클래스, 객체, 인스턴스의 차이

클래스(Class) VS 객체(Object)

- 클래스는 '설계도', 객체는 '설계도로 구현한 모든 대상'을 의미한다.

객체(Object) VS 인스턴스(Instance)

- 클래스의 타입으로 선언되었을 때 객체라고 부르고, 그 객체가 메모리에 할당되어 실제 사용될 때 인스턴스라고 부른다.
- 객체는 현실 세계에 가깝고, 인스턴스는 소프트웨어 세계에 가깝다.
- 객체는 '실체', 인스턴스는 '관계'에 초점을 맞춘다.
  - 객체를 '클래스의 인스턴스'라고도 부른다.
- *'방금 인스턴스화하여 레퍼런스를 할당한' 객체를 인스턴스라고 말하지만, 이는 원본(추상적인 개념)으로부터 생성되었다는 것에 의미를 부여하는 것일 뿐 엄격하게 객체와 인스턴스를 나누긴 어렵다.*

실세계에 존재하거나, 실세계에서 생각할 수 있는 것(관념)

객체(Object) 란

- 개념
  - 소프트웨어 세계에 구현할 대상 (현실 세계에 가까운 측면)
  - 클래스에 선언된 모양 그대로 생성된 실체
- 특징
  - *'클래스의 인스턴스(instance)'*라고도 부른다.
  - 객체는 모든 인스턴스를 대표하는 포괄적인 의미를 갖는다.
  - oop의 관점에서 클래스의 타입으로 선언되었을 때 '객체'라고 부른다.

※ 메모리에 할당된 인스턴스의 이름을 불러주면, 그 인스턴스는 비로소 '객체'가 된다.

인스턴스(Instance) 란 ← 실제로 메모리에서 생성된 것

- 개념
  - 설계도를 바탕으로 소프트웨어 세계에 구현된 구체적인 실체 (소프트웨어 세계에 가까운 측면)
    - 즉, 객체를 소프트웨어에 실체화 하면 그것을 '인스턴스'라고 부른다.
    - 실체화된 인스턴스는 메모리에 할당된다.
- 특징
  - 인스턴스는 객체에 포함된다고 볼 수 있다.
  - oop의 관점에서 객체가 메모리에 할당되어 실제 사용될 때 '인스턴스'라고 부른다.
  - 추상적인 개념(또는 명세)과 구체적인 객체 사이의 **관계** 에 초점을 맞출 경우에 사용한다.
    - *'~의 인스턴스'*의 형태로 사용된다.
    - 객체는 클래스의 인스턴스다.
    - 객체 간의 링크는 클래스 간의 연관 관계의 인스턴스다.
    - 실행 프로세스는 프로그램의 인스턴스다.
  - 즉, 인스턴스라는 용어는 반드시 클래스와 객체 사이의 관계로 한정지어서 사용할 필요는 없다.
  - 인스턴스는 어떤 원본(추상적인 개념)으로부터 '생성된 복제본'을 의미한다.