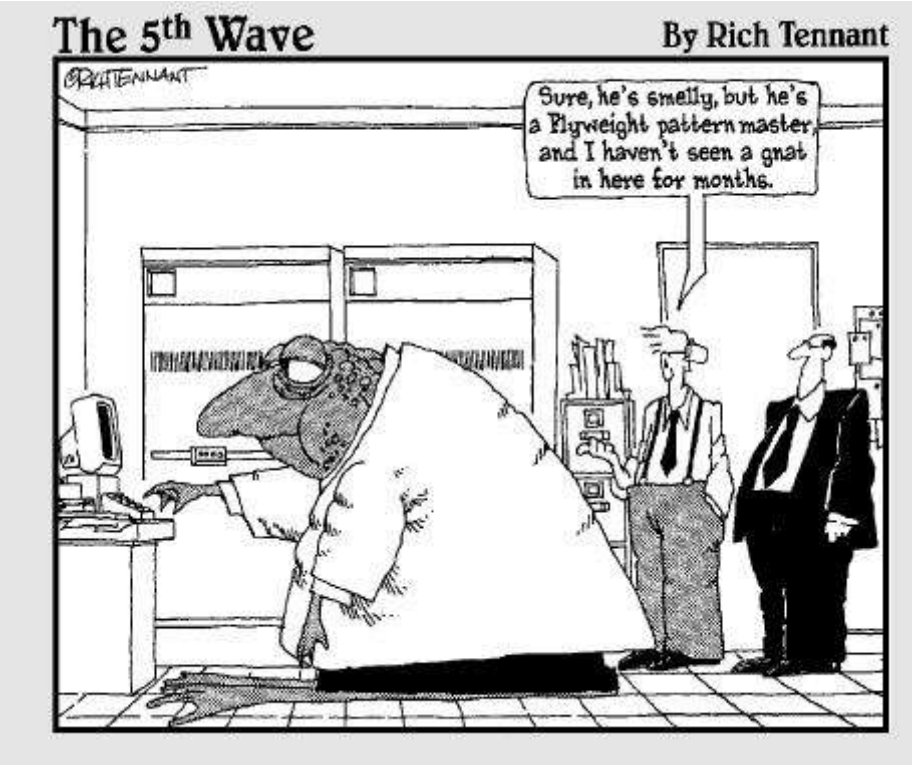
DP4Dummies – Chương 1 – Tổng quan các mẫu DP



Chương 1: Xin chúc mừng, rắc rối của bạn đã được giải quyết triệt để.

Trong chương này, chúng ta sẽ nói đến:

- Giới thiệu về Mẫu thiết kế Design Patterns là gì?
- Hiểu biết về tác dụng của Design Patterns
- Mở rộng lập trình hướng đối tượng
- Điểm sơ qua một số mẫu Design Pattern

Là một lập trình viên, bạn biết rằng thật khó khăn để nhớ chi tiết những việc bạn đang thực hiện. Và khi bạn không nắm bắt được tổng quát công việc, bạn có thể dễ dàng bỏ lỡ những việc quan trọng. Khi đó, mã nguồn bạn đang viết có thể vẫn còn làm việc tốt đẹp, nhưng trừ khi bạn bao quát được bức tranh lớn hơn, lúc đó mã nguồn bạn viết mới thực sự hoàn hảo.

Những vấn đề nghiêm trọng thực sự thường xuất hiện sau khi bạn đã chỉnh sửa chúng ít nhất một lần. Những nhà phát triển thường tự mình xử lý bằng cách viết lại mã nguồn và

sửa các lỗi. Tuy nhiên trong môi trường công việc, những nhà lập trình thường bỏ phần lớn thời gian để bảo trì, chỉnh sửa những công việc cũ hơn là tập trung vào những sản phẩm mới.

