# ( 디지털 컨버전스 ) 스마트 콘텐츠와 웹 융합 응용 SW개발자 양성과정

훈련기간 : 2021.05.07 ~ 2021.12.08

KH정보교육원 | 전승리

```java
class A {
    int a = 10;

    void b () {
        System.out.println("A");
    }
}

class AA extends A {
    int a = 20;

    void b () {
        System.out.println("AA");
    }
    void c () {
        System.out.println("C");
    }
}
```

```java
public class ExtendsTest {
    public static void main(String[] args) {
        A a = new A();
        a.b();
        System.out.println("A a: " + a.a);

        AA aa = new AA();
        aa.b();
        aa.c();
        System.out.println("AA aa: " + aa.a);

        A a1 = new AA();
        a1.b();
        System.out.println("A a1: " + a1.a);
    }
}
```

출력 값

```
A
A a: 10
AA
C
AA aa: 20
AA
A a1: 10
```

A a1 = new의 대상은 AA()이다.
접근 데이터는 데이터타입 A를 참조해야한다. 상속관계여야만 선언 가능

상속의 개념

선언 방법 : class 자식클래스명 extends 부모클래스명{}

상속 : 말 그대로 재산을 물려 받는것이다.

자바의 경우 클래스는 단일 상속만 가능하다.

다중 상속시 동일 메서드명이 존재하면 충돌이 발생하기 때문.

부모클래스

```java
class Vehicle {
    private float rpm;
    private float fuel;
    private float pressure;

    public Vehicle(float rpm, float fuel, float pressure) {
        this.rpm = rpm;
        this.fuel = fuel;
        this.pressure = pressure;
    }

    @Override
    public String toString() {
        return "Vehicle{" +
                "rpm=" + rpm +
                ", fuel=" + fuel +
                ", pressure=" + pressure +
                '}';
    }
}
```

자식클래스

```java
class Airplane extends Vehicle {
    private float aileron;


    public Airplane(float rpm, float fuel, float pressure,
                    float aileron) {
        super(rpm, fuel, pressure);

        this.aileron = aileron;

    }

    @Override
    public String toString() {
        return "Airplane{" +

                "super.Vehicle()=" + super.toString() +
                ", aileron=" + aileron +
                '}';
    }
}
```

super()는 무엇이 되었든 상속자인 부모를 호출한다.
super()만 적혀 있으니 생성자를 호출하게 된다.

아래와 같이 받은 부모클래스의 iv를 그대로 사용할수있다.

```java
public class InheritanceWithSuperTest {
    public static void main(String[] args) {
        Vehicle v = new Vehicle( rpm: 200,  fuel: 1.2f,  pressure: 1.0f);

        System.out.println(v);

        Airplane a = new Airplane( rpm: 1000,  fuel: 112.5f,  pressure: 12.3f,  aileron: 77.3f);

        System.out.println(a);
    }
}
```

```java
interface Remocon {
    public void turnOn();

    public void turnOff();
}
```

인터페이스 작성법
1. 일단 interface를 적는다.
2. 인터페이스명(일종의 클래스 같은 것이라고 보면 됨)을 적는다.
3. 인터페이스 내부에는 매서드 프로토타입을 작성한다.
(프로토타입 : 매서드의 접근 제한자, 리턴 타입, 매서드 이름, 입력등을 기록한 형태)

```java
class Lamp {
    LampMethod lamp = new LampMethod() {
        @Override
        public void lightOn() {
            System.out.println("Lamp를 킵니다.");
        }

        @Override
        public void lightOff() {
            System.out.println("Lamp를 끕니다.");
        }
    };
}
```

```java
public class InterfaceTest2 {
    public static void main(String[] args) {
        Lamp lamp = new Lamp();

        lamp.lamp.lightOn();
        lamp.lamp.lightOff();

    }
}
```

인터페이스

목적 : 동일한 목적 하에 동일한 기능을 보장하게 하기 위함
인터페이스란?
인터페이스는 추상 메서드의 집합이다.
인터페이스에 정의된 추상 메서드를 완성하는 것을 구현(implements)이라한다.

특징 :
1. 다중 상속이 가능하다.{}이 없기때문에 클래스처럼 충돌이 발생하지 않는다.
2. 구현된 것이 없는 설계도, 껍데기라고 할수있다. 모든 멤버가 public이다.
3. 추상메서드,상수는 가능하지만 인스턴스 메서드 iv 사용 불가.