Nhận xét của giáo viên

Nội dung

[1. Tổng quan 3](#_Toc469737536)

[1.1. Giới thiệu 3](#_Toc469737537)

[1.2. Phạm vi báo cáo 3](#_Toc469737538)

[1.3. Các khái niệm OOP được sử dụng 3](#_Toc469737539)

[2. Giới thiệu về mẫu thiết kế 3](#_Toc469737540)

[2.1. Khái niệm mẫu thiết kế: 3](#_Toc469737541)

[2.2. Lịch sử ra đời 3](#_Toc469737542)

[2.3. Phân loại 4](#_Toc469737543)

[2.4. Lợi ích khi sử dụng mẫu 4](#_Toc469737544)

[2.5. Cấu trúc trình bày mẫu 6](#_Toc469737545)

[3. Các mẫu Thiết kế GoF 7](#_Toc469737546)

[3.1. Mẫu Singleton 7](#_Toc469737547)

[3.2. Mẫu Proxy 7](#_Toc469737548)

[3.3. Mẫu Template 7](#_Toc469737549)

[3.4. Mẫu Façade 7](#_Toc469737550)

[3.5. Mẫu Observer 7](#_Toc469737551)

[4. Tổng kết 8](#_Toc469737552)

[5. Tài liệu tham khảo 9](#_Toc469737553)

1. **Tổng quan** 
   1. **Giới thiệu**

Design Pattern có vai trò cực kì quan trọng trong ngành công nghiệp phần mềm. Đã từ lâu, việc thiết kế các chương trình theo hướng đối

* 1. **Phạm vi báo cáo**

Trong phạm vi báo cáo này trình bày về 5 mẫu DP có tần suất sử dụng tương đối cao, được áp dụng trong nhiều lĩnh vực khác nhau, bao gồm : Façade, Template, Observe, Proxy và Singleton.

* 1. **Các khái niệm OOP được sử dụng**

Trong quá trình mô tả các mẫu thiết kế, có khá nhiều các khái niệm trong OOP (lập trình hướng đối tượng) được sử dụng lại. Một số thuật ngữ tương đối khó hiểu và dễ nhầm lẫn khi chuyển đổi từ tiếng Anh sang tiếng Vệt cũng sẽ được nói sơ qua nhằm phục vụ cho nội dung báo cáo.

1. Ngôn ngữ lập trình

Ra đời vào đầu thế kỉ XIX, cùng với sự phát triển của ngành công nghiệp chế tạo máy tính, hiểu biết của con người đối với lập trình cũng ngày càng tăng lên. Từ các chỉ thị máy được đục trên bằng đến các câu lệnh được máy tính biên dịch,

1. Hướng tiếp cận
2. Lập trình hướng đối tượng

* Abstraction:

Trong lập trình hướng đối tượng, đầu tiên ta sẽ gặp các khái niệm về sự trừu tượng và sự hiện thực. Sự trừu tượng hay Abstraction là khái niệm mô tả cách chúng ta nhìn nhận các đối tượng trong thực tế thành các thành phần trong mã nguồn của chúng ta. Ví dụ: một chiếc xe trong thực tế sẽ được chúng ta trừu tượng hóa thành một lớp tên là Xe trong C#. <<mã nguồn>>. Còn sự hiện thực là khái niệm mô tả cách thức một đối tượng sau khi trừu tượng hóa sẽ hoạt động như thế nào. Ví dụ Xe ở trên, nếu ta hiện thực lớp này, nó sẽ có các khả năng run, stop, ….

* Interface
* Implementation
* Inheritance
* Composition
* Is-a, Has-a
* Polymorphism
* Creation of object

1. **Giới thiệu về mẫu thiết kế** 
   1. **Khái niệm mẫu thiết kế:**

Trong quá trình thiết kế phần mềm, chúng ta có thể gặp phải một vài vấn đề khó giải quyết, hoặc cách giải quyết của chúng không phù hợp, hoặc giải quyết vấn đề không trọn vẹn, chúng ta có thể nghĩ đến các mẫu thiết kế phần mềm. Nói chung, mẫu thiết kế là một giải pháp chung được các lập trình viên và các nhà phát triển đi trước tìm ra và tổng hợp lại, chúng thường được áp dụng trong việc giải quyết một số vấn đề thường xảy ra trong quá trình chúng ta thiết kế phần mềm.