Bạn thấy rằng thật vô lý khi cứ phải làm, rồi sửa, làm lại, sửa tiếp... Giải pháp hợp lý nhất là bạn đưa ra được một quy trình tổng quan cho việc thiết kế và bảo trì, có như vậy, bạn mới tránh được các rắc rối phát sinh khi môi trường ứng dụng thay đổi, hoặc ít nhất bạn cũng giúp cho việc bảo trì, chỉnh sữa dễ dàng hơn khi có phát sinh.

Ý tưởng đằng sau cuốn sách này là: Bạn sẽ sử dụng một tập hợp các mẫu thiết kế Design Patterns để làm đơn giản hóa quá trình trên. Kế hoạch này sẽ giúp bạn có một cái nhìn tổng quát. Một mẫu thiết kế Design Pattern là một giải pháp đã được kiểm nghiệm thành công khi đối diện một vấn đề lập trình phát sinh cụ thể. Khi bạn quen thuộc hết các mẫu thiết kế trong sách này, bạn nhìn vào một chương trình. Bam! Một giải pháp đúng đắn sẽ xuất hiện trong tâm trí bạn, thay vì bạn phải đập đầu vào tường trong vô vọng, giờ bạn có thể ung dung nói "Ở đây, tôi sẽ sử dụng mẫu Factory, mẫu Observer, hay mẫu Adapter..."

Đó là chưa nói, một số sách về thiết kế khuyên bạn nên dành phần lớn thời gian để phân tích và lên kế hoạch cho một đề án. Vẻ đẹp thật sự ở đây là một người nào đó đã đối mặt với vấn đề bạn đang gặp phải, họ đã có giải pháp đúng đắn cho nó. Và giờ khi bạn nhuần nhuyễn mẫu thiết kế, bạn có thể áp dụng các thiết kế đó một cách dễ dàng.

Làm sao để trở thành chuyên gia thiết kế trong lĩnh vực phần mềm, điều mà ai cũng thèm muốn? Thật dễ dàng, hãy đọc cuốn sách này, nghiền ngẫm những mẫu thiết kế mà tôi dành nhiều tâm huyết để viết. Bạn không cần phải nhớ mọi thứ, bạn chỉ cần biết là có những mẫu thiết kế đó. Và khi bạn đối diện với một vấn đề thực tế, sâu thẳm trong bạn tự nhiên thốt lên "À, có vẻ chỗ này có thể dùng mẫu Iterator..." Sau đó bạn chỉ cần tìm kiếm mẫu thiết kế đó trong cuốn sách này, duyệt qua các ví dụ để biết phải làm gì. Và vì vậy, chương này sẽ là một tour du lịch nho nhỏ, giúp bạn đi qua một số mẫu thiết kế tiện dụng và hữu ích.

Chỉ cần tìm ra mẫu thiết kế thích hợp

Điểm kỳ diệu của Design Patterns là nó giúp cho công việc của bạn dễ dàng tái sử dụng, dễ mở rộng và bảo trì. Khi bạn thiết kế không tốt, phần mềm của bạn không có khả năng tái sử dụng và bảo trì, khi gặp vấn đề phát sinh, bạn sẽ phải dành nhiều thời gian, có khi là nhiều hơn cả lúc bạn viết ban đầu, chỉ là để sửa chữa chúng.

Ví dụ: Bạn đang muốn tạo một đối tượng Java, nhiệm vụ là đọc và phân tích một tài liệu XML. Bạn cần phải tạo một lớp Parser (chuyên dùng để phân tích XML) sau đó bạn tạo một đối tượng của lớp này. Bạn thầm nghĩ "Tới giờ mọi việc vẫn ổn". Nhưng thực tế thì đã có hàng tá lớp Parser do người khác viết, và họ luôn muốn sử dụng lại những tính năng đặc biệt trong lớp của họ. Nếu bạn có thể sử dụng mẫu thiết kế Nhà máy Factory, giờ đây bạn có thể sử dụng bất cứ lớp Parser nào, kể cả của những người khác, thay vì cứ khư khư xài lớp Parser do chính bạn viết ra. Và vì vậy, chương trình của bạn đã trở nên dễ mở rộng, tái sử dụng được và bảo trì dễ dàng.

