# **Thợ lành nghề #49: Túi nâu 6 – Abstract Factory**

Robert C. Martin

18/10/2006

…Tiếp nối phần trước

*Tháng 10, 1944*

*“Ý tưởng của Tiến sĩ Ulam thật ngây thơ.” Tiến sĩ Von Braun nói với Tiến sĩ Robert Goddar khi họ ăn trưa tại trụ sở Nimbus. “Một tàu không gian chạy bằng năng lượng hạt nhân có thể được đẩy đi dễ dàng bằng cách thả bom nguyên tử ra phía sau và bơm chúng lên sao!”*

*“Tôi biết,” Goddard nói, “Điều đó có vẻ hơi kinh khủng. Tôi đoán anh sẽ sử dụng một tấm chắn, một loại tấm chắn đẩy loại lớn để hấp thụ xung lượng của vụ nổ và truyền nó đến toàn bộ con tàu. ”*

*“Chính xác, Robert! Chúng ta sử dụng những quả bom có sức nổ 1kt với tỷ lệ xấp xỉ một quả bom mỗi giây. Một vài nghìn quả bom như vậy có thể đẩy toàn bộ cấu trúc của Nimbus, và tất cả phi hành đoàn, đến được* [*sao Thổ*](https://vi.wikipedia.org/wiki/Sao_Th%E1%BB%95) *và trở về! ”*

*“Ừ, tôi cho rằng khi anh sử dụng bom nguyên tử làm sức đẩy đẩy nhiên liệu không phải là vấn đề. Anh thực sự đang nghĩ về việc đi đến sao Thổ ư? ”*

*“Ồ, không phải, đó chỉ là vài thứ chúng ta có thể làm nếu chúng ta muốn. Tuy nhiên chúng ta sẽ cố gắng hạ cánh trên* [*sao Hỏa*](https://vi.wikipedia.org/wiki/Sao_H%E1%BB%8Fa) *vào tầm này năm sau.”*

*Goddard sửng sốt. Ông ta nhìn chằm chằm vào Von Braun với sự hoài nghi. Sau một vài giây ông ta lên tiếng: “Điều này khá là, ừm …, táo bạo, đúng chứ?”*

*Von Braun nghiêm nghị nhìn vào Goddard và không nói gì cả. Goddard rốt cuộc nhún nhường: “Ừ, tôi cho rằng chúng tôi đang có một chút vội vàng.” Von Braun chỉ gật đầu.*

*“Tại sao lại là sao Hỏa?”*

*“Đó là một thử nghiệm cần thiết về khả năng của con tàu, và nó có khi lại là trở thành ngôi nhà mới của chúng ta.”*

*Goddard gật đầu. “Có, kênh rạch, thảm thực vật, có lẽ thậm chí còn có cả một nền văn minh cổ đại. Tôi hy vọng họ không phẫn nộ chúng ta xuất hiện trong bộ dạng như thế này.”*

*“Không có ai ở đó đâu, Robert, tất cả họ đã chết cách đây hàng thế kỷ, hàng nghìn năm trước rồi. Và nếu có ai đó ở đó, chúng ta sẽ phải giải quyết vấn đề vào lúc đó. Anh có thể đặt cược rằng chúng ta đã sẵn sàng. ”*

*“Hmmph.” Goddard thừa nhận. Sau đó ông ta trầm tư một lúc và nói: “Werner, Anh đã xem xét đến việc nếu anh có thể đẩy một con tàu bằng bom nguyên tử, thì anh cũng có thể thay đổi quỹ đạo của Clyde theo cách tương tự hay chưa?”*

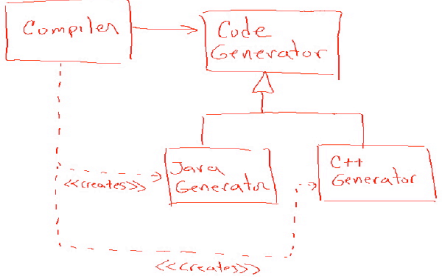
*Von Braun bắt đầu lắc đầu nhưng sau đó liền dừng lại và nhìn chằm chằm vào Goddard, và đôi mắt của ông ấy sáng bừng lên.*

Thứ 2, 4/3/2002, 1100

Ở đây có những người trước đây tôi chưa từng gặp. Tôi đếm có 25 người trong phòng. Kể cả Jean cũng ở đó.

