

[디지털 컨버전스] 스마트 콘텐츠와 웹 융합 응용SW 개발자 양성과정

강사 : 이상훈

학생 : 임초롱

Interface

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day18/src/InterfaceTest.java>

〈 인터페이스에서 implements를 사용했을 경우 〉

```
1 interface LampMethod {
2     public void lightOn();
3     public void lightOff();
4 }
5
6 // 저번에는 클래스 내부에 인터페이스에 대한 객체를 생성한 반면
7 // 이번에는 implements를 사용하여 해당 클래스에서
8 // 인터페이스 내부의 미구현 메서드를 구현해줌으로서 동작을 하게 된다.
9
10 class Lamp implements LampMethod {
11     @Override
12     public void lightOn() {
13         System.out.println("Lamp를 켭니다.");
14     }
15
16     @Override
17     public void lightOff() {
18         System.out.println("Lamp를 끕니다.");
19     }
20 }
21
```

```
46 public class InterfaceTest {
47     public static void main(String[] args) {
48         Lamp lamp = new Lamp();
49
50         lamp.lightOn();
51         lamp.lightOff();
52     }
53 }
```

〈 인터페이스에서 implements를 사용하지 않았을 경우 〉

```
10 interface Light {
11     // 인터페이스는 자체는 무조건 public
12     // private일 이유가 없다. 인터페이스 사용 불가능
13     public void lightOn();
14     public void lightOff();
15 }
16
17 class Lamp {
18     Light lamp = new Light() {
19         @Override
20         public void lightOn() { System.out.println("램프에 불이 켜졌습니다! 반짝반짝!"); }
21
22         @Override
23         public void lightOff() { System.out.println("램프에 불이 꺼졌습니다! 어둑어둑!"); }
24     };
25 }
26
```

```
57 public class Quiz54 {
58     public static void main(String[] args) {
59         Lamp lgt = new Lamp();
60         lgt.lamp.lightOn();
61         lgt.lamp.lightOff();
62     }
63 }
```



HashSet

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day18/src/HashSetTest.java>

```
4 public class HashSetTest {
5     public static void main(String[] args) {
6         HashSet<String> set = new HashSet<String>();
7         // HashSet 속도가 가장 빠르다.
8         // 또한 HashSet의 집합(Set)의 특성을 가지고 있어서 중복을 허용하지 않는다.
9         // 집합의 수학적 특성 : 중복을 허용하지 않는다.
10        // 어떤 데이터 사용시 자체적으로 중복을 허용하지 않겠다고 했을때 HashSet 사용
11        // 일종의 배열, 순서는 보장하지 않지만 속도는 빠른 ArrayList
12
13        set.add("우유");
14        set.add("빵");
15        set.add("베이컨");
16        set.add("소시지");
17        set.add("HAM");
18        set.add("파스타");
19        set.add("계란");
20        set.add("아메리카노");
21        set.add("HAM");
22        set.add("ham");
23        // 순서대로 출력되지 않는다.
24        // 그렇다고 랜덤으로 매 출력 시 달라지는 것은 아니다.
25
26        System.out.println(set);
27    }
```

[빵, HAM, ham, 우유, 파스타, 베이컨, 계란, 아메리카노, 소시지]

```
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
```

```
HashSet<Integer> set2 = new HashSet<Integer>();

set2.add(10);
set2.add(20);
set2.add(40);
set2.add(80);
set2.add(5);
set2.add(15);

System.out.println(set2);
```

[80, 20, 5, 40, 10, 15]

```
// -----
Integer[] a = { 1, 3, 4, 5, 3, 10 };
HashSet<Integer> aa = new HashSet<>(Arrays.asList(a));
System.out.println(aa);

// 의 경우는 ?
// 이 경우 순서도 지키고, 중복도 피할 수 있지않나 ? 하는 생각이 듭니다
// 잘 안쓰는 방법일까요?

}
```

[1, 3, 4, 5, 10]

HashSet

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day18/src/HowToUseHashSet.java>

```
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         // 집합의 특성: 중복 허용 x
10
11         for (String str : sample) {
12             if (!s.add(str)) {
13                 // str 값("안녕", "하이", "헬로", "안녕", "안녕")을
14                 // s에 하나씩 추가했을때,
15                 // !(not)인 경우
16                 // 즉, 중복일때 중복되었습니다 : (중복값) 이 출력되고
17                 System.out.println("중복되었습니다 :" + str);
18             }
19         }
20         // 중복이 아닐때는 s에 추가된다.
21         System.out.println(s.size() + "중복을 제외한 단어 : " + s);
22     }
23 }
```

