Strategy | 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.Cung cấp một họ giải thuật cho phép client chọn  lựa linh động một giải thuật cụ thể khi sử dụng.  Tần suất sử dụng:  cao trung bình |
| 22 | Template method | 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.  Tần suất sử dụng:  trung bình |
| 23 | Visitor | 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.  Tần suất sử dụng:  thấp |