Nói cách khác, Mẫu thiết kế Design Patterns là những giải pháp giúp cho ta có một thiết kế tốt khi đối diện những vấn đề phát sinh trong việc lập trình. Nhiều người đã gặp vấn đề này, và đã giải quyết tốt, việc của bạn là áp dụng chúng. Bạn không cần phải ghi nhớ mọi thứ, chỉ cần nhìn ra đâu là mẫu thiết kế phù hợp và đặt nó vào đúng chỗ.

Thật tuyệt đúng không các bạn.

Đôi nét về cuốn sách tên "Gang of Four" Bộ tứ tác giả.

Quyển sách là tập hợp 23 mẫu thiết kế được phát hành bởi Erich Gamma, Richard Helm, Ralph Johnson và John Vlissides, trong một nghiên cứu của họ năm 1995. Với tựa đề gốc là "Design Patterns: Elements of Reusable Object-Oriented Software". Tạm dịch "Mẫu thiết kế: những thành phần tái sử dụng trong lập trình hướng đối tượng". Họ được giới lập trình gọi là Bộ tứ Gang of Four, hay GoF. (ND: Bộ tứ ở đây là ẩn dụ với Tên nhóm nhạc nổi tiếng Gang Of Four của Anh hay Bộ tứ quyền lực Mafia trong tác phẩm Bố già hay là bộ tứ quyền lực chính trị của Trung quốc, ở VN cũng có bộ tứ quyền lực của Vietnam Next Top Model...).

Đã có rất nhiều sự thay đổi kể từ khi xuất hiện, một số trong 23 mẫu được sử dụng nhiều, số khác ít khi được sử dụng. Tôi sẽ nói đến đầy đủ 23 mẫu trong cuốn sách này, nhưng tôi sẽ nhấn mạnh ở những mẫu thường được sử dụng hơn. Và kể cả mẫu mới không có trong sách của GoF, trong chương 11.

Có một sự thật là, bạn không chỉ phải nhớ kỹ từng mẫu thiết kế, bạn phải hiểu sâu sắc về nó, để có thể áp dụng đúng đắn trong thực tiễn. Tôi cũng sẽ lưu ý nhiều về lập trình hướng đối tượng xuyên suốt quyển sách này. Lập trình hướng đối tượng OOP là một bước tiến tuyệt vời trong lĩnh vực lập trình. Nhưng có quá nhiều lập trình viên sử dụng chúng một cách tùy tiện, thiếu chiều sâu, và điều đó gây ra nhiều rắc rối tiềm ẩn.

Phần lớn việc tìm hiểu những mẫu thiết kế chính là việc mở rộng khái niệm lập trình hướng đối tượng. Ví dụ: đóng gói những gì thay đổi nhiều nhất (encapsulating what changes most), cách chuyển đổi một quan hệ kế thừa is-a sang quan hệ kết hợp has-a (xem chương 2 để biết chi tiết) – và tôi sẽ nói chi tiết về chúng.

Hãy bắt đầu bằng mẫu Mediator Pattern (Người trung gian)

