추상클래스와 인터페이스

추상클래스

추상클래스

추상클래스(abstract class)

- 몸체가 없는 메소드(추상메소드)를 포함한 클래스
- 추상클래스일 경우 선언부에 abstract 키워드를 사용

```
[접근제어지시자] abstract class 클래스명{
}
```

```
public abstract class TV{
    private boolean power;
    public abstract void powerOn();
}
```

추상클래스

추상메소드(abstract method)

- 몸체({})가 없는 메소드(추상메소드)
- 메소드의 선언부만 있고 실행코드가 없는 메소드
- 추상클래스를 상속한 경우 반드시 구현 해야하는 메소드(오버라이딩 강제화)

```
[접근제어지시자] abstract class 클래스명{
}
```

```
public abstract class TV{
    private boolean power;
    public abstract void powerOn();
}
```

추상클래스

추상클래스의 특징

1. 미완성 클래스(abstract 키워드 사용)로 자체적으로 객체 생성 불가
   → 반드시 상속하여 객체를 생성

2. abstract 메소드가 포함된 클래스 → 반드시 abstract 클래스
   단, abstract메소드가 없어도 abstract 클래스 선언이 가능(객체생성방지용)

3. Abstract메소드 외에 일반 메소드와 변수를 포함할 수 있음

4. 객체 생성은 안되나, 참조형 변수 type으로는 사용 가능(다형성으로는 가능)

인터페이스

인터페이스

인터페이스(interface)

- 상수와 abstract메소드만으로 이루어진 추상클래스의 변형체
- 메소드의 통일성을 부여하기 위해서 abstract 메소드만 따로 모아 놓은 것으로, 상속 시 인터페이스에 정의된 모든 메소드를 구현해야 함

[접근제어지시자] interface 인터페이스명{
}

```
public interface TV{
    public static final boolean power=true;//상수형변수만 가능
    public void powerOn();
    //모든 메소드가 abstract메소드이므로 abstract생략이 가능
}
```

인터페이스

인터페이스의 특징

- 인터페이스의 모든 메소드는 묵시적으로 public이고 abstract이다.
- 변수는 public static final이다.
- 객체 생성은 안되나, 참조형 변수로서는 가능하다.(다형성으로는 가능)

인터페이스의 장점

1. 상위 타입의 역할로 다형성을 지원하여 연결해주는 역할 수행
2. 해당 객체가 다양한 기능을 제공시에도 인터페이스에만 해당하는 기능만을 사용하도록 제한 가능
3. 공통 기능상의 일관성 제공 / 공동작업을 위한 가이드라인 제공

추상클래스/인터페이스

추상클래스 VS 인터페이스

구분	인터페이스	추상클래스
상속	다중상속	단일상속
구현	implements 사용	extends 사용
추상메소드	모든 메소드	추상메소드 0개 이상
abstract	묵시적으로 abstract	명시적으로 사용
객체	객체 생성불가	객체 생성불가