[디지털 컨버전스] 스마트 콘텐츠와 웹 융합 응용 SW 개발자 양성과정

17회차 수업
2021/06/01 월요일

강사 : 이상훈
학생 : 김원석

1. 상속 (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");
    }
}
```

```java
public class ExtendsTest {
    public static void main(String[] args) {
        A a = new A();
        a.b();   //A
        System.out.println("A a: " + a.a);  // 10

        AA aa = new AA();
        aa.b();   //AA
        aa.c();   //C
        System.out.println("AA aa: " + aa.a); //20

        // new의 대상은 AA()이며
        // 접근 데이터는 데이터타입 A를 참조해야한다.
        A a1 = new AA();
        a1.b();   //AA
        System.out.println("A a1: " + a1.a);    //10

    }
}
```

//중요 포인트

```java
class Car {
    private float rpm;          //Car라는 클래스를 만들고 그곳에
    private float fuel;         //private으로 각각 필요한 정보를 넣는다.
    private float pressure;
    private String color;

    public void setRpm (float rpm) { this.rpm = rpm; }     //Getter 와 Setter로
    public float getRpm() { return rpm; }                  //각 정보들을 받고 입력할수 있도록 준비한다.
    public float getFuel() { return fuel; }
    public void setFuel(float fuel) { this.fuel = fuel; }
    public float getPressure() { return pressure; }
    public void setPressure(float pressure) { this.pressure = pressure; }
    public String getColor() { return color; }
    public void setColor(String color) { this.color = color; }
}

class SportsCar extends Car {
    private Boolean booster;

    public Boolean getBooster() { return booster; }
    public void setBooster(Boolean booster) { this.booster = booster; }

    @Override
    public String toString() {
        //super의 경우엔 상속해준 상속자를 직접 호출한다.
        return "SportsCar{" +
                "rpm=" + super.getRpm()+            //super로 car의 정보들을 상속 받은 모습
                ", fuel=" + super.getFuel()+
                ", pressure=" + super.getPressure()+
                ", color=" + super.getColor()+
                ", booster=" + booster +
                '}';
```

//Car라는 클래스를 만들고
그곳에 private으로 각각 수행할
정보들을 넣는다.

//Getter와 Setter로
각 정보들을 받고 또 입력할수
있도록 준비한다.

//super로 Car클래스의 정보들을
toString으로 상속 받는다.

```java
// 기존에 잘 만들어진 정보에 새로운 내용을 추가하여 작업하고자 한다.
// 내용을 변경하는것보다는 새로운 클래스에 상속을 활용하여 작업하는 것을 권장한다.
// (일전에 잠깐 언급했던 SRP 규칙 때문에 그렇다) // 잘 짜여진 클래스를 건드려 버리면 같이 작업하던 많은 사람들이 피해 보게된다.
public class CarTest {
    public static void main(String[] args) {
        SportsCar sc = new SportsCar();

        sc.setRpm(100);
        sc.setFuel(2.5f);
        sc.setPressure(1.0f);
        sc.setColor("Dark Gray");
        sc.setBooster(false);

        System.out.println(sc);
```

```java
//new로 객체를 생성하고
//Car의 정보값을 새롭게 값을넣어
//호출한 모습
```

추상화 (Abstraction)와 인터페이스

-복잡한 문제로부터 본질을 이해하기 위해, 불필요한 세부사항은
배제하고
사용자들은 편하게 라이브러리를 사용하여 개발의 집중하도록

```
// 인터페이스 작성법
// 1. 일단 interface를 적는다.
// 2. 인터페이스명(일종의 클래스 같은 것이라고 보면 됨)을 적는다.
// 3. 인터페이스 내부에는 매서드 프로토타입을 작성한다.
//    (프로토타입이 뭘까요 ? 매서드의 접근 제한자, 리턴 타입, 매서드 이름, 입력등을 기록한 형태)
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!");
        }
    };
```

//리모컨 인터페이스를 new 를 통해 호출하여 함수 오버로딩을 사용해
각 필요한 기능들을 입력한다.

```
    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();
    }
}
```

//testMethod 함수와
testMethod2 함수에
호출할 값의 명을 적는다.

//호출