Adelaide gõ thành tiếng để thu hút sự chú ý, và đám đông lắng xuống. Cô rõ ràng đang lo lắng, nhưng không để tâm trạng đi xuống. Cô ấy xốc lại tinh thần và bắt đầu.

“Được rồi, mọi người, Jasmine yêu cầu tôi nói về mô hình Abstract Factory. Cô ấy và tôi đã sử dụng mẫu này trong dự án SMCRemote, vậy nên tôi nghĩ tôi có thể nói vài lời về nó. Đây là những gì tôi biết:” Adelaide dùng tay bắt đầu vẽ lên tường.



“Đây là thiết kế cũ của trình biên dịch SMC, ngoại trừ việc chúng tôi đã thêm một tính năng cho phép nó tạo ra cả mã Java và C++. Lát nữa khi tôi trình bày cho các bạn, chúng ta đang đưa ra cấu trúc plug-in để các ngôn ngữ mới có thể được thêm vào trình biên dịch một cách đơn giản bằng cách tạo các dẫn xuất mới của lớp CodeGenerator. Nhưng thật không may, như mọi người có thể thấy trong thiết kế hiện tại, để cho lớp Compiler tạo ra instance thích hợp, nó phải có một dependency trực tiếp trong các dẫn xuất.”

“Vì vậy, chúng tôi đề xuất sử dụng mô hình ABSTRACT FACTORY để phá vỡ sự phụ thuộc này và cách ly lớp Compiler từ bất kỳ dẫn xuất mới nào của CodeGenerator.”

Rõ ràng là Adelaide đã luyện cách diễn đạt này một vài lần rồi. Cô ấy có một chút trơn tru quá mức, và cũng có một chút hơi nóng vội. Jerry hẳn đã nhận ra những điều này vì khi Adelaide quay lại để xóa sơ đồ, hắn ta đã ngăn cô lại với một câu hỏi hiển nhiên.

“Adelaide, tại sao điều đó lại là vấn đề khi Compiler biết về JavaGenerator và C++Generator? Với tôi dường như những dependency đó là hoàn toàn phù hợp. Xét cho cùng, lớp Compiler chỉ gọi các lớp mới và sau đó sử dụng chúng thông qua giao diện CodeGenerator, phải không?”

Adelaide khó chịu liếc nhìn Jasmine, nhưng sau đó hít một hơi và trực tiếp trả lời Jerry.

“Chúng tôi không muốn bất kỳ sự phụ thuộc nào từ Compiler đối với các dẫn xuất của CodeGenerator vì chúng tôi không muốn biên dịch lại Compiler cứ mỗi khi có một trong những dẫn xuất thay đổi. Lớp CodeGenerator ở đó là để cách ly có cách Compiler từ những thay đổi như thế. Sẽ thật đáng tiếc khi phí phạm sự biệt lập đó một cách đơn giản chỉ vì từ khóa new.”

Tôi không chắc mình đã thỏa mãn với lập luận đó, vậy nên tôi hỏi:

“Tại sao việc chuyển đổi thành các dẫn xuất bắt buộc phải biên dịch lại lớp Compiler?”

Adelaide trông hơi bối rối và liếc nhìn Jasmine một cách đầy ý nhị. Jasmine đứng lên để giải vây giúp cô ta và đưa tôi một cái nhìn lạnh lùng.

“Được rồi, Cao thủ, chúng ta đã nói về điều này trong suốt cuộc thảo luận về VISITOR tuần trước. Vì vậy, tôi tin là cậu biết câu trả lời. Nhưng vì lợi ích của những người khác trong căn phòng này, hệ thống xây dựng của chúng ta biên dịch lại các module dựa trên lịch sử sửa đổi của các tệp. Vì Complier phụ thuộc vào JavaGenerator, nếu JavaGenerator.java có ngày muộn hơn của Compiler.class, hệ thống xây dựng sẽ tự động xây dựng lại Compiler. Vậy đã đủ chưa hả Alphonse? ”

“Đủ rồi, ừm, cảm ơn.”

