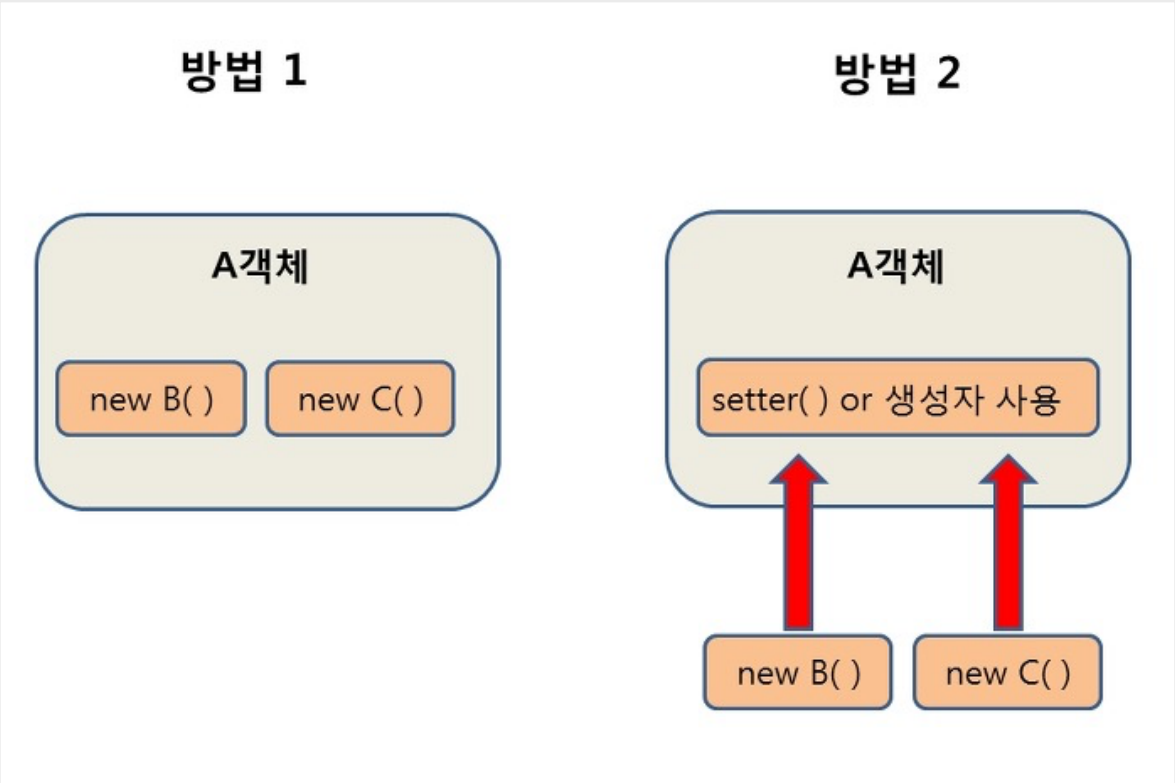
# [Spring] DI, IoC 정리

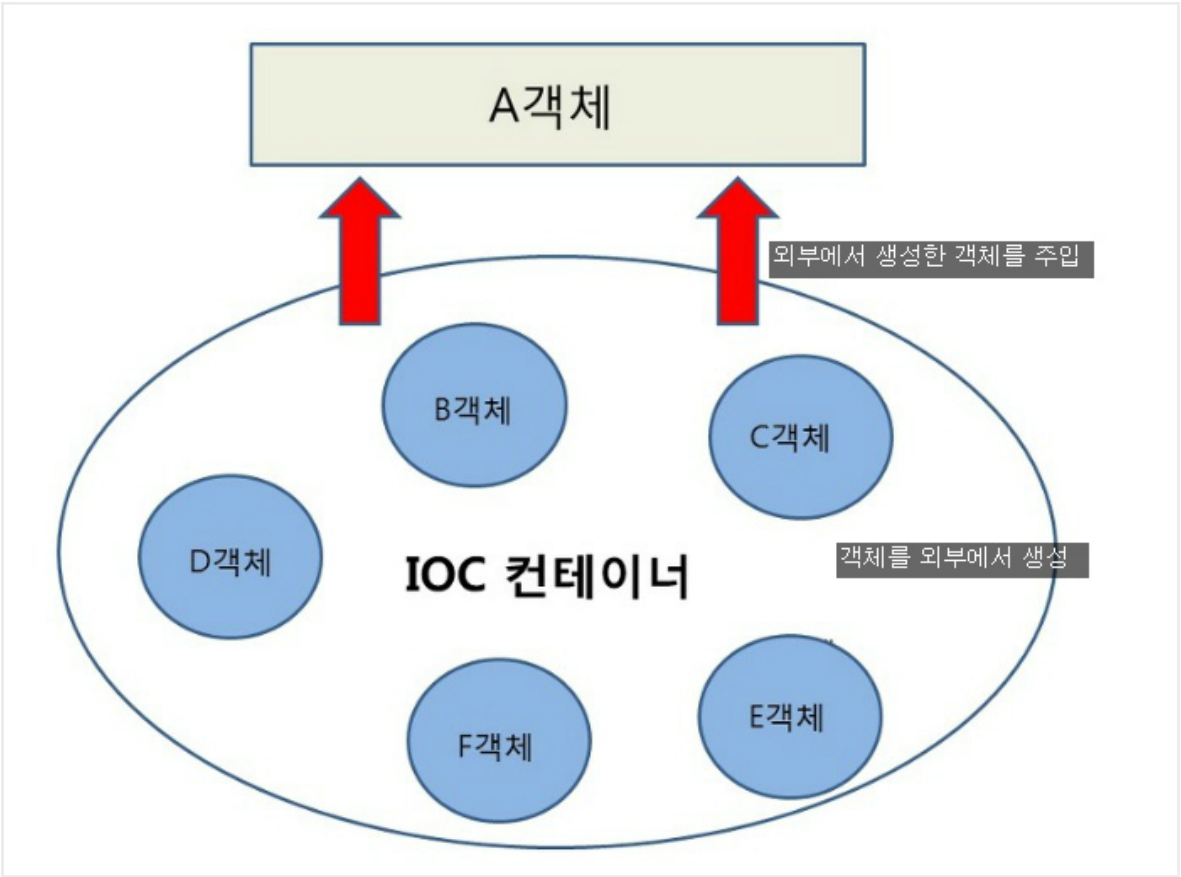
## DI(Dependency Injection)

DI(Dependency Injection) 이란 스프링이 다른 프레임워크와 차별화 되어 제공하는 의존 관계 주입 기능이다.
객체를 직접 생성해주는 것이 아닌 외부에서 생성을 한 후에 객체를 주입 시켜주는 방식이다.
DI(의존성 주입)를 통해서 모듈 간의 결합도가 낮아지고 유연성이 높아진다.




위의 그림에서 보여주는 [방법1]은 A객체가 B와 C 객체를 new 생성자를 통하여 직접 생성하는 방법이고
[방법2]는 외부에서 생성한 객체를 A객체에 주입하는 방식이다

위의 [방법2]가 의존성 주입의 예시이다.
***A 객체*** 에서 ***B,C 객체*** 를 사용(의존) 할때 ***A객체*** 에서 직접 생성하는 것이 아니라 외부(IOC컨테이너 에서 생성된 ***B,C 객체*** 를 조립(주입) 시켜 setter 또는 생성자를 통해 사용하는 방식이다.




스프링에서는 객체를 [Bean] 이라고 부르며, 프로젝트가 실행될 때 사용자가 Bean으로 관리하는 객체들의 생성과 소멸에 관련된 작업을 자동적을 수행해주는데 객체가 생성되는 곳을 스프링에서는 Bean 컨테이너라고 부른다.