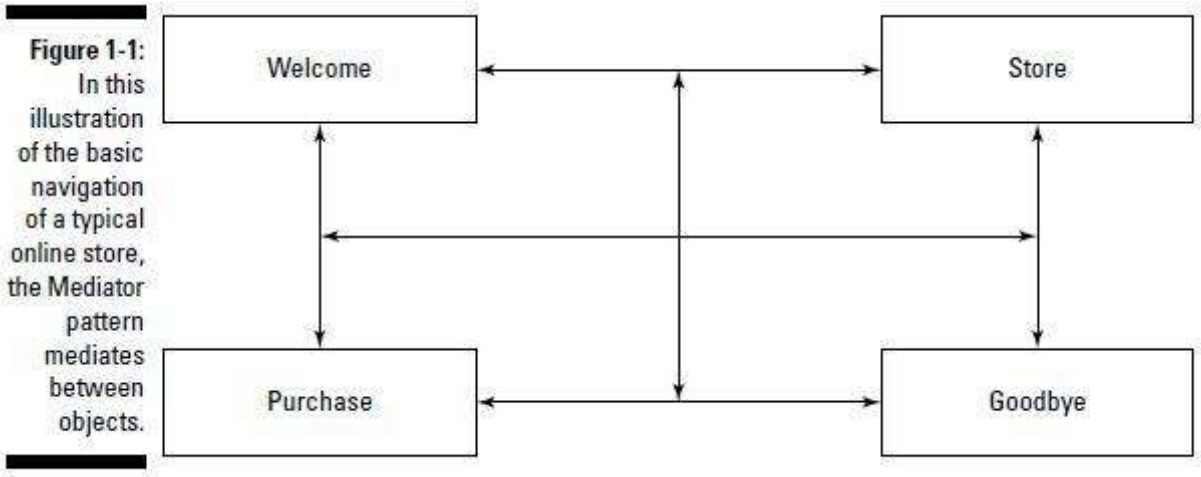


Figure 1-1: In this illustration of the basic navigation of a typical online store, the Mediator pattern mediates between objects.

Hình bên trên là một ví dụ về mẫu thiết kế, mẫu Mediator. Hình cho chúng ta thấy chức năng của một mẫu Mediator. Theo hình ta đang có một website với 4 trang. Website cho phép khách hàng duyệt qua kho hàng và đặt mua. Khách hàng có thể đi từ trang này qua trang khác theo đường vẽ trên hình. Ở đây có một vấn đề phát sinh. Tại từng trang, bạn phải viết mã để nhận biết khi nào khách hàng muốn nhảy qua trang khác và kích hoạt trang đó. Tại một trang bạn có quá nhiều đường để đi tới trang khác, và vì vậy sẽ phát sinh nhiều đoạn code trùng lặp trên nhiều trang khác nhau.

Bạn có thể sử dụng mẫu Mediator để đóng gói tất cả các đường dẫn tới trang vào một module duy nhất, và đặt nó vào trong một đối tượng Mediator. Từ bây giờ, từng trang chỉ cần phải thông báo bất cứ sự thay đổi nào cho Mediator, và Mediator tự biết dẫn trang cần thiết cho khách hàng, như trong hình bên dưới.