Nói một cách khác, ta có thể coi một mẫu thiết kế như một khuôn mẫu mô tả cách thức giải quyết một vấn đề mà ta có thể sử dụng trong nhiều tình huống khác nhau.

* 1. **Lịch sử ra đời**

Khái niệm mẫu thiết kế mà chúng ta đang dùng có nguồn gốc trong lĩnh vực kiến trúc xây dựng. Một kỹ sư người Áo là Christopher Alexander (sinh năm 1936) đã nhận thấy có rằng có nhiều thiết kế được sử dụng nhiều lần trong quá trình thiết kế các công trình kiến trúc. Ông đã xuất bản hai quyển sách có tầm ảnh hưởng lớn là “A Pattern Language : Towns, Buildings, Construction” (xuất bản năm 1977) và “The Timeless Way of Building” (xuất bản năm 1979). Trong các quyển sách này, ông đã mô tả các mẫu thiết kế kiến trúc “Mỗi mẫu mô tả một vấn đề mà chúng xảy ra lặp đi lặp lại trong môi trường của chúng tôi. Đồng thời mô tả giải pháp cốt lõi để giải quyết vấn đề đó, theo một cách mà bạn có thể sử dụng giải pháp này hàng triệu lần mà không cần phải giải quyết một vấn đề giống nhau hai lần.

Gần hai thập kỉ sau, các chuyên gia trong lĩnh vực phần mềm bắt đầu kết hợp các nguyên lý của Alexander và việc tạo ra các tài liệu hướng dẫn đầu tiên về mẫu thiết kế cho các nhà phát triển mới. Vào năm 1994, hội nghị đầu tiên về các mẫu thiết kế được tổ chức. Tên của hội nghị là Pattern Languages of Program Design (PLoP), tạm dịch là “Các ngôn ngữ mô mẫu của thiết kế chương trình”. Không lâu sau đó (1995), cuốn sách có tầm ảnh hưởng lớn nhất đến các mẫu thiết kế phần mềm là *Design Patterns: Elements of Reusable Object-Oriented Software* đã được xuất bản, nó giới thiệu 23 mẫu thiết kế đầu tiên, do bốn người Erich Gamma, Richard Helm, Ralph Johnson và John Vlissides đồng tác giả. Các tác giả sau này thường được nhắc đến với biệt danh *Gang of Four*  hay GoF. Kể từ đó, càng ngày càng có nhiều mẫu thiết kế được ghi chép và phân loại. Tuy nhiên, 23 mẫu GoF đầu tiên được biết đến và được ứng dụng nhiều nhất.

* 1. **Phân loại**

Các mẫu thiết kế được chia thành 3 loại chính: Mẫu kiến tạo, mẫu kiến trúc và mẫu hành vi.

1. *Mẫu kiến tạo – Constructional Pattern*

Các mẫu kiến tạo được sử dụng cho việc quản lý việc khởi tạo lớp (class instantiation). Các mẫu này cũng có thể được chia thành 2 nhóm nhỏ hơn là các mẫu khởi tạo lớp (class-creation) và các mẫu khởi tạo đối tượng (object-creation)

* Mẫu khởi tạo lớp: sử dụng kế thừa một cách hiệu quả trong quá trình khởi tạo.
* Mẫu khởi tạo đối tượng: sử dụng sự ủy nhiệm một cách hiệu quả để hoàn thành công việc.