↑ "C:\Program Files\Java\jdk-15.0.2\bin\j
↓ 중복되었습니다 :안녕
중복되었습니다 :안녕
3중복을 제외한 단어 : [안녕, 하이, 헬로]

HashSet

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day18/src/SetFeatureTestWithHashSet.java>

```
4 // Set은 인터페이스
5 public class SetFeatureTestWithHashSet {
6     public static void main(String[] args) {
7         Set<String> s1 = new HashSet<String>();
8         Set<String> s2 = new HashSet<String>();
9
10        s1.add("Apple");
11        s1.add("Tesla");
12        s1.add("Microsoft");
13
14        s2.add("Tesla");
15        s2.add("Alphabet");
16        s2.add("Texas Instruments");
17
18        // 합집합 : addAll()
19        // 알 그대로 모두 다 더한다. (but, 중복은 피한다.)
20        Set<String> union = new HashSet<String>(s1);
21        union.addAll(s2);
22        //s1과 s2의 합집합 : Apple, Tesla, Microsoft, Alphabet, Texas Instruments
23        // 위 목록들이 순서대로 출력되지 않음 (그렇다고 매출력시 랜덤은 아니다.)
24
25        // 교집합 : retainAll()
26        Set<String> intersection = new HashSet<String>(s1);
27        intersection.retainAll(s2);
28        //s1과 s2의 교집합 : Tesla
29
30        System.out.println("합집합 = " + union);
31        System.out.println("교집합 = " + intersection);
32    }
33 }
34 }
```

```
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" -javaagent:C:\
합집합 = [Alphabet, Texas Instruments, Apple, Tesla, Microsoft]
교집합 = [Tesla]
```

Set 사용 시,

addAll() : 합집합의 개념으로, 두 배열의 값을 모두 다 더한다.
(집합의 특성으로 중복은 피한다.)

retainAll() : 교집합의 개념으로, 두 배열의 값의 공통 값 만을
남긴다.

HashMap

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day18/src/HashMapTest.java>

```
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 public class HashMapTest {
22     public static void main(String[] args) {
23         // 사물함에 물건을 보관하려함
24         // 보관하기 이전 key가 있음
25         // 보관하기 위해서 키를 뽑아서 내용물을 넣고 키를 꽂아?
26         // 다른 사람은 나의 사물함을 풀 수 없다
27         // 그런 특성을 적용한게 key
28         // value은 소지품
29         // Map(key, value)
30
31         // Map의 특성중 하나가 key와 value가 분리됨
32         // Map<Key, Value>
33         // 특별히 특정 데이터타입을 지켜줘야 하는 것은 없다.
34         // 나는 "열쇠"를 키로 사용하고 "으아아악!"을 값으로 쓸거야! 하면 쓰면 된다.
```

```
37 Map<Integer, Student> st = new HashMap<Integer, Student>();
38
39 // 앞에 오는 숫자는 인덱스가 아니다.
40 // 단지 사물함을 여는데 필요한 열쇠일 뿐
41
42 st.put(7, new Student( age: 42, name: "Bob"));
43 st.put(2, new Student( age: 33, name: "Chris"));
44 st.put(44, new Student( age: 27, name: "Denis"));
45 st.put(3, new Student( age: 29, name: "David"));
46
47
48 System.out.println("===== put으로 입력한 값 =====");
49 System.out.println(st);
50 ===== put으로 입력한 값 =====
51 {2=Student{age=33, name='Chris'}, 3=Student{age=29, name='David'}, 7=Student{age=42, name='Bob'}, 44=Student{age=27, name='Denis'}}
52 st.remove( key: 2);
53
54 System.out.println("===== key 2를 remove =====");
55 System.out.println(st);
56 ===== key 2를 remove =====
57 {3=Student{age=29, name='David'}, 7=Student{age=42, name='Bob'}, 44=Student{age=27, name='Denis'}}
```

HashMap

링크 <https://github.com/limcholong/LectureContents/blob/main/java/CholongLim/Day18/src/HashMapTest.java>

```
57
58 st.put(3, new Student( age: 77, name: "Jesica"));
59 System.out.println("===== key 3에 새로운 값 (중복불가) =====");
60 System.out.println(st);
```



===== key 3에 새로운 값 (중복불가) =====

{3=Student{age=77, name='Jesica'}, 7=Student{age=42, name='Bob'}, 44=Student{age=27, name='Denis'}}

