

[디지털 컨버전스] 스마트 콘텐츠와 웹 융합 응용SW 개발자 양성과정

강사 : 이상훈

학생 : 임초롱

Inheritance (상속)

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day17/src/ExtendsTest.java>

상속 (Inheritance) :

상속은 어떠한 클래스가 있을 때 그 클래스가 갖고 있는 변수와 매서드를 확장하여 (상속하여) 다른 클래스가 활용할 수 있도록 하는 것이다.

부모가 갖고 있는 기능을 덮어쓰기 하는 것.

재정의 = Overriding

상속의 목적 :

1. 재사용성을 높일 수 있다.
2. 유지보수의 편의성을 높일 수 있다.
3. 가독성을 높일 수 있다.
4. 코드의 양을 줄일 수 있다.

Overriding vs Overloading

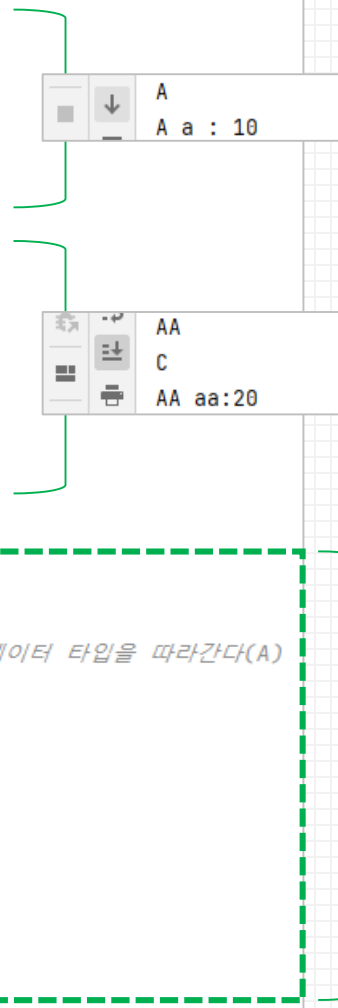
- **Overriding**은 상속의 개념으로 부모클래스에 있는 매서드나 변수를 extends를 통해 굳이 같은 형태의 매서드를 입력하지 않아도 부모클래스 있는 매서드를 쓸 수 있는 기능이다.
- **Overloading**은 부모클래스가 갖고있는 매서드의 형태를 자식클래스에서 조금 바꿔 이름을 같게 사용할 수 있다. (매서드의 매개변수 타입, 개수변화 등)
원래 매서드는 이름이 같을 수 없지만 Overloading은 가능하다.

Inheritance (상속)

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day17/src/ExtendsTest.java>

```
1 class A{
2     int a = 10;
3
4     void b() {
5         System.out.println("A");
6         // A를 출력한다.
7     }
8 }
9 // extends 키워드가 바로 상속!
10 // 상속: 말 그대로 재산을 물려 받는것이다.
11 // 클래스의 내용물들을 활용할 수 있게 된다.
12 class AA extends A{
13     // 클래스 A를 상속받은 AA
14     int a = 20;
15     void b() {
16         System.out.println("AA");
17         // AA를 출력한다
18     }
19
20     void c(){
21         System.out.println("C");
22         // C를 출력한다.
23     }
24 }
25
```

```
26 public class ExtendsTest {
27     public static void main(String[] args) {
28         A a = new A();
29         // A클래스 타입의 a 객체
30         a.b();
31         // A를 출력한다.
32         System.out.println("A a : " + a.a);
33         // 클래스 A의 int a = 10
34
35
36         AA aa = new AA();
37         aa.b();
38         // AA를 출력한다
39         aa.c();
40         // C를 출력한다.
41         System.out.println("AA aa:" + aa.a);
42         // 클래스 AA의 int a = 20
43
44
45         A a1 = new AA();
46         // new한 대상이 중요함, AA의 객체를 만든것
47         // 매서드는 new된 대상을 따라가고, 데이터는 데이터 타입을 따라간다(A)
48
49         // 접근 데이터는 데이터타입 A를 참조해야한다.
50         // 접근 데이터 : 객체 내부에 있는 데이터
51         // 접근데이터는 데이터 타입 A를 참조해야한다.
52
53         a1.b();
54         // AA를 출력한다
55         System.out.println("A a1: " + a1.a);
56         // 클래스 A의 int a = 10
57     }
58 }
```



Inheritance (상속)

