클래스의 상속 2

클래스의 상속 2

- 상속의 기본 조건 : 'is a kind of' 관계
  - 하위 클래스는 상위 클래스의 모든 특성 보유
  - 하위 클래스는 자신만의 추가적인 특성 확장
  - 휴대폰(부모 클래스)와 스마트폰(자식 클래스)으로 본 상속의 조건

```
public class MobilePhone {
    protected String number;
    public MobilePhone(String num) {
        this.number = num;
    }
    public void answer() {
        System.out.println("PhoneNumber : " + this.number);
    }
}
```

클래스의 상속 2

```
public class SmartPhone extends MobilePhone {
    private String androidVer;
    public SmartPhone(String num, String androidVer) {
        super(num);
        this.androidVer = androidVer;
    }
    public void playApp() {
        System.out.println("안드로이드 버전 : " + this.androidVer);
    }
}
```

```
public class MobileSmartPhone {
    public static void main(String[] args) {
        SmartPhone phone = new SmartPhone("010-2511-7635", "정민욱");
        phone.answer();
        phone.playApp();
    }
}
```

클래스의 상속 2

- 상위 클래스의 참조 변수가 참조할 수 있는 대상의 범위
  - 스마트폰 클래스의 객체 생성 : SmartPhone phone = new SmartPhone();
  - 상위 클래스의 참조변수는 하위 클래스의 인스턴스 참조 가능
    - 스마트폰은 일종의 휴대폰 : '스마트폰을 가리키며 휴대폰이다'라고 말할 수 있음
    -> 휴대폰 클래스을 상속하는 스마트폰 인스턴스는 휴대폰 클래스의 인스턴스 이기도 함
    - 따라서 휴대폰 클래스의 참조 변수는 스마트폰 인스턴스 참조 가능
    - 'new SmartPhone()' 으로 생성된 인스턴스는 스마트폰의 인스턴스이며 동시에 휴대폰의 인스턴스
    - MobilePhone phone = new SamrtPhone(); 가능
    - 단, 위의 인스턴스 phone은 playApp() 메소드를 사용할 수 없음
      a. 실행 시간을 늦추는 결과 발생
      b. 참조변수의 형을 기준으로 접근 가능한 멤버를 제한하는 것은 코드를 단순하게 함

클래스의 상속 2

```
public class MobileSmartPhone {
    public static void main(String[] args) {
        SmartPhone phone1 = new SmartPhone("010-2511-7635", "정민욱");
        MobilePhone phone2 = new SmartPhone("010-2511-7636", "정민이");
        phone1.answer();
        phone1.playApp();
        phone2.answer();
        phone2.playApp();
    }
}
```

클래스의 상속 2

- 메소드 오버라이딩
  - 상위 클래스에 정의된 메소드를 하위 클래스에서 다시 정의
  - 메소드의 이름, 메소드의 반환형, 메소드의 매개변수가 모두 같아야 오버라이딩이 성립
  - 오버라이딩 되는 메소드에는 '@Override' 어노테이션을 첨부하여 오버라이딩된 메소드임을 표시하는 것을 추천

```
public class Cake {
    public void yummy() { System.out.println("Yummy Cake"); }
}
```

```
public class CheeseCake extends Cake {
    @Override
    public void yummy() { System.out.println("Yummy Cheese Cake"); }
}
```

```
public class YummyCakeOverriding {
    public static void main(String[] args) {
        Cake c1 = new CheeseCake();
        CheeseCake c2 = new CheeseCake();
        c1.yummy();
        c2.yummy();
    }
}
```