Key3는 값이 중복이므로 put 값으로 대체된다.

```
61
62 // entrySet()은 map안에 있는 value 값을 하나씩 나열해준다.
63 // key와 value 한 세트가 Entry
64 System.out.println("===== entrySet =====");
65 for (Map.Entry<Integer, Student> s : st.entrySet()) {
66     Integer key = s.getKey();
67     Student value = s.getValue();
68     System.out.println("key = " + key + ", value = " + value);
69 }
70
71 // 나는 "열쇠"를 키로 사용하고 "으아아악!"을 값으로 쓸거야! 하면 쓰면 된다.
72 Map<String, String> strMap = new HashMap<String, String>();
73
74 strMap.put("열쇠", "으아아악!");
75 System.out.println("===== entrySet =====");
76
77 // HashMap을 사용할때는 이 방식이 변하지 않습니다.
78 // 추상화의 연장선 관점에서 아래 사항을 준수하여 코딩하면 어떤 상황에서든 key, value 값을 얻을 수 있습니다.
79 // Entry<키 데이터타입, 밸류 데이터타입> 형식은 지켜주세요.
80 for (Map.Entry<String, String> map : strMap.entrySet()) {
81     String key = map.getKey();
82     String value = map.getValue();
83     System.out.println("key = " + key + ", value = " + value);
84 }
85 }
86 }
```

```
===== entrySet =====
key = 3, value = Student{age=77, name='Jesica'}
key = 7, value = Student{age=42, name='Bob'}
key = 44, value = Student{age=27, name='Denis'}
```

entrySet() :

map안에 있는 value값을 하나씩 나열해준다.

st 안에 있는 value값을 하나씩 나열해준다.

Map.Entry< Integer , Student >에 각각 값이 나열된다.
그 값이 s에 대입되었고, get을 통해 불러와진다.

```
===== entrySet =====
key = 열쇠, value = 으아아악!
```

str 안에 있는 value값을 하나씩 나열해준다.

Map.Entry< String , String >에 각각 값이 나열된다.
그 값이 map에 대입되었고, get을 통해 불러와진다.

56번 문제

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

```
//Map(key, value)
//Map 과 HashMap의 차이는 무엇일까 ?

// 1. Map
// key와 value를 가진 집합이며 중복을 허용하지 않는다.
// 즉 한개의 key에 한개의 value가 매칭된다.

// 2. HashMap
// Map interface를 implements한 클래스로서 중복을 허용하지 않는다.
// Map의 특징인 key와 value의 쌍으로 이루어지며,
// key 또는 value 값으로서 null을 허용한다.
// 내부적으로 Entry<K,V>[] Entry의 array로 되어있다.

Map<String, Map<String, Integer>> map = new HashMap<String, Map<String, Integer>>();
Map<String, Integer>[] preparedMap = new HashMap[3];

String[] pattern = {"spear", "sword", "arrow"};
// 문양 배열에 창, 검, 활

for (int i = 0; i < 3; i++) {
    preparedMap[i] = new HashMap<String, Integer>();
    // preparedMap[0] = new HashMap<String, Integer>();
    // preparedMap[1] = new HashMap<String, Integer>();
    // preparedMap[2] = new HashMap<String, Integer>();
}
```

Set<>으로도 해보고 for each로도 해봤지만,
그림 중복 && 숫자 중복인 경우를 피하는 코드를 만들지 못해 복습만 했습니다..
오늘 수업 때 풀이 듣고 복습하여 다시 만들어보겠습니다.

```
for (int j = 0; j < 10; j++) {
    preparedMap[i].put(pattern[i] + j, j);
    // preparedMap[0].put(pattern[0] + 0, 0) ~ preparedMap[0].put(pattern[0] + 9, 9)
    // preparedMap[1].put(pattern[1] + 0, 0) ~ preparedMap[1].put(pattern[1] + 9, 9)
    // preparedMap[2].put(pattern[2] + 0, 0) ~ preparedMap[2].put(pattern[2] + 9, 9)
}

for (int i = 0; i < 3; i++) {
    map.put(pattern[i], preparedMap[i]);
    //(pattern[i], (패턴당 10개 카드 * 3))
    // 각 패턴 당, 패턴과 숫자의 이름이 기입된 3장의 카드들
}

System.out.println(map);
System.out.println("카드를 분배합니다.");

// 사용자에게 랜덤하게 4장의 카드를 나눠주자
for (int i = 0; i < 4; i++) {
    String sculpture = pattern[(int) (Math.random() * 3)]; // 3개의 패턴
    int randNum = (int) (Math.random() * 10); // 0 ~ 9 까지 10개의 숫자

    // 어떤 경우 중복불가인지 ?
    // 그림 숫자 모두 중복인 경우는 불가하다.
    // 그림 중복 && 숫자 중복

    System.out.println("사용자에게 분배된 카드는 = " + sculpture +
        " 문양의 " + randNum + " 카드입니다!");
}
}
```