JAVA는 객체지향언어 이므로 유지보수성이 높다.

1. 분리의 원칙(결합도는 낮추고 응집성은 높이고)
2. 중복 제거

Ex) Day2의 Student, Teacher, Employee의 공통점 -> 사람이다

( \_\_\_\_\_\_\_\_\_\_ is a \_\_\_\_\_\_\_\_\_\_\_\_\_ 원칙)

Ex) Student is a Person, Teacher is a Person…..

그럼 Person(사람)이 공통점이니 사람에 해당하는 부분을 따로 빼자!

부모의 생성자를 호출할 때

Super()

모든 생성자의 첫번째 줄에는 super(); 가 자동으로 생성

단, 생성자에 super 키워드가 없는 경우에만.

Super()는 항상 맨 첫줄에 와야함

\*메소드 오버라이딩

부모로부터 상속받은 메소드의 내용을 변경하는 것

1. 메소드 선언부는 변경 불가
2. 접근자는 같거나 크게 변경 가능 (private -> protected -> public)
3. Super().메소드이름이면 부모메소드를 가져와서 쓸 수 있다.

@ 애노테이션(annotation)

컴파일러한테 정보 제공

변수/클래스/메소드 앞에다가 final을 붙이면 상속불가가 된다.

Final 변수 값 변경 불가/클래스 상속 불가/메소드 오버라이딩 불가

추상(abstract)

접근 제어자 뒤에 abstract를 붙여서 설정한다.

Abstract를 붙인 메소드는 반드시 상속받은 클래스(자식)에서 구현하여야만 한다.

\*추상클래스를 사용하는 목적

1. 자식들이 공통적인 것을 가진다.

추상클래스 안에 있는 것들이 모두 추상이라서 구현이 안 되어있다면?

* Interface로 할 수 있다.

모든 클래스는 Object 를 상속한다.

* 다형성

자식의 주소를 부모타입으로 받는 것

여러 자손의 주소를 한번에 받기 위해서

자식의 모든 멤버를 사용하기 위해서는 형변환(Type Casting)을 해야 한다.