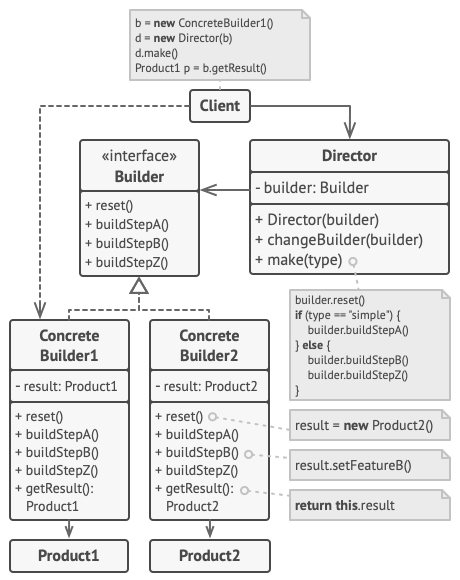
1. **Định nghĩa**

Builder là một pattern cho phép xây dựng một đối tượng phức tạp theo từng bước, việc xây dựng đối tượng này sẽ độc lập với các đối tượng khác

1. **Cấu trúc**

Bao gồm năm thành phần cơ bản là: **Builder**, **Concrete Builder**, **Product**, **Director** và **Client**.



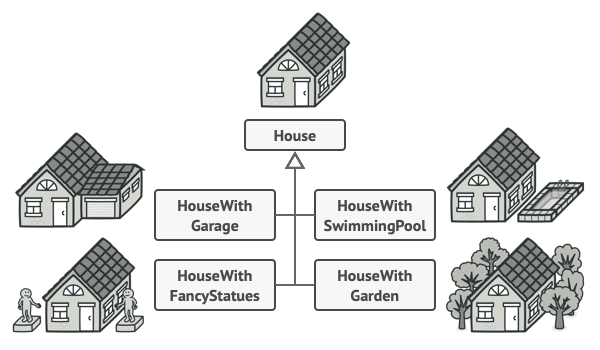
* Builder: Là thành phần định nghĩa một lớp trừu tượng
* Concrete Builder: Là thành phần triển khai
* Product: Là thành phần sẽ đại diện cho các đối tượng phức tạp phải tạo ra.
* Director: Là thành phần sẽ khởi tạo đối tượng Builder (Có thể có hoặc không).
* Client: Là thành phần sẽ sử dụng các ConcreteBuilder và Director.

1. **Hướng sử dụng**

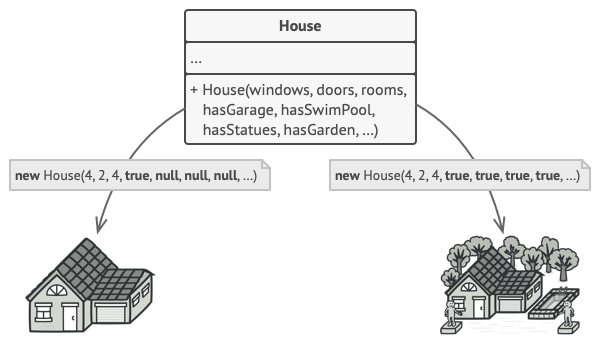
* Các class của bạn chứa quá nhiều hàm khởi tạo hoặc những hàm khởi tạo quá cồng kềnh và phức tạp.
* Gán giá trị cho các tham số của hàm khởi tạo không cần đúng theo thứ tự

1. **Bài toán**

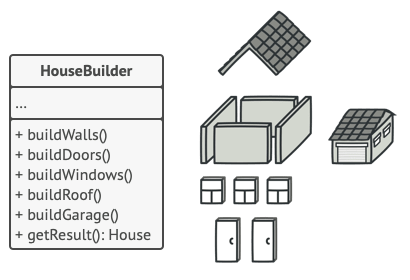
Bài toán xây dựng đối tượng là một ngôi nhà. Để xây dựng một ngôi nhà đơn giản, bạn cần xây dựng bốn bức tường và một tầng, lắp cửa ra vào, lắp một cặp cửa sổ và xây dựng một mái nhà. Nhưng nếu bạn muốn một ngôi nhà lớn hơn, sáng sủa hơn, có sân sau và các tiện ích khác (như hệ thống sưởi, hệ thống ống nước và hệ thống dây điện)?



Thông thường chúng ta sẽ tạo một hàm khởi tạo bao gồm tất cả các chi tiết khác của một ngôi nhà. Khi cần chi tiết nào ta sẽ truyền tham số cho nó, ngược lại thì truyền vào null.



Nhưng cách này quá cồng kềnh và dễ dàng truyền nhầm tham số, vì vậy chúng ta có thể sử dụng Builder pattern.



Với vấn đề đã đặt ra, chúng ta sẽ tiến hành xây dựng class HouseBuilder chứa tất cả các thuộc tính của class House và chúng ta sẽ chỉ sử dụng class Builder này để khởi tạo một đối tượng mới thay vì sử dụng hàm khởi tạo của class House như trước.

1. **Ưu và nhược điểm**
2. Ưu điểm

* Builder pattern giúp người dùng tránh được việc phải viết nhiều hàm khởi tạo cho class.
* **Không cần phải truyền giá trị null** cho các tham số mà đối tượng của họ không cần sử dụng tới.
* Builder pattern giúp code của chúng ta trở nên dễ đọc và dễ bảo trì hơn trong tương lai.

1. Nhược điểm

* Code có thể trở nên nhiều hơn và phức tạp hơn do đòi hỏi phải sử dụng nhiều class mới có thể cài đặt được pattern này.