Một trong những bạn cũ học cùng lớp với tôi, Alex, đứng dậy và nói: “Vâng, chắc chắn rồi, nhưng mà cô có thể khắc phục điều này chỉ cần dùng tên của lớp và class.forName trong Complier.”

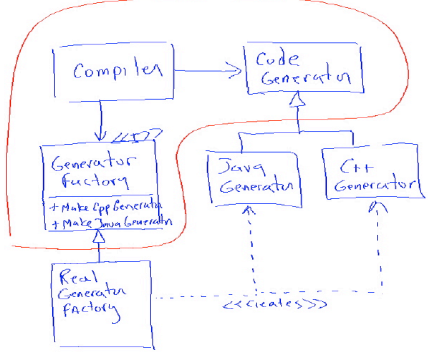
“Ừ, chúng ta có thể làm thế,” Jasmine đáp lại. “Nhưng chúng ta không muốn loại cơ chế chi tiết đó bên trong lớp Complier. Ta đang cố gắng tách rời các mối quan hệ.” Jasmine kéo dài hai chữ cuối với giọng điệu mỉa mai buộc Alex lại phải ngồi xuống.

Joseph đứng lên và nói: “Jasmine, Alex hỏi câu hỏi hợp lý, cô không cần phải ngắt lời cậu ấy như vậy.”

Jasmine thở dài và vai cô hơi chùng xuống một chút. Rồi cô ấy nói: “Được thôi, anh đã đúng, Joseph. Alex, tôi xin lỗi vì đã thô lỗ. Mục tiêu của chúng tôi ở đây là để duy trì Compiler từ việc biết cách các dẫn xuất được tạo ra, hoặc, thực tế hơn thì là, hiểu cặn kẽ về bất cứ thứ gì liên quan đến các dẫn xuất. Chúng tôi muốn điều này vì ta nghĩ rằng cơ chế tạo các dẫn xuất có thể thay đổi theo thời gian và ta không muốn những thay đổi đó tác động đến lớp Compiler. Chúng tôi muốn Complier chỉ đơn giản là trình biên dịch và không có thêm gì khác.”

Jasmine ngồi xuống và tôi thoáng thấy ánh mắt cô ấy hướng về phía Jerry.

Adelaide đứng quay lưng lại, vẫy tay vào tường để xóa sơ đồ cuối cùng, và sau đó vẽ những thứ sau đây:



Sau đó, cô ấy bắt đầu lại dùng giọng điệu đùa cợt lão luyện . “Đây là giải pháp được đề xuất của chúng tôi. Nó là ví dụ điển hình của mẫu ABSTRACT FACTORY. Lưu ý đến giao diện GeneratorFactory. Lớp Compiler gọi một trong hai phương thức makeXXXGenerator của giao diện này. Để phản hồi RealGeneratorFactory tạo ra instance thích hợp và trả nó về cho Complier. Bây giờ hãy chú ý đến đường màu đỏ… ”

“Tôi rất tiếc, thân yêu à.” Jean ngắt lời. “Nhưng tôi e rằng tất cả UML này đều đang làm tôi hơi đau đầu. Tôi đoán tôi có chút lạc hậu khi chỉ muốn xem một số đoạn mã. Cô là một cô gái quá sức ngọt ngào, cô có phiền khi cho tôi xem mã không? “

Bằng cách nhìn vào khuôn mặt của Adelaide, tôi nghĩ rằng với cô ấy đã quá rõ ràng rằng nàng sẽ không bao giờ hoàn thành được bài thuyết trình được chuẩn bị sẵn của mình. Với địa vị của cô ấy, nàng không hoảng sợ, hoặc thậm chí liếc qua Jasmine để được nhận được trợ giúp. Thay vào đó, cô dừng lại một lúc, rồi mở một cửa sổ TextMate trên tường. Sử dụng bàn phím ảo của tường cô ấy gõ:

*public interface GeneratorFactory {*

*public CodeGenerator makeCppGenerator();*

*public CodeGenerator makeJavaGenerator();*

*}*

*public class RealGeneratorFactory implements GeneratorFactory {*

*public CodeGenerator makeCppGenerator() {return new CppGenerator();}*

*public CodeGenerator makeJavaGenerator() {return new JavaGenerator();}*

*}*

“Đoạn mã này có hữu ích chứ?” Cô lịch sự hỏi.