## Ioc(Inversion of Control)

IoC(Inversion of Control) 란 "제어의 역전" 이라는 의미이다.
말 그대로 **메소드나 객체의 호출 작업을 개발자가 직접 결정하는 것이 아니라, 외부에서 결정되는 것을 의미**한다.
IoC는 제어의 역전이라고 말하며, 간단하게 말해서 "제어의 흐름을 바꾼다" 라고 한다.

객체의 의존성을 역전시켜 객체간의 결합도를 줄이고 유연한 코드를 작성할 수 있게 하여 가독성 및 코드 중복, 유지 보수를 편하게 할 수 있게 한다.

기존에 객체가 만들어지고 실행되었던 방식은
1. 객체생성
2. 의존성 객체 생성 ( 클래스 내부 생성)
3. 의존성 객체 메소드 호출

하지만, 스프링에서는 다음과 같은 순서로 객체가 만들어지고 실행된다.
1. 객체 생성

2. 의존성 객체 주입
   - 스스로 만드는 것이 아닌 제어권을 가진 개발자가 스프링에게 위임하여 스프링이 만들어 놓은 객체를 주입한다.
3. 의존성 객체 메소드 호출

스프링이 모든 의존성 객체를 스프링이 실행될때 다 만들어주고 필요한 곳에 주입 시켜 줌으로써 Bean(객체)들은 싱글턴 패턴의 특징을 가지며,
제어의 흐름을 사용자가 컨트롤하는 것이 아닌 스프링에게 맡겨서 작업을 처리하게 된다.