(디지털컨버전스) 스마트 콘텐츠와 웹 융합 응용 SW개발자 양성과정

-17일차 학습 및 질문 노트-

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – Kyeonghwan Lee(이경환)
airtrade7@naver.com

▣extends

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

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

// extends 키워드가 바로 상속!
// 상속: 말 그대로 재산을 물려 받는것이다.
//       클래스의 내용물들을 활용할 수 있게 된다.
class AA extends A {
    int a = 20;

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

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);

        // new의 대상은 AA() 이며
        // 접근 데이터는 데이터타입 A를 참조해야한다.
        A a1 = new AA();
        a1.b();
        System.out.println("A a1: " + a1.a);
    }
}
```

extends 작성법
class 자식 클래스 extends 부모 클래스

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

Process finished with exit code 0
```

▣extends2

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

    public Vehicle(float rpm, float fuel, float pressure, String color) {
        this.rpm = rpm;
        this.fuel = fuel;
        this.pressure = pressure;
        this.color = color;
    }
    @Override
    public String toString() {
        return "Vehicle{" +
                "rpm=" + rpm +
                ", fuel=" + fuel +
                ", pressure=" + pressure +
                ", color='" + color + '\'' +
                '}';
    }
}
class Airplane extends Vehicle {
    private float aileron;
    private float pitch;
    private float rudder;

    public Airplane(float rpm, float fuel, float pressure, String color,
                    float aileron, float pitch, float rudder) {
        // super()는 무엇이 되었든 상속자인 부모를 호출한다.
        // super()만 적혀 있으니 생성자를 호출하게 된다.
        super(rpm, fuel, pressure, color);

        this.aileron = aileron;
        this.pitch = pitch;
        this.rudder = rudder;
    }
```

```java
    @Override
    public String toString() {
        return "Airplane{" +
                // super.toString()은 부모 클래스의 toString()을 호출한 것이다.
                "super.Vehicle()=" + super.toString() +
                ", aileron=" + aileron +
                ", pitch=" + pitch +
                ", rudder=" + rudder +
                '}';
    }
}

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

        System.out.println(v);

        Airplane a = new Airplane(
                rpm: 1000,  fuel: 112.5f,  pressure: 12.3f,  color: "White",
                aileron: 77.3f,  pitch: 0.02f,  rudder: 33.9f);

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

```
Vehicle{rpm=200.0, fuel=1.2, pressure=1.0, color='Red'}
Airplane{super.Vehicle()=Vehicle{rpm=1000.0, fuel=112.5, pressure=12.3, color='White'}, aileron=77.3, pitch=0.02, rudder=33.9}

Process finished with exit code 0
```

extends(상속)

- 상위 개념의 특징을 하위 개념이 물려 받는 것
- 하나의 부모 클래스는 여러 개의 자식 클래스를 가질 수 있다.
- 반대로 하나의 클래스는 여러 개의 클래스로부터 상속을 받을 수는 없다.(즉 자식은 여러 명이 될 수 있지만 부모가 여러 명일 수는 없다)
- 부모 클래스로부터 상속받은 자식클래스는 부모 클래스의 자원 모두를 사용할 수 있으나 부분적으로 선택해서 사용하지는 못한다.
- 자식클래스는 부모클래스로부터 물려받은 자원을 override 하여 수정해서 사용 할 수 있다.
- 부모 클래스가 상속받은 자원도 자식클래스가 사용 가능합니다.
- Super()는 무엇이 되었든 상속자인 부모를 호출한다.

▣interface

```java
interface Remocon {
    public void turnOn();
    public void turnOff();
}
class AbstractTest {
    Remocon rc = new Remocon() {
        @Override
        public void turnOn() {
            // 여기에 필요한 기능은 필요한 사람이 알아서 만드세요 ~
            System.out.println("나는 RC 자동차용 리모콘이야! RF 송수신기가 지금 활성화되었어!");
        }

        @Override
        public void turnOff() {
            System.out.println("이제 헤어질 시간이야! RF 송수신기 신호 출력을 차단할게!");
        }
    };
    Remocon radio = new Remocon() {
        @Override
        public void turnOn() {
            System.out.println("나는 라디오야! 지금부터 주파수 채널 매칭을 시작할게!");
        }

        @Override
        public void turnOff() {
            System.out.println("이젠 안녕! 주파수 채널 매칭을 끊을게!");
        }
    };
    public void testMethod () {
        Remocon tv = new Remocon() {
            @Override
            public void turnOn() {
                System.out.println("나는 TV야! AM/FM 신호를 수신할게! 이제부터 방송을 보자!");
            }
            @Override
            public void turnOff() {
                System.out.println("AM/FM 신호를 차단할게! 내일 또 보자!");
            }
        };
        tv.turnOn();
        radio.turnOff();
    }
    public void testMethod2 () {
        rc.turnOn();
        radio.turnOff();
    }
}

public class InterfaceTest {
    public static void main(String[] args) {
        AbstractTest at = new AbstractTest();

        at.testMethod();
        at.testMethod2();
    }
}
```

```
나는 TV야! AM/FM 신호를 수신할게! 이제부터 방송을 보자!
이젠 안녕! 주파수 채널 매칭을 끊을게!
나는 RC 자동차용 리모콘이야! RF 송수신기가 지금 활성화되었어!
이젠 안녕! 주파수 채널 매칭을 끊을게!

Process finished with exit code 0
```

Interface작성법

1. 일단 interface를 적는다.
2. 인터페이스명(일정의 클래스 같은 것이라고 보면 됨)을 적는다.
3. 인터페이스 내부에는 매서드 프로토 타입을 작성한다.(프로토 타입이란 매서드의 접근 제한자, 리턴 타입, 매서드 이름 , 입력 등을 기록한 형태)