Jean thậm chí không mất một giây để xem mã. Bà ấy chỉ mỉm cười dịu dàng và nói, “Ừ, thân yêu, nó thực sự tốt.”

Avery thì thầm vào tai tôi: “Jean chỉ muốn chắc chắn rằng Adelaide biết cách viết mã.”

Tôi cũng có chiều hướng đồng ý, nhưng không nói gì để đáp lại lời.

“Được rồi, giờ nhìn vào vạch đỏ.” Adelaide tiếp tục. “Đường màu đỏ phân định kiến trúc plug-in của chúng tôi. Mọi thứ bên ngoài đường màu đỏ là một plug-in. Mọi thứ bên trong ranh giới màu đỏ là động cơ cốt lõi của chúng tôi. Lưu ý tới cách tất cả các phụ thuộc vượt qua các điểm màu đỏ vào bên trong! Điều này có nghĩa là các thay đổi đối với các yếu tố bên ngoài vòng ranh không ảnh hưởng đến các yếu tố bên trong vòng ranh. Ví dụ như: nếu một trong các dẫn xuất của trình tạo mã thay đổi, thì khu vực trong ranh giới không cần phải được biên dịch lại.”

“Quả thật có quá nhiều thứ tốt!” Avery đã đứng dậy và đang nói rất to và mạnh mẽ. “Cấu trúc này có cùng chung một vấn đề mà VISITOR gặp phải. Có một chu kỳ phụ thuộc giữa GeneratorFactory và các dẫn xuất CodeGenerator. Mỗi lần ta thêm một đạo hàm generator mới, ta cũng phải thêm phương thức tạo mới vào giao diện GeneratorFactory, điều đó có nghĩa là vùng ranh màu đỏ của cô thực sự cần phải được biên dịch lại. Vì vậy, toàn bộ mô hình này thực là vô giá trị! ”

Avery đứng đó trừng trừng tức giận trong khi Adelaide trông như sắp khóc. Jasmine lao ra khỏi chỗ của mình. “Avery, cậu hơi…”

Nhưng trước khi Jasmine có thể hành động, Jean đứng dậy một cách cứng nhắc và nói: “Avery thân mến, cậu có thể đi cùng tôi không.” Jasmine tròn mắt, cặp lông mày cô ta thoáng nhướn lên, rồi lại ngồi xuống. Mặt của Avery trở nên trắng bệch. Jean chỉ tay ra hiệu hắn ta hướng về phía cửa, và hai người họ rời đi trong sự im lặng khó chịu.

Sự căng thẳng không thoải mái vẫn còn duy trì trong một lúc, và sau đó Jasper nói: “Thế đấy, không phải chỉ mới tuần trước những buổi họp này thật vui sao!” Không khí căng thẳng bị xua tan và mọi người cùng cười và thư giãn … một chút.

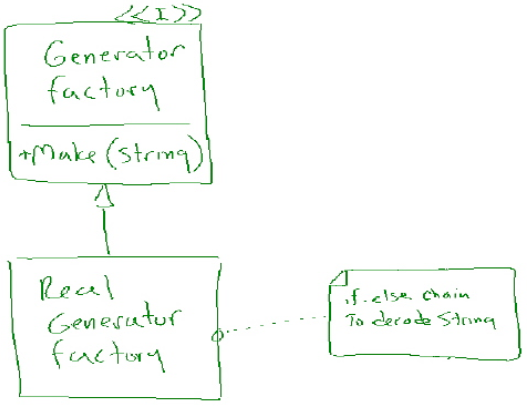
Jerry đứng lên. “Avery có thể diễn đạt nó tốt hơn, nhưng cậu ta cũng có ý đúng, phải không? Ta thực sự phải thay đổi và biên dịch lại động cơ chủ chốt mỗi lần một trình tạo mã mới được thêm vào.”

