< 상속 >

1. 상속 (Inheritance)

- 목적 : 클래스의 재사용성, 연관된 일련의 클래스들에 대해 공통적인 규약을 정의
- 다른 클래스가 가지고 있는 멤버(필드와 메소드)들을 새로 작성할 클래스에서 직접 만들지 않고 상속을 받음으로써 새 클래스가 자신의 멤버처럼 사용할 수 있는 기능

➔ 보다 적은 양의 코드로 새로운 클래스 작성 가능

➔ 공통적 관리로 코드의 추가 및 변경에 용이

➔ 코드 중복제거를 통해 생산성과 유지보수 크게 기여

* 자바에서의 상속 처리 : 클래스간의 상속 시에는 **extends** 키워드 사용함

[public] class 클래스명 extends 클래스명 { }

상속 받는 클래스 상속 하는 클래스

자식/후손/파생/서브 부모/선조/상위/슈퍼

클래스 클래스

* 인터페이스 간의 상속 : **extends** 키워드 사용

[public] interface 인터페이스명 extends 부모인터페이스명 { }

* 클래스가 인터페이스를 상속받을 때 : **implements** 사용

[public] class 클래스명 implements 부모인터페이스명 { }

2. 상속의 특징

* 단일상속, 다중상속

- 클래스간 extends 상속은 단일상속만 허용, 다중상속 안됨
- 하지만 인터페이스 간 extends 상속은 다중상속을 허용
- 클래스가 인터페이스를 상속받는 implements 상속도 다중상속을 허용

* 부모클래스의 생성자, 초기화블럭은 상속 안됨

- 후손클래스 객체 생성시, 부모클래스 생성자가 먼저 실행되도록 되어있음 (기본구조)
- 후손클래스 생성자 안에 부모클래스 생성자 호출을 명시하고 싶으면 super(); 입력

* 부모의 private 멤버는 상속은 되나 후손도 접근 불가능

- 후손클래스 객체 생성시 부모의 필드값도 전달받은 경우 후손 생성자 안에서 부모의 private 필드에 직접 초기값 대입 못함

  → 전달받은 부모 필드 값을 부모 생성자 쪽으로 넘기는 방법 사용

  → super(전달값, 전달값….); // 부모의 매개변수 있는 생성자가 받아서 초기화작업

< super(), super(전달값, 전달값…) >

- 후손클래스 생성자에서 부모클래스 생성자를 호출할 때 사용
- 후손클래스 생성자에 명시되어있지 않지만 첫 줄에 super(); 가 존재

  → 후손클래스 객체 생성시 부모클래스 생성자가 먼저 실행되도록 되어있음

  → 부모클래스에 기본생성자 무조건 쓰기!!

< super. >

- 부모클래스의 주소값을 가리키는 레퍼런스
- 메소드, 생성자에 숨겨진 채로 존재하여 부모쪽에 접근하고 싶을 때 사용 가능

*** java.lang.Object 클래스가 최상위 클래스임**

- 자바에서 제공되는 클래스이든, 프로그래머가 만든 클래스이던 모두 다 Object 클래스의 후손으로 자동 처리

→ Object 클래스가 제공하는 메소드를 자신의 메소드 처럼 사용해도 됨

>> public boolean equals(비교할 레퍼런스) : 두 레퍼런스의 주소가 같은지 물어볼 때

>> public String toString() : 클래스명@16진수해시코드 를 리턴함

>> public int hashCode() : 할당된 객체의 위치 정보를 십진수 해시코드 값으로 리턴함

3. 오버라이딩(Overriding)

- 후손클래스가 상속받은 부모 메소드를 재정의하는 것

  → 부모가 제공하는 기능을 후손이 일부 고쳐 쓰겠다는 의미

- 후손 객체를 통한 메소드 호출 시 후손 것이 우선권을 가짐

  → 부모메소드는 자동 은닉됨

- Annotation : 자바 컴파일러에게 알리는 주석문

  @Diprecated, **@Override**,

  @Supress Warning("무시할 경고") 무시할 경고문을 입력하면 무시가능

*** 오버라이딩 시 접근제어자 수정 가능함**

→ 단, 부모 메소드 접근제어자보다 같거나 넓은 범위로 변경 가능

*** 오버라이딩 시 부모 메소드의 예외처리 클래스 개수보다 같거나 줄일 수 있음**

*** 모든 클래스가 java.lang.Object 클래스의 후손으로 자동 상속되어있음**

→ Object 클래스가 제공하고 있는 메소드를 오버라이딩 해서 메소드 본래 기능을 변형 할 수 있음

ex) java.lang.String 클래스의 equals() 메소드가 오버라이딩 되어 두 객체가 가진 값이 일치하는지 비교하도록 변경되어 있음

toString()도 오버라이딩 해서 문자열 값 리턴하는 기능으로 변경하였음