디지털 컨버전스 기반 UXUI Pront전문 개발자 양성과정

- 강사 이상훈
- gcccompil3r@gmail.com
- 학생 김도혜
- kimdohye0728@gmail.com

▷implement 활용

- 인터페이스 객체를 만들어서 구현을 하는 방법도
- 있지만, implement를 사용하여 객체 만들지 않고 바로 메서드를 구현하였다.

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

// 저번에는 클래스 내부에 인터페이스에 대한 객체를 생성한 반면
// 이번에는 implements를 사용하여 해당 클래스에서
// 인터페이스 내부의 미구현 매서드를 구현해줌으로서 동작을 하게 된다.
class Lamp implements LampMethod {
    @Override
    public void lightOn() {
        System.out.println("Lamp를 킵니다.");
    }
```

```java
public class InterfaceTest {
    public static void main(String[] args) {
        Lamp lamp = new Lamp();

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

▷ HashSet

- HashSet은 ArrayList의 일종으로 특이사항은 중복을 허용하지 않는다.

- 순서대로 출력을 하지 않으므로, 만약 순서를 지켜야 할 필요가 있다면 ArrayList를 권장한다.

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

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

        String[] sample = {"안녕", "하이", "헬로", "안녕", "안녕"};

        // 집합의 특성: 중복 허용 x
        for (String str : sample) {
            if (!s.add(str)) {
                System.out.println("중복되었습니다: " + str);
            }
        }

        // size()는 원소의 개수
        System.out.println(s.size() + " 중복을 제외한 단어: " + s);
    }
```

▷HashSet

- addAll();은 두 집합 중 전체를 다 선택하는 기능이 있고
- retainAll();은 서로 교차되는 부분을 선택하는 기능이 있다.

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

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

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

▷Hashmap

```java
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
        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"));
```

```java
st.remove( key: 2);

System.out.println(st);

st.put(3, new Student( age: 77,  name: "Jesica"));
```

- HashMap은 키 값과 밸류 값을 나눌 수 있다는 장점이 있다.
- remove();와 put();을 사용해 키 값과 밸류 값을 넣고 뺄 수 있다.

▷Hashmap사용법

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

strMap.put("열쇠", "으아아앜!");

// 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);
}
```