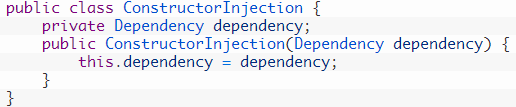
1. **Inversion of Control (IoC)**

* Là 1 design pattern được tạo ra để code tuân thủ Dependency Inversion
* Cung cấp 1 cơ chế đơn giản để chứa và quản lý các đối tượng phụ thuộc
* 1 Object bị phụ thuộc sẽ yêu cầu 1 số lượng nhất định các Object phụ thuộc được quản lý bởi IoC
* Ioc sẽ cung cấp các cách để Object bị phụ thuộc có thể truy cập và tương tác được với Object phụ thuộc
* Có 2 loại
  + **Dependency Injection**

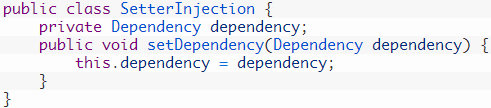
Đưa luôn Object phụ thuộc vào Object bị phụ thuộc

Có 2 cách

* + - **Constructor Injection**



* + - **Setter Injection**



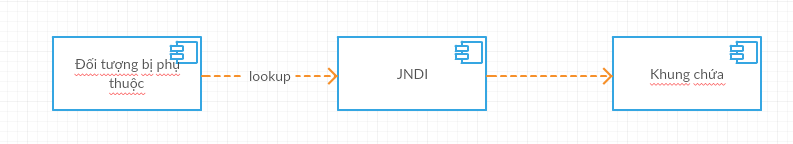
* + **Dependency Lookup**

Tìm kiếm các đối tượng phụ thuộc trong Ioc, sau đó dùng code để đưa đối tượng phụ thuộc vào trong đối tượng bị phụ thuộc

Có 2 loại

* + - **Dependency Pull**

Các Object phụ thuộc sẽ được lấy ra từ 1 nơi mà các Object phụ thuộc được đăng ký (không phải lấy từ khung chứa)



* + - **Contextualized Dependency Lookup (CDL)**

Các Object phụ thuộc sẽ được lấy ra trực tiếp từ khung chứa



* **Các cách để thực hiện**
  + Dependency Injection
  + ServiceLocator
  + Event
  + Delegate
* **Reference**

<https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring>

<https://huongdanjava.com/vi/inversion-control-va-dependency-injection.html>