Adelaide vẫn đang nhìn chằm chằm vào sàn tàu, gắng nén nước mắt, và dường như không thể trả lời nổi. Rốt cuộc Jasmine đứng dậy và nói: “Đúng vậy, đó là một phần có chủ ý của thiết kế. Chúng tôi hy vọng rằng các ngôn ngữ mới sẽ không thường xuyên gây ra quá nhiều phiền toái trong việc tự bảo vệ chúng khỏi vấn đề đó. Những gì chúng tôi đang cố gắng để cách ly phần trọng yếu ra khỏi đó là sự bảo trì thay đổi các trình tạo mã đã tồn tại. ”

Jared đứng lên và nói: “Được rồi, thế là đủ rồi, nhưng nếu ta muốn tự bảo vệ bản thân khỏi những ngôn ngữ mới thì sao? Cô sẽ làm gì sau đó? Làm thế nào cô có thể duy trì từ việc biên dịch lại động cơ lõi mỗi khi ai đó viết một đạo hàm tạo mã mới? ”

Có một khoảng im lặng ở phòng họp trong một chốc lát, và rồi tôi đứng dậy.

“Tôi đã nghĩ về vấn đề này trong vài ngày qua vì chúng tôi cũng có một vấn đề tương tự trong dự án Dtrack. Chúng tôi không muốn biên dịch lại động cơ cốt lõi của chúng tôi mỗi khi một loại đồ phi hành gia mới được tạo ra.” Tôi vẫy tay vào tường để xóa nó, và sau đó vẽ những thứ sau:



“Và chỉ là phòng khi Jean trở lại đây, tôi sẽ viết đoạn mã.”

*public interface GeneratorFactory {*

*public CodeGenerator makeGenerator(String type);*

*}*

*public class RealGeneratorFactory implements GeneratorFactory {*

*public CodeGenerator makeGenerator(String type) {*

*if (type.equals(“C++”)) return new CppGenerator();*

*if (type.equals(“Java”)) return new JavaGenerator();*

*return null;*

*}*

*}*

“Bây giờ hãy tưởng tượng rằng các chuỗi “C ++” và “Java” là các đối số dòng lệnh cho lớp Compiler. Như mọi người có thể thấy, chúng ta có thể thêm các trình tạo mã mới mà không bao giờ chạm vào, biên dịch lại hoặc làm theo bất kỳ cách nào khác mà tác động đến động cơ lõi.”

Jasmine nói: “Nhưng mà, Alphonse, đó không phải kiểu an toàn! Nếu cậu viết sai chính tả chuỗi thì sao? ”

“Các kiểm thử đơn vị của cô không có bắt được lỗi chính tả đó à?”

“Ừ…”

Jerry nói: “Tôi nghĩ quan điểm của Jasmine là chúng tôi muốn trình biên dịch kiểm tra việc này cho chúng tôi.”

“Tại sao?” Tôi hỏi lại “Hỏi lần nữa, cô sẽ không có kiểm thử đơn vị để đảm bảo rằng cô không sai chính tả?”

“Đúng thế, nhưng đây là một đối số dòng lệnh!”

“Phải, nhưng nếu cô viết sai chính tả một đối số dòng lệnh, chả lẽ cô không nghĩ tới việc gặp lỗi sao? Trong trường hợp này, công cụ Compiler có thể nói: ‘Ruby không phải là ngôn ngữ được hỗ trợ.’”

“Được rồi, tốt thôi, nhưng nếu cậu gọi Compiler từ một tập lệnh hoặc một chương trình khác, và có một lỗi chính tả trong một trong những chương trình đó. Trình biên dịch sẽ không cảnh báo cậu.”

“Đúng, nhưng một lần nữa, kiểm thử đơn vị của cô sẽ như thế.”

“Er…”

Điều này thật thú vị. Tôi dường như đã đụng phải một ý tưởng mà cả Jerry lẫn Jasmine đều không hề nghĩ tới. Thật vậy, tất cả người hướng dẫn trong phòng đều tỏ vẻ đăm chiêu trên mặt.

“Tại sao điều đó lại giống như một vấn đề vậy?” Tôi hỏi.

Jasper lên tiếng. “Chà, Fonze, er… Alphonse. Tôi không biết những người khác thế nào, nhưng cậu đã khiến tôi băn khoăn liệu kiểu an toàn có giá trị như chúng ta đang nghĩ hay không. Công cụ kiểm thử đơn vị này chúng tôi đã thực hiện trong vài năm qua có thể làm cho loại kiểu an toàn không cần thiết. ”

Tôi thấy một vài người gật đầu, và nhiều người khác lại lắc. Nhưng khi chúng tôi rời khỏi phòng, tâm trạng vẫn tư lự.