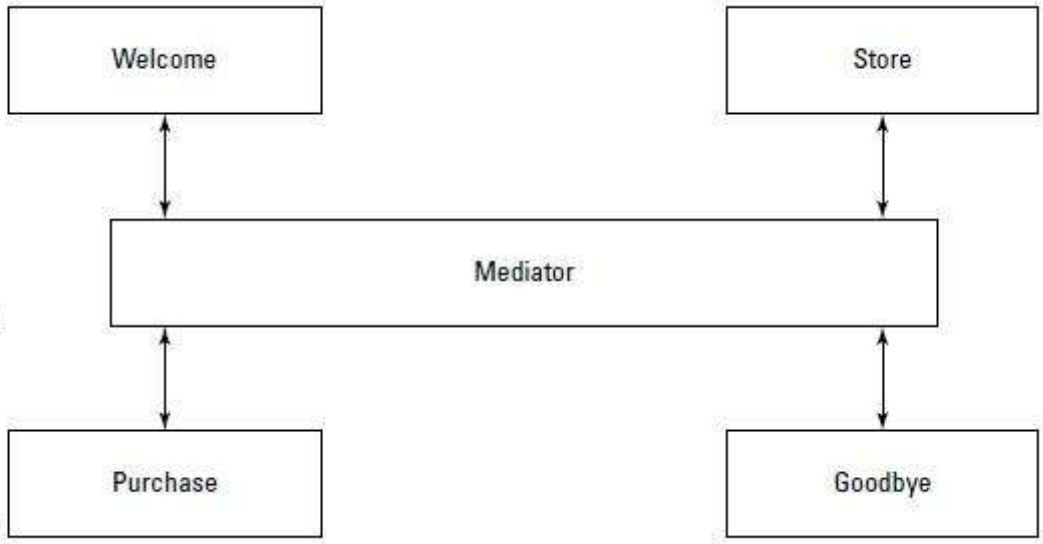


Figure 1-2: The mediator directs Web site traffic.

Bạn có thể tạo ra một Mediator với chức năng dẫn trang. Tại đây bạn có thể chỉnh sửa và thay đổi dễ dàng. Đó chính là chức năng của Mediator (Người trung gian)

Chuyển đổi với mẫu thiết kế Adaptor (Người chuyển đổi)

Đây là một mẫu khác, mẫu chuyển đổi Adaptor.

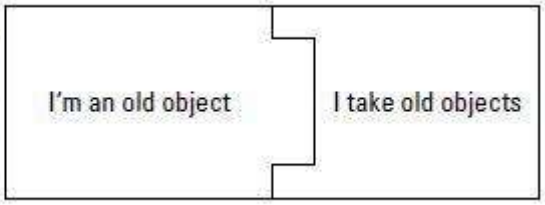


Figure 1-3: Everything seems to be working here.

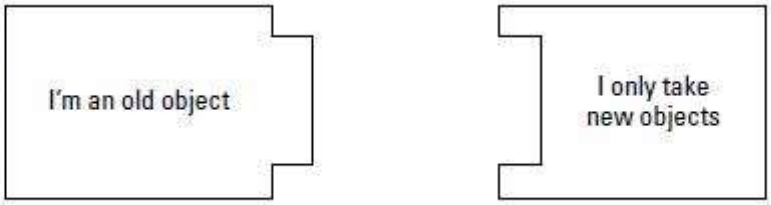


Figure 1-4: This isn't going to work.

Figure 1-5: Old objects work with new objects via an adapter.

Hãy nhìn vào hình 1-3. Đầu vào là một đối tượng cũ. Hệ thống tiếp nhận đối tượng cũ. Với hình 1-4. Khi hệ thống thay đổi, hệ thống không tiếp nhận đối tượng cũ nữa, chỉ tiếp nhận đối tượng mới. (ND: thực ra hình ảnh 1-4 có chút chưa chính xác, phần "I only take new objects, nên vẽ nhỏ lại)
Hình 1- 5. Đây là nơi xuất hiện mẫu Adaptor (Người chuyển đổi). Mục đích là chuyển đổi đối tượng cũ, thành đối tượng mới, khi đó hệ thống sẽ sẵn sàng tiếp nhận đối tượng này.

"ND: Các bạn hãy tưởng tượng. Các bạn có một chiếc tivi với đầu cắm điện 3 chân. Ổ cắm điện ở nhà bạn là loại ổ 2 chân. Bạn ra ngoài cửa tiệm, mua 1 cục chuyển đổi, từ 3 chân ra 2 chân. Lúc đó bạn có thể sử dụng được ổ điện 2 chân rồi. Cục chuyển đổi từ 3 chân ra 2 chân, chính là Adaptor"

Vấn đề đã được giải quyết. Ai nói học các mẫu thiết kế là khó khăn nhỉ.

