1. protected 접근 제한자

* 같은 패키지내의 클래스나 상속받은 자식 클래스에서 사용할 수 있는 필드나 메소드를 선언할 때 사용
* 접근 제한자의 범위

pubic > protected > default > private

1. 오버로딩

* 매개변수의 개수나 타입을 다르게 하여 같은 이름의 메소드를 새롭게 생성

1. 오버라이딩(메소드 재정의)

* 상속받은 자식클래스에서 부모클래스에 존재하는 메소드를 재정의
* 부모클래스의 메소드와 오버라이딩 메소드의 이름과 형태가 동일해야 됨
* 상속이 필수적인 조건

1. super 키워드

* 상속받은 자식클래스 내에서 부모클래스에 접근할 때 사용
* 자식클래스에서 부모클래스의 생성자나 메소드를 사용할 때 super 키워드 사용
* 자식클래스의 생성자에는 부모클래스의 생성자가 포함되어 있음

super.부모생성자() 형태로 생략되어 있음

* 오버라이딩으로 재정의된 메소드에서 부모클래스의 원본 메소드를 사용하고 싶을 때

주로 사용

1. 다형성

* 기본 성질은 동일하지만 각각의 객체의 특색있는 성질나 동작을 가지는 성질

ex) 티비 ----------------- 가격

전원켜기

전원끄기

삼성티비 티비상속 ----------------- 빅스비

빅스비 켜기

빅스비 끄기

엘지티비 티비상속 -------------------- 구글어시턴스

구글어시스턴스 켜기

구글어시스턴스 끄기

* 프로그래밍에서의 다형성은 상속받은 부모클래스에서 공통되는 성질과 동작을 가지고

자식클래스만의 고유의 성질과 동작을 만드는 행위

1. 상속에서의 타입변환

* 상속받은 자식클래스는 본인의 형태도 가지면서 부모클래스의 형태도 가지기 때문에

부모클래스로의 형변환 가능

* 부모클래스로 형변환을 한 경우 자식클래스에 있는 필드나 메소드는 사용 불가능

1. 강제 타입변환

* 부모객체가 자식객체의 형태로 변환되는 것
* 부모객체가 생성될 때 자식객체가 삽입된 경우에만 강제 타입변환 가능

1. 객체 타입 확인

* instanceof 라는 키워드를 사용하여 진행
* 결과 값은 boolean값으로 리턴
* boolean result = 타입을 확인할 객체의 이름 instanceof 클래스명

1. 타입변환의 사용 목적

* 필드객체를 생성할 때 부모객체로 선언하게 되면 다양한 자식객체들을 담을 수 있기 때문에 많은 결과 값 창출 가능

1. 추상클래스

* 여러 개의 클래스들의 공통된 필드랑 메소드만 모아서 관리하는 클래스
* 삼성티비 <---------------- 티비(추상클래스) ----------🡪 엘지티비

삼성티비와 엘지티비에 공통된 필드와 메소드를 티비라는 추상클래스 관리

* 추상클래스와 일반클래스는 상속관계로 묶이게 됨
* 추상클래스를 사용하는 이유

공통으로 사용하는 변수명이나 메소드명을 통일할 수 있다.

공통되는 코드가 추상클래스에서 관리되기 때문에 소스코드 작성 시간 단축

* 추상클래스의 선언은 abstract 키워드 사용

public abstract 클래스명 {

}

* 추상클래스는 객체 생성 불가능

1. 추상메소드

* 자식객체에서 공통적으로 가지고 있는 메소드중 서로 동작이 다른 메소드
* 추상클래스안에 추상메소드로 선언
* 선언부만 추상클래스에 존재하고 자식클래스에서 정의한다.
* 추상메소드는 자식클래스에서 반드시 구현
* 추상메소드를 포함하는 클래스 반드시 추상클래스여야 함