Bao gồm các mẫu:

|  |  |
| --- | --- |
| Mẫu | Mô tả ngắn gọn |
| Abstract Factory | Tạo ra thể hiện của một vài lớp thuộc cùng một gia đình. |
| Factory Method | Tạo ra thể hiện của các lớp dẫn xuất con |
| Singleton | Một lớp chỉ có duy nhất một đối tượng tồn tại |
| Prototype | Một đối tượng được khởi tạo đầy đủ có thể được sao chép |
| Builder | Tách quá trình khởi tạo đối tượng khỏi hiện thực chi tiết của nó |

1. *Mẫu kiến trúc - Structural Pattern*

Các mẫu kiến trúc đều nói về việc kết hợp các lớp và các đối tượng.l

Bao gồm các mẫu:

|  |  |
| --- | --- |
| Mẫu | Mô tả Ngắn gọn |
| Adapter | Tương thích giao diện thành của các lớp khác nhau. |
| Bridge | Tách giao diện một đối tượng ra khỏi hiện thực của nó. |
| Composite | Một cấu trúc cây của các đối tượng (đơn giản và phức tạp). |
| Decorator | Thêm các trách nhiệm vào một đối tượng một cách tự động. |
| Facade | Một lớp duy nhất đại diện toàn bộ cho một hệ thống con. |
| Flyweight | Một thể hiện nhỏ vừa đủ để sử dụng trong việc chia sẻ. |
| Proxy | Một đối tượng đại diện cho một đối tượng khác. |

1. *Mẫu hành vi - Behavioral Pattern*

|  |  |
| --- | --- |
| Mẫu | Mô tả Ngắn gọn |
| Chain of resp | Cách thức để chuyển một yêu cầu qua một chuỗi các đối tượng. |
| Command | Đóng gói một yêu cầu điều khiển thành một đối tượng. |
| Interpreter | Cách thức thêm các phần tử ngôn ngữ vào một chương trình. |
| iterator | Truy cập tuần tự đến các phần tử của một tập hợp. |
| Mediator | Định nghĩa sự giao tiếp giản thể hóa giữa các lớp. |
| Memento | Lưu giữ và phục hồi trạng thái bên trong của một đối tượng. |
| Observer | Cách thức để thông báo sự thay đổi đến một hay nhiều lớp. |
| state | Thay đổi hành vi của một đối tượng khi trạng thái của nó thay đổi. |
| Strategy | Đóng gói một thuật toán được bên trong một lớp. |
| Template Method | Chuyển một số bước thực hiện một thuật toán của một lớp đến các lớp con của nó. |
| visitor | Định nghĩa một hoạt động mới trên một class mà không làm thay đổi lớp đó. |

* 1. **Lợi ích khi sử dụng mẫu**

Việc sử dụng các mẫu thiết kế trong quá trình sản xuất phần mềm có rất nhiều lợi ích

1. Mẫu thiết kế tạo điều kiện cho việc tái sử dụng

Một trong những vấn đề gây nhiều tranh cãi nhất trong phát triển phần mềm là làm thế nào để kết hợp tái sử dụng nhiều hơn. Nhiều người trong số các bên liên quan tham gia phát triển phần mềm nhận ra sự cần thiết và cơ hội để tái sử dụng mã, nhưng thử thách sẽ ngày càng lớn hơn để có thể tái sử dụng trên quy mô lớn. Các mẫu thiết kế cung cấp cho chúng ta một cách để tái sử dụng các giải pháp thiết kế qua nhiều ứng dụng.

1. Mẫu thiết kế làm cho việc thiết kế đơn giản hơn

The previous section illustrates how the application of design patterns can make the design phase easier. When a problem from the analysis phase is one for which a design pattern exists the design pattern offers an instant solution of high quality. Design patterns make design easier but their application still requires intellect. You must be able to recognize when the pattern applies and how to specialize the pattern for the particular context. Applying design patterns is much easier than solving design problems from first principles but it's still not algorithmic

