

# (디지털컨버전스) 스마트 콘텐츠와 웹 융합 응용SW개발자 양성과정

---

2021년 6월 2일  
[ 18일차 복습 ]

- 수강생 : 김 민 규
- 강의장 : 강 남 C
- 수강 기간 : 2021. 05. 07 ~ 2021. 12. 08
- 수강 시간 : 15:30 ~ 22:00
- 이상훈 강사님 | 이은정 취업담임



## ▶ 내용 : 인터페이스(interface)\_implements

```
+ class Lamp implements LampMethod {  
+     @Override  
+     public void lightOn() {  
+         System.out.println("Lamp를 켭니다.");  
+     }  
+  
+     @Override  
+     public void lightOff() {  
+         System.out.println("Lamp를 끕니다.");  
+     }  
+ }
```

```
public class InterfaceTest {  
    public static void main(String[] args) {  
        Lamp lamp = new Lamp();  
  
        lamp.lightOn();  
        lamp.lightOff();  
  
        StreetLamp streetLamp = new StreetLamp();  
  
        streetLamp.lightOn();  
        streetLamp.lightOff();  
  
        Led led = new Led();  
  
        led.lightOn();  
        led.lightOff();  
    }  
}
```

```
6 class Lamp {  
7     LampMethod lamp = new LampMethod() {  
8         @Override  
9         public void lightOn() {  
10             System.out.println("Lamp를 켭니다.");  
11         }  
12  
13         @Override  
14         public void lightOff() {  
15             System.out.println("Lamp를 끕니다.");  
16         }  
17     };  
18 }
```

```
48 public class g_Quiz54_Interface {  
49     public static void main(String[] args) {  
50         Lamp lamp = new Lamp();  
51  
52         lamp.lamp.lightOn();  
53         lamp.lamp.lightOff();  
54  
55         StreetLamp streetLamp = new StreetLamp();  
56  
57         streetLamp.streetLamp.lightOn();  
58         streetLamp.streetLamp.lightOff();  
59  
60         Led led = new Led();  
61  
62         led.led.lightOn();  
63         led.led.lightOff();  
64     }  
65 }
```

## 주요 내용

--  
Implements를 사용하게 된다면,  
중간 매개체를 지워도된다.

Class 클래스명 impements 인터페이스명

--

## ▶ 내용 :

```
1 + class LampMethod2 {
2 +     public void lightOn() {
3 +         System.out.println("독립은 무슨 종속이다!");
4 +     }
5 +     public void lightOff() {
6 +         System.out.println("무조건 우리말에 따르라 ~!!~!");
7 +     }
8 + }
9 +
10 + // 저번에는 클래스 내부에 인터페이스에 대한 객체를 생성한 반면
11 + // 이번에는 implements를 사용하여 해당 클래스에서
12 + // 인터페이스 내부의 미구현 메서드를 구현해줌으로서 동작을 하게 된다.
13 + class Lamp2 extends LampMethod2 {
14 +
15 + }
16 +
17 + class StreetLamp2 extends LampMethod2 {
18 +
19 + }
20 +
21 + class Led2 extends LampMethod2 {
22 +
```

```
25 + public class InterfaceVersusExtendsTest {
26 +     public static void main(String[] args) {
27 +         Lamp2 lamp = new Lamp2();
28 +
29 +         lamp.lightOn();
30 +         lamp.lightOff();
31 +
32 +         StreetLamp2 streetLamp = new StreetLamp2();
33 +
34 +         streetLamp.lightOn();
35 +         streetLamp.lightOff();
36 +
37 +         Led2 led = new Led2();
38 +
39 +         led.lightOn();
40 +         led.lightOff();
41 +     }
42 + }
```

## 주요 내용

--

LampMethod2에 종속되어

Lamp2 와 StrretLamp2와 Led2는

LampMethod2의 메서드를 동작(출력)하게된다.

--

## ▶ 내용 : HashSet\_addAll 과 retainAll

```
1  + import java.util.HashSet;
2  + import java.util.Set;
3  +
4  + public class SetFeatureTestWithHashSet {
5  +     public static void main(String[] args) {
6  +         Set<String> s1 = new HashSet<String>();
7  +         Set<String> s2 = new HashSet<String>();
8  +
9  +         s1.add("Apple");
10 +         s1.add("Tesla");
11 +         s1.add("Microsoft");
12 +
13 +         s2.add("Tesla");
14 +         s2.add("Alphabet");
15 +         s2.add("Texas Instruments");
16 +
17 +         Set<String> union = new HashSet<String>(s1);
18 +         union.addAll(s2);
19 +
20 +         Set<String> intersection = new HashSet<String>(s1);
21 +         intersection.retainAll(s2);
22 +
23 +         System.out.println("합집합: " + union);
24 +         System.out.println("교집합: " + intersection);
25 +     }
26 + }
```

## 주요 내용

--

HashSet

Java내의 collection 중 가장 빠른 속도를 자랑함

또한 Set(집합)의 특성에 따라 중복을 허용하지 않는다.