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day17/src/CarTest.java>

```
1 class Car{
2     // ex_ 모든 자동차가 공통으로 가지는 객체
3     private float rpm; //공회전
4     private float fuel; //연료
5     private float pressure; //타이어 공기압
6     private String color; //색상
7
8     public float getRpm() { return rpm; }
9     public void setRpm(float rpm) { this.rpm = rpm; }
10    public float getFuel() { return fuel; }
11    public void setFuel(float fuel) { this.fuel = fuel; }
12    public float getPressure() { return pressure; }
13    public void setPressure(float pressure) { this.pressure = pressure; }
14    public String getColor() { return color; }
15    public void setColor(String color) { this.color = color; }
16
17    // 기존에 잘 만들어진 정보에 새로운 내용을 추가하여 작업하고자 한다.
18    // 내용을 변경하는것보다는 새로운 클래스에 상속을 활용하여 작업하는 것을 권장한다.
19    // (일전에 잠깐 언급했던 SRP 규칙 때문에 그렇다)
20
21    class SportsCar extends Car {
22        // 클래스 Car를 상속받은 SportsCar
23        // 스포츠카는 모든 자동차가 공통으로 가지는 객체를 가지면서도
24        // 스포츠카만의 특별한 객체를 가진다.
25        // 따라서 Car와 동일한 객체에 대해서는 상속받는다.
26
27        private Boolean booster;
28        // 스포츠카의 특별한 객체
29    }
30
31    // ... (rest of the code is not visible in the image)
```

```
46    public Boolean getBooster() { return booster; }
47    public void setBooster(Boolean booster) { this.booster = booster; }
48
49    @Override
50    public String toString() {
51        // super의 경우엔 상속해준 상속자를 직접 호출한다.
52        return "SportsCar{" +
53            "rpm=" + getRpm() +
54            ", fuel=" + getFuel() +
55            ", pressure=" + getPressure() +
56            ", color=" + getColor() +
57            ", booster=" + booster +
58            '}';
59    }
60
61    public class CarTest {
62        public static void main(String[] args) {
63            SportsCar sc = new SportsCar();
64
65            sc.setRpm(100);
66            sc.setFuel(2.5f);
67            sc.setPressure(1.0f);
68            sc.setColor("Dark Grey");
69            sc.setBooster(false);
70
71            System.out.println(sc);
72
73            "C:\Program Files\Java\jdk-15.0.2\bin\java.exe" -javaagent:C:\Users\user\AppData\
74            SportsCar{rpm=100.0, fuel=2.5, pressure=1.0, color=Dark Grey, booster=false}
75        }
76    }
```

Inheritance (상속)

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day17/src/InheritanceWithSuperTest.java>

```
1 class Vehicle{
2     // ex. 모든 탈 것이 공통으로 가지는 객체
3     private float rpm; //공회전
4     private float fuel; //연료
5     private float pressure; //타이어 공기압
6     private String color; //색상
7
8     public Vehicle(float rpm, float fuel, float pressure, String color) {
9         // Vehicle 클래스의 생성자
10        this.rpm = rpm;
11        this.fuel = fuel;
12        this.pressure = pressure;
13        this.color = color;
14    }
15
16    @Override
17    public String toString() {
18        return "Vehicle{" +
19            "rpm=" + rpm +
20            ", fuel=" + fuel +
21            ", pressure=" + pressure +
22            ", color='" + color + '\'' +
23            '}';
24    }
25 }
26 class Airplane extends Vehicle{
27     // 클래스 Vehicle을 상속받은 Airplane
28     // 비행기는 모든 탈 것이 공통으로 가지는 객체를 가지면서도
29     // 비행기만의 특별한 객체를 가진다.
30     // 따라서 Airplane과 동일한 객체에 대해서는 상속받는다.
31
32     // 비행기에 관련된 객체체
33     private float aileron;
34     private float pitch;
35     private float rudder;
```

```
37     public Airplane(float rpm, float fuel, float pressure, String color,
38         float aileron, float pitch, float rudder){
39
40         // super()는 무엇이 되었든 상속자인 부모를 호출한다.
41         // super()만 적혀 있으니 생성자를 호출하게 된다.
42         super(rpm, fuel, pressure, color);
43         // super가 부모 클래스를 불러옴
44
45         this.aileron = aileron;
46         this.pitch = pitch;
47         this.rudder = rudder;
48         // this : 자기 자신/ class의 것을 호출한다.
49         // super : 자기 부모 class를 호출한다.
50     }
51
52     @Override
53     public String toString() {
54         return "Airplane{" +
55             // super.toString()은 부모 클래스의 toString()을 호출한 것이다.
56             "super.Vehicle()" + super.toString() +
57             ", aileron=" + aileron +
58             ", pitch=" + pitch +
59             ", rudder=" + rudder +
60             '}';
61     }
62 }
63 public class InheritanceWithSuperTest {
64     public static void main(String[] args) {
65         Vehicle v = new Vehicle( rpm: 200, fuel: 1.2f, pressure: 1.0f, color: "Red");
66         System.out.println(v);
67
68         Airplane a = new Airplane( rpm: 1000, fuel: 112.5f, pressure: 12.3f, color: "White",
69             aileron: 77.3f, pitch: 0.02f, rudder: 33.9f);
70
71         System.out.println(a);
72     }
73 }
```

Interface (인터페이스)

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day17/src/InterfaceTest.java>

인터페이스 (Interface) :

- 추상적인 개념.
- 인터페이스는 역할은 어떤 클래스가 있고, 그 클래스가 특정한 인터페이스를 사용한다면 그 클래스는 반드시 그 인터페이스에 포함되어 있는 매서드를 구현하도록 강제하는 것이다.
- 만약 인터페이스에서 강제하고 있는 매서드를 구현하지 않는다면 애플리케이션은 컴파일 되지 않는다.