Đứng trong một đối tượng. Mẫu Proxy. (Người đại diện)

Đây là một mẫu khác. Mẫu Proxy. Mẫu này nói rằng, mã bạn viết chỉ tương tác với đối tượng cục bộ như hình dưới

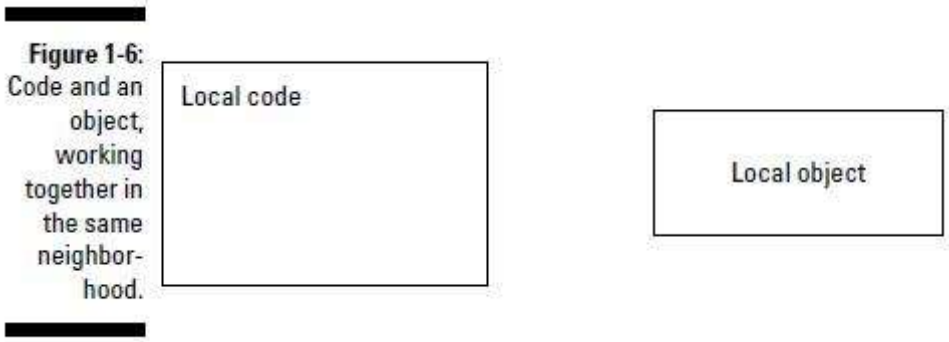


Figure 1-6: Code and an object, working together in the same neighbor-hood.

Thực tế phát sinh, bạn buộc phải tương tác với một đối tượng ở xa, đâu đó trên thế giới. Làm sao bạn có thể làm cho chương trình của bạn tương tác với một đối tượng cục bộ trong khi thực tế là nó đang làm việc với một đối tượng ở xa.

Ở đây mẫu Proxy (Người đại diện) xuất hiện. Nó là một đối tượng nằm bên trong chương trình, làm trách nhiệm tương tác với chương trình, giúp cho chương trình tưởng rằng nó đang tương tác cục bộ, thay vì tương tác với một đối tượng từ xa. Bên trong, Proxy chịu trách nhiệm kết nối với đối tượng từ xa. Như hình dưới

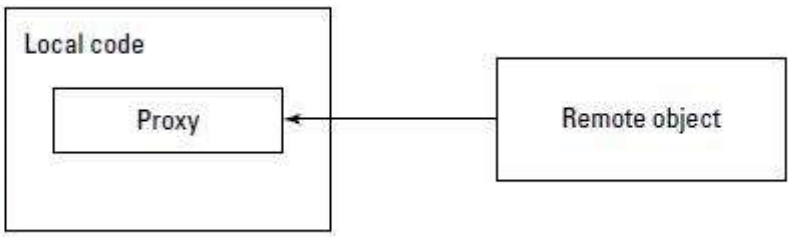


Figure 1-7: Trick your local code and remote object into working together.

Bạn sẽ biết cách mẫu Proxy hoạt động trong chương 9.

Đôi nét về mẫu Observer (Người quan sát)

Bạn có thể quen thuộc với một vài mẫu trong sách này, ví dụ như mẫu Observer này chẳng hạn.

Mẫu Observer có thể đăng ký với hệ thống. Khi hệ thống có sự thay đổi, hệ thống sẽ thông báo cho Observer biết. Khi không nữa cần, mẫu Observer sẽ được gỡ khỏi hệ thống.

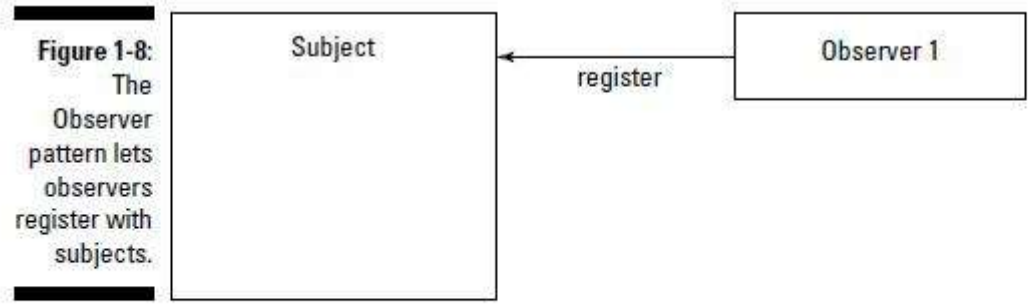


Figure 1-8: The Observer pattern lets observers register with subjects.

