-상속 : 권한이 허용하는 모든 것을 물려받음(가져다 쓸 수 있음)  
1개의 부모만 존재. 1:1 관계(자식입장)  
 \*E의 경우  
 부모클래스가 2개가  
 올 수 없으므로  
 둘 중 하나는 끊어야 함  
  
  
  
자바에서는 다중상속을 허용하지 않음.  
  
접근권한 class 클래스명 extends 부모클래스명 { **// 1:1 관계이므로 하나만 쓸 수 있음**  
 내용  
}

-오버라이딩 : 부모에 존재하는 메소드를 자식에서 재정의하는 것. 상속과 관계 있음.  
 덮어쓴다는 개념(update)

-오버로딩 : 이름만 동일한 메소드를 여러 개 생성(나머지는 다름). 상속과 관계 없음.  
 ex) println(int, string, long…….)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **위치** | **기능** | **조건** |
| **오버라이딩** | 상속관계 | 재정의 (update) | 반환타입, 메소드명, 인자타입, 인자개수 동일. 접근권한은 넓어질 수 있음. 예외처리는 좁아질 수 있음. |
| **오버로딩** | 동클래스 내부 | 신규 (new) | 메소드명은 동일. 인자개수, 인자타입이 틀려야 함. |

-클래스A, 클래스 B. 클래스 B는 A를 상속받은 상태.  
 이 때 객체 생성 가능 여부.  
1) A a1 = new A(); - 정상적으로 객체 생성 가능  
  
  
  
2) B b1 = new B(); - 정상적으로 객체 생성 가능  
  
  
  
3) A a2 = new B(); - 정상적으로 객체 생성 가능. (핸드폰 개통 시, 배터리를 추가로 받았다고 가정. 추가 배터리를 쓰지 않더라도 기능상의 문제는 없음)  
  
  
  
4) B b2 = new A(); - 에러 발생. (핸드폰 개통 시 배터리가 포함되지 않은채로 물건을 받았다고 가정. 배터리 없이는 작동이 안됨.)  
  
  
  
객체 생성을 위해서는 최소한 완제품은 와야함. 일부만 오면 생성되지 않음.  
  
-Object(java.lang.object) : 자바의 최상위 클래스. 별도의 선언이 없는 한 자동 상속  
 object에는 어떠한 데이터도 담을 수 있음(정해진 형태 X)  
 공간적인 효율 좋지 않음.   
 하지만 사용자가 어떤 값을 입력할지 모를 때 사용함.  
 ex) Object d = “abc”; // 위의 3번 그림 생각하면 됨.  
\*클래스간 형변환 -> 캐스팅 이용. 관계만 성립하는 상황에만 가능함.  
(클래스명)변수명  
  
-인터페이스 : 틀. 개발의 형태를 제공하기 위해 사용.  
 기본적으로 내용이 없이 선언만 가능. 자바 1.8이상부터는 내용도 들어감.  
 형태를 일치시켜야 할 경우(협업 시)  
  
 접근권한 interface 인터페이스명 {  
 내용 \*내용의 메소드는 메소드 내용을 가질 수 없음. () x

}

-참조 : 인터페이스를 클래스에서 사용하기 위해 사용. 1:N관계  
 (내용이 없기 때문에 중복되어도 상관없음)

접근권한 class 클래스명 impliments 인터페이스명, …. {  
 내용  
 }  
\*주의사항 : 참조시 반드시 인터페이스의 메소드를 오버라이딩함.

-추상클래스(abstract class) : 일부 미완성 형태로 구현된 클래스.[지폐 앞면 완성, 지폐 뒷면 비어있음]  
 객체 생성 불가능. 상속을 통해 사용. 추상메소드를 오버라이딩해서 사용.  
 반제품. diy제품. 확장성.  
 공통적 영역 + 추가적영역  
  
 접근권한 abstract class 클래스명 {  
 내용 → 추상메소드(내용이 없는 메소드)의 경우 내용은 아래와 같이 입력  
 접근권한 abstract 반환타입 메소드명(인자타입 인자명, .…); // 내용 X  
  
  
 }