* 1. **Cấu trúc trình bày mẫu**

Các mẫu thiết kế trong báo cáo sẽ được trình bày theo cấu trúc dưới đây:

|  |  |
| --- | --- |
| Mục trình bày | Nội dung trình bày |
| Tên, phân loại, bí danh | Mô tả ngắn gọn về mẫu |
| Mục đích, ý định | Mô tả mẫu này làm được những gì |
| Động lực sử dụng | Nêu một ví dụ về một vấn đề cần giải quyết có thể sử dụng mẫu này. |
| Khả năng ứng dụng | Liệt kê một số tình huống cụ thể trong thiết kế phần mềm có thể áp dụng mẫu này |
| Cấu trúc mẫu | Mô tả mẫu bằng một sơ đồ UML bao gồm các lớp và đối tượng |
| Các thành viên | Trình bày ý nghĩa của các lớp/đối tượng tha gia vào mẫu thiết kế và trcahs nhiệm của chúng |
| Sự cộng tác | Mô tả các thức các thành viên của mẫu tương tác với nhau như thế nào để thực hiện trách nhiệm của chúng |
| Các hệ quả mang lại | Trình bày về ưu điểm khi sử dụng mẫu, các nhược điểm sẽ dẫn tới khi sử dụng mẫu. |
| Các chú ý khi cài đặt | Các chú ý đặc biệt khi cài đặt mẫu |
| Mã nguồn minh họa | Trình bày một ví dụ demo cụ thể về mẫu |
| Ví dụ thực tế | Nêu ra những ví dụ thực tế về các hệ thống (đã pt triển và đang chạy) có sử dụng mẫu này |
| Các mẫu liên quan | Những mẫu nào có liên hệ đến mẫu này, những điểm quan trọng cần phân biệt, mẫu này có thể phối hợp với những mẫu nào. |

* 1. **Một số nguyên tắc**
* Bên sử dụng (client) không bao giờ gọi trực tiếp các lớp hiện thực, thay vào đó, chúng gọi các lớp ảo hoặc các interface (sự trừu tượng hóa của các lớp đó).
* Các thay đổi trong tương lai không ảnh hưởng đến hệ thống hiện tại

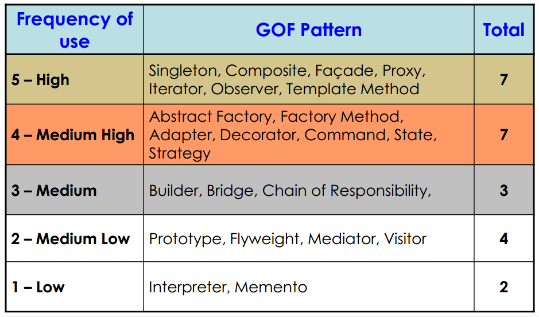
1. **Các mẫu Thiết kế GoF** 
   1. **Mẫu Singleton**
   2. **Mẫu Proxy**
   3. **Mẫu Template**
   4. **Mẫu Façade**
   5. **Mẫu Observer**
2. **Tổng kết**
3. **Phụ lục**

Một số phụ lục về Design Pattern

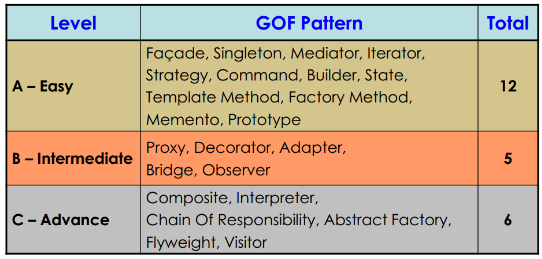
* 1. Phân loại

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | Mục đích | | |
| Khởi tạo | Cấu trúc | Hành vi |
| Phạm vi | Lớp | Factory Method | Adapter(class) | Interpreter  Template Method |
| Đối tượng | Abstract Factory  Builder  Prototype  Singleton | Adapter(object)  Bridge  Composite  Decorator  Facade  Flyweight  Proxy | Chain Of Resp  Command  Iterator  Mediator  Memento  Observer  State  Strategy  Visitor |
| Tổng cộng | | 5 | 7 | 11 |

* 1. Tuần suất sử dụng các mẫu



* 1. Độ khó của các mẫu



1. **Tài liệu tham khảo**

* Sách : thinking in java 4th Edition – tác giả - năm xuất bản – nhà xuất bản – chương sử dụng
* Tài liệu internet:
  + <http://sce2.umkc.edu/BIT/burrise/pl/design-patterns/>