(디지털컨버전스)스마트 콘텐츠와 웹 융합응용SW개발자 양성과정

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 –Joongyeon Kim(김중연)
jjjr69@naver.com

2021년 6월 2일 질문노트

[김중연]

```java
interface LampMethod {
    public void lightOn();
    public void lightOff();
}

//저번에는 클래스 내부에 인터페이스에 대한 객체를 생성한반면
//이번에는 implements를 사용하여 해당 클래스에서
//인터페이스 내부의 미구현 메소드를 구현해줌으로서 동작을 하게 된다
//extends하고 implements를 사용하는 것도 가능함
class Lamp implements LampMethod {  //인터페이스와 클래스 사이를 implements로 바로 연결시켜준다 (클래스내부에 인터페이스 객체 생성 필요없음)
    @Override
    public void lightOn() { System.out.println("Lamp를 킵니다."); }
    @Override
    public void lightOff() { System.out.println("Lamp를 끕니다."); }
}
class StreetLamp implements LampMethod {
    @Override
    public void lightOn() { System.out.println("가로등을 킵니다."); }
    @Override
    public void lightOff() { System.out.println("가로등을 끕니다."); }
}
class Led implements LampMethod {
    @Override
    public void lightOn() { System.out.println("LED등을 킵니다."); }
    @Override
    public void lightOff() { System.out.println("LED등을 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();
    }
}
```

Implements

인터페이스와 클래스 사이를 바로
연결시켜줌으로써 클래스내부에 인터페이스
객체를 생성하지 않아도 된다

```java
class LampMethod2 {
    public void lightOn() {
        System.out.println("독립은 무슨 종속이다!");
    }
    public void lightOff() {
        System.out.println("무조건 우리말에 따르라 ~~!!~!");
    }
}

// 저번에는 클래스 내부에 인터페이스에 대한 객체를 생성한 반면
// 이번에는 implements를 사용하여 해당 클래스에서
// 인터페이스 내부의 미구현 메서드를 구현해줌으로서 동작을 하게 된다.
class Lamp2 extends LampMethod2 {

}

class StreetLamp2 extends LampMethod2 {

}

class Led2 extends LampMethod2 {

}

public class InterfaceVersusExtendsTest {
    public static void main(String[] args) {
        Lamp2 lamp = new Lamp2();

        lamp.lightOn();
        lamp.lightOff();

          StreetLamp2 streetLamp = new StreetLamp2();

          streetLamp.lightOn();
          streetLamp.lightOff();

          Led2 led = new Led2();

          led.lightOn();
          led.lightOff();
      }
}
```

Implements를 사용하여 클래스를 인터페이스에 종속시켜 값을 출력할 수 있다.

```java
import java.util.HashSet;

public class HashSetTest {
    public static void main(String[] args) {
        HashSet<String> set = new HashSet<~>();

        set.add("우유");
        set.add("빵");
        set.add("베이컨");
        set.add("소시지");
        set.add("파스타");
        set.add("계란");
        set.add("아메리카노");
        set.add("HAM");
        set.add("ham");
        //HashSet의 핵심 특성중 하나 Java내에 존재하는 Collection중 가장 빠른 속도를 자랑함
        //또한 HashSet의 집합(Set)의 특성을 가지고 있어 중복을 허용하지 않는다.
        //대신 순서는 랜덤(정확히는 자바개발자들이 설정한 알고리즘대로)으로 출력된다.
        //중요: 순서가 중요하다면 ArrayList를 사용하세요.
        //       순서가 별로 중요하지않고 빠른처리를 원한다면 Hashset을 권장합니다.

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

Hashset의 특징
1. 계산처리가 가장 빠르다
2. 중복을 허용하지않는다(Set)
3. 순서가 고정된 랜덤으로 출력된다

```java
import java.util.HashSet;
import java.util.Set;

public class SetFeatureTestWithHashSet {
    public static void main(String[] args) {
        Set<String> s1 = new HashSet<~>();
        Set<String> s2 = new HashSet<~>();

        s1.add("Apple");
        s1.add("Tesla");
        s1.add("MicroSoft");

        s2.add("Tesla");
        s2.add("Alphabet");
        s2.add("Texas Instruments");

        Set<String> union = new HashSet<~>(s1);
        union.addAll(s2); //합집합 출력

        Set<String> intersection = new HashSet<~>(s1);
        intersection.retainAll(s2); //교집합부분만출력

        System.out.println("합집합:" + union);
        System.out.println("교집합"+ intersection);
    }
}
```

addAll
합집합출력

retainAll
교집합출력

```java
import java.util.HashMap;
import java.util.Map;

class Student {
    int age;
    String name;

    public Student (int age, String name) {
        this.age = age;
        this.name = name;
    }

    @Override
    public String toString() {
        return "Student{" +
                "age=" + age +
                ", name='" + name + '\'' +
                '}';
    }
}

public class HashMapTest {
    public static void main(String[] args) {
        // Map의 특성중 하나가 key와 value가 분리됨
        // Map<Key, Value>
        // 특별히 특정 데이터타입을 지켜줘야 하는 것은 없다.
        // 나는 "열쇠"를 키로 사용하고 "으아아앜!"을 값으로 쓸거야! 하면 쓰면 된다.
        Map<Integer, Student> st = new HashMap<~>();

        // 앞에 오는 숫자는 인덱스가 아니다.
        // 단지 사물함을 여는데 필요한 열쇠일 뿐
```

Map
Key와 value가 분리됨Map<Key, Value>
특정 데이터타입 지정필요없음
값은 put을 이용해 입력한다
Key의 수가 같으면 정보를 덮어씌우는 것도 가능하다