인터페이스 작성법 :

1. 일단 interface를 적는다.
2. 인터페이스명(일종의 클래스 같은 것)을 적는다.
3. 인터페이스 내부에는 매서드 프로토타입을 작성한다.
(프로토 타입이란, 매서드의 접근제한자, 리턴타입, 매서드 이름, 입력등을 기록한 형태)

추상화 :

OOP(객체지향) 에서 제일 중요시 여기는 것이 추상화이다.
추상화가 궁극적으로 추구하는 것은 모든 내용을 알지 못하더라도 쉽게 실행할 수 있도록 하는 것이다.

인터페이스 특징 :

1. 하나의 클래스가 여러 개의 인터페이스를 구현할 수 있다.
2. 인터페이스도 상속이 된다.
3. 인터페이스의 멤버는 반드시 public이다.
(인터페이스 내의 접근제한자는 누구나 접근할 수 있는 Public 이어야한다.)
4. 인터페이스는 사용할 기능만 선언할 뿐 자세한 구현이 없다.

Interface (인터페이스)

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day17/src/InterfaceTest.java>

```
7 interface Remocon{
8     public void turnOn();
9     public void turnOff();
10
11 }
12
13 // 리모콘 제조사가 15만개
14 /*
15     public void companyATurnOn();
16     public void companyBTurnOn();
17     public void companyZTurnOn();
18     public void companyAATurnOn();
19     public void companyABurnOn();
20     public void companyAZurnOn();
21     public void companyZZTurnOn();
22     public void companyZZZTurnOn();
23 */
24 // .....
25 // TurnOn() 매서드만 몇 개 ? 15만개
26 // 각 리모콘마다 개별적 프로그래밍이 불가능하다.
27
28 // 추상화란 무엇인가 ??????
29 // 객체 <<<=== 대표적인 추상화의 예
30 // 객체 <<<=== 현 시점에서 우리는 무엇을 생각하는가 ?
31 // new, 메모리에 올라간 데이터들 혹은 정보들 ...
32 // 단어가 어떤 함축된 의미를 포함해버렸음(우리는 알게 모르게 사용하고 있었고)
33 // 객체란 단어만 보고도 이것이 어떻게 어떻게 형성되었는지 등이 이미 뇌리에 스치고 있음
34
35 // KKK사의 컴퓨터를 켜다.
36 // GH사의 라디오를 켜다.
37 // A사의 리모콘을 켜다. =====> 켜다(원진 모르겠지만)
38 // B사의 리모콘을 켜다.
39 // .....
40 // Z사의 리모콘을 켜다.
```

```
43 // OOP(객체지향)에서 제일 중요시 여기는 것이 바로 추상화다.
44 // 현재까지의 내용을 토대로 추상화란 궁극적으로 무엇을 추구하는것인가 ?
45
46 // 복잡하고 어렵고 토나오는것은 우리가 해줄게
47 // (자바 라이브러리 개발자 진영 및 스프링 프레임워크 개발 진영)
48 // 라이브러리 사용자들은 편하게 API 사용해서 개발만 하세요 ~
49 // 이런 입력 ----> Black Box(블랙 박스) ----> 요런 출력이 나와요
50
51 // sout() ==> System.out.println()
52
53
54 class AbstractTest{
55     Remocon rc = new Remocon() {
56         @Override
57         public void turnOn() {
58             System.out.println("나는 RC 자동차용 리모콘이야! RF 송수신기가 지금 활성화되었어!");
59         }
60
61         @Override
62         public void turnOff() {
63             System.out.println("이제 헤어질 시간이야 ! RF 송수신기 신호 출력을 차단할게!");
64         }
65     };
66
67     Remocon radio = new Remocon() {
68         @Override
69         public void turnOn() {
70             System.out.println("나는 라디오야! 지금부터 주파수 채널 매칭을 시작할게!");
71         }
72
73         @Override
74         public void turnOff() {
75             System.out.println("이젠 안녕! 주파수 채널 매칭을 끝낼게!");
76         }
77     }
78 }
```

Interface (인터페이스)

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day17/src/InterfaceTest.java>

```
81 public void testMethod(){
82     Remocon tv = new Remocon() {
83         @Override
84         public void turnOn() {
85             System.out.println("나는 TV야! AM/FM 신호를 수신할게 ! 이제부터 방송을 보자!");
86         }
87
88         @Override
89         public void turnOff() {
90             System.out.println("AM/FM 신호를 차단할게 ! 내일 또 보자!");
91         }
92     };
93     tv.turnOn();
94     radio.turnOff();
95 }
96 public void testMethod2 (){
97     rc.turnOn();
98     radio.turnOff();
99 }
100 }
101
102 public class InterfaceTest {
103     public static void main(String[] args) {
104         AbstractTest at = new AbstractTest();
105
106         at.testMethod();
107         at.testMethod2();
108
109     }
110 }
111 }
```

```
Run: InterfaceTest
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" -jav
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
```