And another observer, Observer 2, can register itself as well, as shown in Figure 1-9.

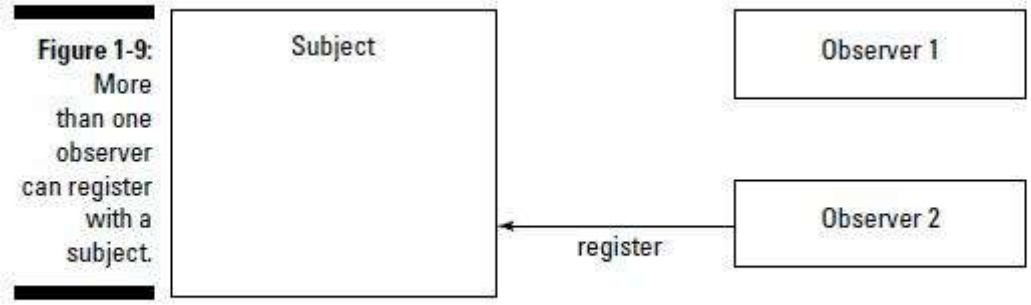


Figure 1-9: More than one observer can register with a subject.

Now the subject is keeping track of two observers. When an event occurs, the subject notifies both observers. (See Figure 1-10.)

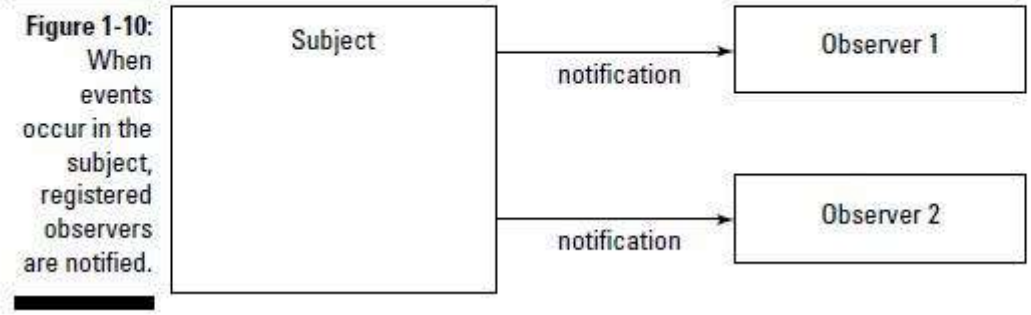


Figure 1-10: When events occur in the subject, registered observers are notified.

Hình 8 cho thấy mẫu Observer cho phép 1 observer đăng ký với hệ thống. Hình 9, cho phép observer thứ 2 đăng ký với hệ thống. Hiện tại hệ thống đang liên lạc với 2 observer. Khi hệ thống phát sinh một sự kiện cụ thể nào đó, nó sẽ thông báo với cả 2 observer như hình số 10.

Tôi cố gắng trình bày tất cả các Mẫu thiết kế theo cách dễ hiểu, dễ tiếp cận nhất. Bạn sẽ không phải nhìn vào đống biểu đồ, cùng với các lớp trừu tượng đầy phức tạp nữa. Các chương trong sách nhắm vào các độc giả là lập trình viên, rất hữu ích cho các bạn, cho dù các bạn không đọc hết tất cả các mẫu. Các mẫu thiết kế trong sách này đã trở thành các tiêu chuẩn về lập trình trên thế giới, và chúng hữu dụng cho các bạn, dù bạn đang ở trình

độ nào. Hy vọng rằng, trong tương lai, khi bạn đối diện với chương trình của mình, bạn đột nhiên nhận ra: Aha, đây chính là mẫu Façade.

Design Patterns For Dummies
Design Patterns
< Series bài dịch Design Patterns for Dummies – Giới thiệu
> DP4Dummies – Chương 2: Strategy