```java
//key와 value는 아래 예제에서 보듯이 put메소드를 이용하여 입력한다.
st.put(7, new Student( age: 42,  name: "Bob"));
st.put(2, new Student( age: 33,  name: "Chris"));
st.put(44, new Student( age: 27,  name: "Denis"));
st.put(3, new Student( age: 29,  name: "David"));

System.out.println(st);

st.remove( key: 2);

System.out.println(st);

st.put(3, new Student( age: 77,  name: "Jesica"));  //key 3이 덧씌어짐

System.out.println(st);

// entrySet()은 key와 value 모두가 필요할 경우 사용된다
for (Map.Entry<Integer, Student> s : st.entrySet()) {
    Integer key = s.getKey();
    Student value = s.getValue();
    System.out.println("key = " + key + ", value = " + value);
}

// 나는 "열쇠"를 키로 사용하고 "으아아악!"을 값으로 쓸거야! 하면 쓰면 된다.
Map<String, String> strMap = new HashMap<~>();

strMap.put("열쇠", "으아아악!");

    // HashMap을 사용할때는 이 방식이 변하지 않습니다.
    // 추상화의 연장선 관점에서 아래 사항을 준수하여 코딩하면 어떤 상황에서든 key, value 값을 얻을 수 있습니다.
    // Entry<키 데이터타입, 밸류 데이터타입> 형식은 지켜주세요.
    for (Map.Entry<String, String> map : strMap.entrySet()) {
        String key = map.getKey();
        String value = map.getValue();
        System.out.println("key = " + key + ", value = " + value);
    }
}
}
```

entrySet은 key와 value값 모두가 필요한 경우 사용한다

```java
import java.util.HashSet;
import java.util.Set;

public class HowToUseHashSet {
    public static void main(String[] args) {
        Set<String> s = new HashSet<String>();  // Set은 한국말로는 "집합"이고, 따로 저장 순서를 유지하지는 않는다.
                                                // 또한 중복 값을 허용하지 않는다는 특징을 갖고 있다.
        String[] sample = {"안녕", "하이", "헬로", "안녕", "안녕"};

        // 집합의 특성 : 중복 허용 x
        for (String str : sample) {
            if(!s.add(str)) {  //거짓을 부정해서 참으로 만든 후에  if문안에 들어감, 왜 거짓이지?(안녕이 첨으로 들어감(참)-> !로 인해 거짓이됨 -> if문을 빠져나간후 밑에 s에 출력됨
                               // 다시 한번 안녕이 나옴 -> 이미 안녕이 있으니 거짓 -> !로인해 결과가 참이됨 -> if문에 있는 str에 들어감)
                System.out.println("중복되었습니다"+ str);
            }
        }
        //size()는 원소의 개수
        System.out.println(s.size()+"중복을 제외한 단어 :" +s);
    }
}
```

```java
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

//56번 문제에서 게임을 진행하도록 한다.
//컴퓨터와 사용자가 이 카드 게임을 진행하도록 만들어보자!
//승리 판정 공식
//
//같은 문양의 숫자가 연속되게 3개 나오는 경우(예 9 8 7) > 서로 다른 문양의 같은 숫자
//서로 다른 문양의 같은 숫자                          > 서로 다른 문양의 숫자가 연속되게 3개 나옴
//서로 다른 문양의 숫자가 연속되게 3개 나옴            > 서로 같은 문양이 3개 나옴
//
//위와 같은 카드 게임을 만들어보자!
class CardGame {
    Map<String, Map<String, Integer>> map;
    Map<String, Integer>[] preparedMap;

    String[] pattern= {"spear", "sword", "arrow"};

    Set<Integer> s = new HashSet<Integer>();

    final int length=4;

    int randNum[] = new int[length];
    String sculptureArr[] = new String[4];

    public CardGame(){
        map = new HashMap<String, Map<String, Integer>>();
        preparedMap = new HashMap[3];
    }
```

57번문제를 풀려고 했지만
실패했습니다 ㅜ ㅜ
오늘 강사님 풀이하신 것을 토대로 다시
복습하겠습니다

```java
public void controlGame(){

    do {
        //1.먼저 카드의 정보를 입력하자
        cardSetting();
        //2. 그 다음 카드를 섞어 나눠준다
        cardDraw();
        //3. 중복된 숫자를 체크한다
        //numberOverlap();
        //4. 게임의 승리 조건을 만든다.
        //cardRule();
        //5. 컴퓨터와 사람중 누가 승리하였는지 판별하여 출력한다
        //gameWinner();
    }while(false);
}

public void cardSetting() {
    for (int i = 0; i < 3; i++) {
        preparedMap[i] = new HashMap<String, Integer>();

        for (int j = 0; j < 10; j++) {
            preparedMap[i].put(pattern[i] + j, j);
        }
    }

    for (int i = 0; i < 3; i++) {
        map.put(pattern[i], preparedMap[i]);
    }
    System.out.println(map);
}
public void cardDraw(){
```

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

        for (int i = 0; i < 4; i++) {
            String sculpture = pattern[(int)(Math.random() * 3)];
            sculptureArr[i] = sculpture;
            int randNum = (int)(Math.random() * 10);
            System.out.println("사용자에게 분배된 카드는 = " + sculpture +
                    " 문양의 " + randNum + " 카드입니다!");
        }
    }
}

public class Prob57 {
    public static void main(String[] args) {
        CardGame cardGame = new CardGame();

        cardGame.controlGame();

    }
}
```