--

순서가 중요할 경우 ArrayList 사용  
빠른 처리만 원한다면 HashSet 권장

--

addAll – 합집합

retainAll – 교집합

## ▶ 내용 : HashSet\_add

```
1  + import java.util.HashSet;
2  + import java.util.Set;
3  +
4  + public class HowToUseHashSet {
5  +     public static void main(String[] args) {
6  +         Set<String> s = new HashSet<String>();
7  +
8  +         String[] sample = {"안녕", "하이", "헬로", "안녕", "안녕"};
9  +
10 +         // 집합의 특성: 중복 허용 x
11 +         for (String str : sample) {
12 +             if (!s.add(str)) {
13 +                 System.out.println("중복되었습니다: " + str);
14 +             }
15 +         }
16 +
17 +         // size()는 원소의 개수
18 +         System.out.println(s.size() + " 중복을 제외한 단어: " + s);
19 +     }
20 + }
```

## 주요 내용

--

HashSet은 집합의 특성으로 중복을 허용하지 않음

if (!s.add(str)) 에서  
s.add(str)는 str에  
지정된 값들이 있을 경우 True이지만

!s.add(str)이므로  
지정되지 않은 값들에 대하여 출력이  
이루어짐.

즉, 중복된 값들에 대해서만  
출력이 이루어짐.

--

Size는 원수의 갯수를 의미함.  
Length와 유사.

## ▶ 내용 :

```
1 import java.util.HashMap;
2 import java.util.Map;
3
4 class Student {
5     int age;
6     String name;
7
8     public Student (int age, String name) {
9         this.age = age;
10        this.name = name;
11    }
12
13    @Override
14    public String toString() {
15        return "Student{" +
16            "age=" + age +
17            ", name='" + name + '\'' +
18            '}';
19    }
20 }
21
22 public class HashMapTest {
23     public static void main(String[] args) {
24         // Map의 특성중 하나가 key와 value가 분리됨
25         // Map<Key, Value>
26         // 특별히 특정 데이터타입을 지켜줘야 하는 것은 없다.
27         // 나는 "열쇠"를 키로 사용하고 "오아아악!"을 값으로 쓸거야! 하면 쓰면 된다.
28         Map<Integer, Student> st = new HashMap<Integer, Student>();
29
30         // 앞에 오는 숫자는 인덱스가 아니다.
31         // 단지 사물함을 여는데 필요한 열쇠일 뿐
32         st.put(7, new Student(42, "Bob"));
33         st.put(2, new Student(33, "Chris"));
34         st.put(44, new Student(27, "Denis"));
35         st.put(3, new Student(29, "David"));
36
37         System.out.println(st);
38
39         st.remove(2);
40
41         System.out.println(st);
42
43         st.put(3, new Student(77, "Jessica"));
44
45         System.out.println(st);
46
47         for (Map.Entry<Integer, Student> s : st.entrySet()) {
48             Integer key = s.getKey();
49             Student value = s.getValue();
50             System.out.println("key = " + key + ", value = " + value);
51         }
52
53         // 나는 "열쇠"를 키로 사용하고 "오아아악!"을 값으로 쓸거야! 하면 쓰면 된다.
54         Map<String, String> strMap = new HashMap<String, String>();
55
56         strMap.put("열쇠", "오아아악!");
57
58         // HashMap을 사용할때는 이 방식이 변하지 않습니다.
59         // 추상화의 연장선 관점에서 아래 사항을 준수하여 코딩하면 어떤 상황에서든 key, value 값을 얻을 수 있습니다.
60         // Entry<키 데이터타입, 밸류 데이터타입> 형식은 지켜주세요.
61         for (Map.Entry<String, String> map : strMap.entrySet()) {
62             String key = map.getKey();
63             String value = map.getValue();
64             System.out.println("key = " + key + ", value = " + value);
65         }
```

## 주요 내용

--

# Map

Map의 특성 중 하나가  
Key와 value가 분리됨  
Map<Key, Value>

--

# put

객체.put( a, b) -> a는 열쇠이며, b는값

Put을 통해 값을 추가하거나  
수정할 수 있음

--

# remove

St.remove(7)

St 객체의 키값 7의 내용을 지움

--

# Entry

Entry<키데이터타입, 밸류데이터타입>  
형식을 지켜줘야함.