▣interface2

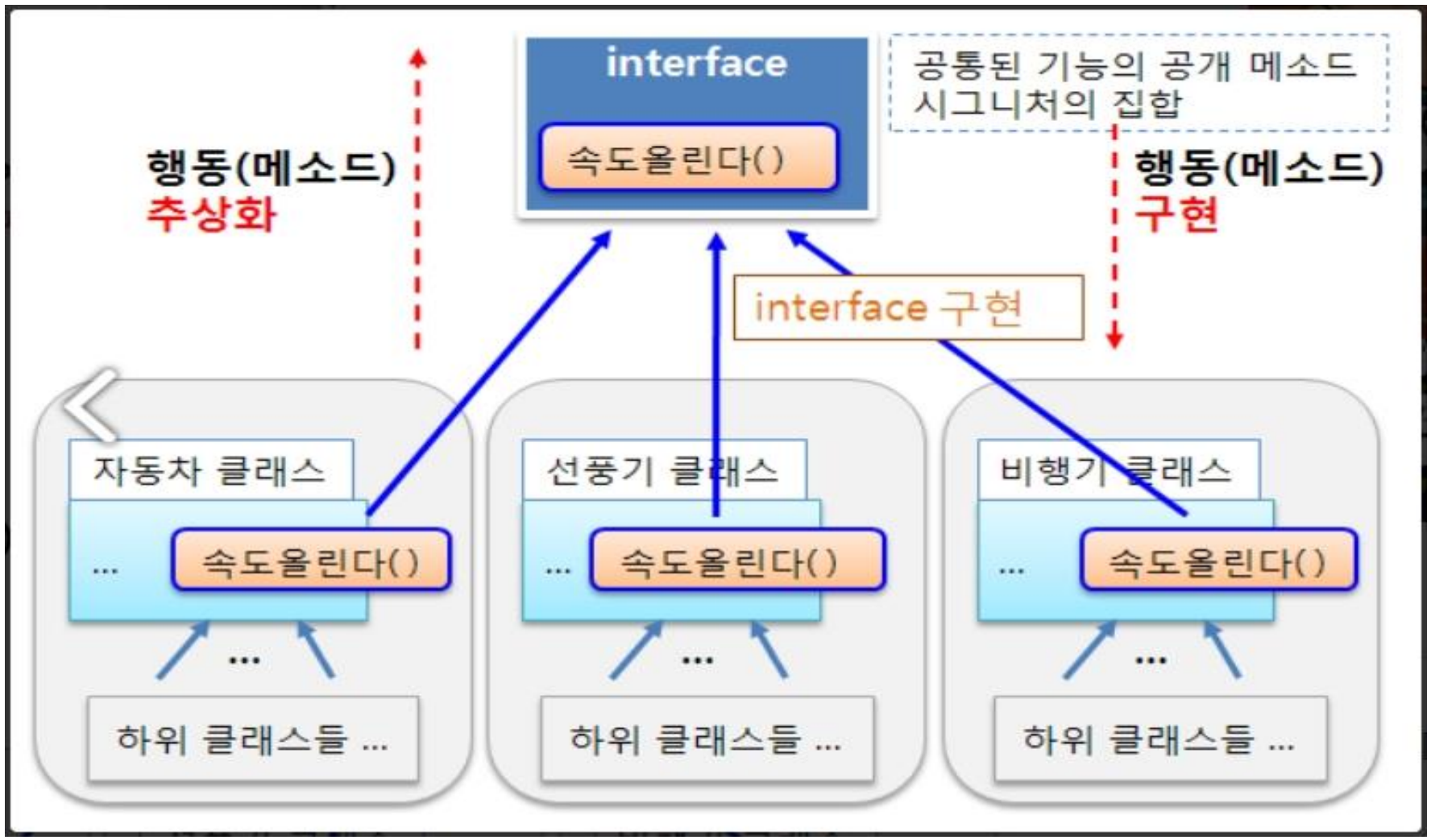
```java
class 이건희 {
    BigInteger money;
    String company;

    public 이건희(BigInteger money, String company) {
        this.money = money;
        this.company = company;
    }

    @Override
    public String toString() {
        return "이건희{" +
                "money=" + money +
                ", company='" + company + '\'' +
                '}';
    }
}
class 이재용 extends 이건희 {
    String recentInvest;

    public 이재용(BigInteger money, String company, String recentInvest) {
        super(money, company);
        this.recentInvest = recentInvest;
    }

    @Override
    public String toString() {
        return "이재용{" +
                "money=" + money +
                ", company='" + company + '\'' +
                ", recentInvest='" + recentInvest + '\'' +
                '}';
    }
}
```

```java
public class Prob52 {
    public static void main(String[] args) {
        BigInteger bigNum = new BigInteger( val: "100000000000000");
        이건희 samsung = new 이건희(bigNum, company: "삼성그룹");

        이재용 newSamsung = new 이재용(bigNum.multiply(BigInteger.TEN),
                company: "삼성전자", recentInvest: "바이오 사업");

        System.out.println(samsung);
        System.out.println(newSamsung);
    }
}
```

```
이건희{money=100000000000000, company='삼성그룹'}
이재용{money=1000000000000000, company='삼성전자', recentInvest='바이오 사업'}


Process finished with exit code 0
```

한글 클래스 가능
- 되도록이면 자제하자

▣ 추상화



- 상세한 정보는 무시하고 필요성에 의해 있어야할 정보들만 간추려서 구성하는 것
- 모든 객체에 공통적인 핵심적인 개념.속성이나 기능을 묶어 이름을 붙이는 것
- 추상화 목적은 프로그램을 사람이 읽고 작성하기 쉽게 하기 위함이다.