

오전

상속 이어서

상속은 필드과 메소드 모두 물려받아 사용한다.

(코드의 재사용성 증가 / 길이 감소 / 유지보수 용이)

상속받은 자식들에 대한 공통적인 규약 정의할 수 있다

상속 받은 자식들은 모두 부모와 같은 필드 메소드르 가지고 있음으로 클래스들이 일부 공통된 형태를 띈다

**super()** 생성자

부모의 생성자를 호출하는 코드

반드시 자식생성자에 최상단에 작성되어야한다.

생성자

객체가 생성될 때 필드 초기화 + 특정 기능을 수행

부모 클래스에서

**private**일 경우 자식 클래스에서도 직접접근 할 수 없다.

**setter**을 사용해도 접근 가능하지만 비효율적이다.

**super**( 전달 할 값 )

//1) 자식 내 부모 객체 생성

//2) 부모 객체 생성 시 초기화

오버라이딩 : 재정의 하는 것.

자식 클래스가 부모에게 받은 메소드를 재작성 하는 것

**@Overriding**


\*\*\*오버라이딩 성립 조건\*\*\*

1. 메소드 이름이 동일해야 한다
2. 반환형이 동일해야 한다
3. 매개변수 (있는 경우 똑같아야한다)
4. 접근제한자 같거나 더 넓은 범위이어야 한다.
  1. **ex) protected >** 오버라이딩 하고 싶은 경우 **protected or public**이어야 한다.
5. 예외처리 범위는 같거나 더 좁게

@ 부모의**private** 메서드는 오버라이딩이 불가

@ -> 자식이 접근 할 수 없기때문에

**Object**




**OverRiding**하면 오버라이딩한 클래스가 표시됨


자식이 상속 받으면 받을 수록 사이즈가 커진다.

**extends** 확장 되기때문에.

자바는 단일 상속 : **extends** 옆에는 하나의 부모만 가능.

오버 로딩과 오버라이딩의 차이





**Final**을 클래스나 메소드에 활용할 경우.

**final** 클래스 상속 불가능하다 : 상속 불가 클래스


**The type Child cannot subclass the final class Parent**

마지막 클래스라는 의미로 더 이상 상속이 불가를 뜻함.

**final method** : **final**이후로는 재정의 할 수 없다.

**Cannot override the final method from Parent**

## 다형성

 09\_다형성(Polymorphism).pdf 572 kB

객체 하나가 다양한 모양을 가진다.

※상속에 기반하는 지식이다.

상속 + 연산 규칙 + 얇은 복사

객체지향의 4대 특징중 하나

부모클래스 변수명 = new 자식 클래스 ();

**Car c = new Sonanta();**

//원래라면 다른 자료형이기때문에 연산이 안된다

///하지만 이걸 되게하는게 '상속'

Tesla 객체는 Car형태로도 변할 수 있다

Tesla가 두가지 자료형을 가지고 있다.

- 이런것들을 다형성 이라한다
- 한가지 객체가 여러 모양을 가지는 느낌.

## 다형성 (업캐스팅)의 사용법

1. 자식 객체가 부모 객체로 변하였기 때문에 자식만의 고유한 필드 필드 메소드를 사용할 수 없다



2. 다형성을 이용한 객체 배열

1. 객체 배열 : 같은 객체 참조 자료형의 변수를 하나의 묶음으로 다루는 것.
  1. 다형성 적용 > 부모타입 참조 자료형의 변수를 하나의 묶음으로 다루는 것.



배열 하나에 서로 다른 자료형 3개가대입했다.

3)다형성을 이용한 매개변수 사용방법

4)다형성을 이용한 반환형 사용법

instance of 연산자 : 객체의 자료형을 검사하는 연산자.

참조하는 객체가 특정 자료형이거나 부모쪽 상속 관계인지를 확인.

arr[1] instanceof Tesla

## DownCasting

Car car2 = new Tesla();

(Tesla)car2 == 강제형 변환을 통해 다운 캐스팅한다

※ 다만 형 변환을 할때 변환 받는 객체의 소속을 정확히 해야한다.

주말에 다형성 바인딩 공부.