

22.01.17 수업 review

1. List 인터페이스

중복을 허용하고 저장순서가 유지됨

1) ArrayList : Vector를 개선한 것으로 구현원리는 동일하다.

*for each : (for 변수선언 : 배열) {반복할 문장}

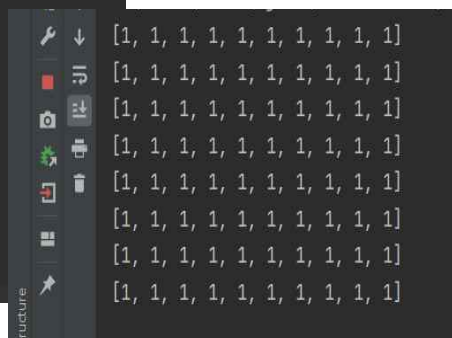
*.add(value): ArrayList에 데이터값을 넣어준다.

*shuffle : 배열이나 리스트를 랜덤으로 섞어준다.

```
1 import java.util.ArrayList;
2 public class ForEachTest {
3     public static void main(String[] args) {
4         ArrayList<Test> list = new ArrayList<Test>();
5         list.add (new Test());
6         list.add (new Test());
7         list.add (new Test());
8
9         //foreach
10        //list에 들어있는 값을 하나씩 빼와서 t에 배치한다.
11        for(Test t : list){
12            System.out.println(t);
13        }
14    }
15 }
```

```
1 public class Test {
2     int num1, num2;
3     static int cnt = 0;
4
5     public Test () {
6         num1 = 3 + cnt;
7         num2 = 7 + cnt++;
8     }
9
10    @Override
11    public String toString() {
12        return "Test{" +
13            "num1=" + num1 +
14            ", num2=" + num2 +
15            '}';
16    }
17 }
```

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.List;
4
5 public class ShuffleMap {
6     public static void main(String[] args) throws InterruptedException {
7         List<Integer> list = new ArrayList<>();
8
9         for (int i = 1; i <= 10; i++) {
10             list.add(1);
11         }
12         while (true) {
13             Collections.shuffle(list);
14             System.out.println(list);
15             Thread.sleep(1000);
16         }
17     }
18 }
```



2) Vector

*addElement : 벡터에 데이터값을 추가한다

*insertElementAt(value, index) : 인덱스에 데이터를 넣고 해당 자리에 있던 값은 뒤로 보냄

*setElementAt(value, index) : 특정 인덱스의 값을 변경한다.

*remove(index) : 특정 인덱스의 데이터를 지운다.

*elementAt(index) : 특정 인덱스의 데이터값을 불러온다.

```
1 import java.util.Vector;
2 public class JavaVectorTest {
3     public static void main(String[] args) {
4         Vector<String> v = new Vector<>();
5         System.out.println("Vector size : " + v.size());
6
7         v.addElement(new String( original: "아아"));
8         v.addElement(new String( original: "JPA"));
9         v.addElement(new String( original: "VUE"));
10        v.addElement(new String( original: "MySQL"));
11        v.addElement(new String( original: "TypeScript"));
12        System.out.println("Vector size 2 : " +v.size());
13
14        for(int i=0; i<v.size(); i++){
15            String tmp = v.elementAt(i);
16            System.out.println("Vector v의"+ i +"번째 "+tmp);
17        }
18    }
19 }
```

```
Vector size : 0
Vector size 2 :5
Vector v의0번째 아아
Vector v의1번째 JPA
Vector v의2번째 VUE
Vector v의3번째 MySQL
Vector v의4번째 TypeScript
```

```
4 public class JavaVectorTest2 {
5     public static void main(String[] args) {
6         Vector<Object> v = new Vector<>();
7
8         v.addElement(new Character( value: 'A'));
9         v.addElement(new String( original: "ABC"));
10        v.addElement(new Integer( value: 56));
11        v.addElement(new BigInteger( val: "3333"));
12        System.out.printf("vector 의 크기 :%d\n", v.size());
13
14        //insertElementAt()의 역할 : 인덱스 1번에 데이터를 넣고 해당 자리에 있는 데이터를 뒤로 보낸다.
15        v.insertElementAt(new Float( value: 3.141592), index: 1);
16        System.out.println("insertElementAt 이후 size "+v.size());
17
18        //setElementAt()의 역할 : 특정 인덱스 데이터값만 변한다.
19        v.setElementAt( obj: "이런것도 되니?", index: 3);
20        System.out.println("setElementAt 이후 size "+v.size());
21        System.out.printf("v의 0번째 : %c\n", (Character)v.elementAt( index: 0));
22        System.out.printf("v의 1번째 : %f\n", (Float)v.elementAt( index: 1));
23        System.out.printf("v의 2번째 : %s\n", (String)v.elementAt( index: 2));
24        System.out.printf("v의 3번째 : %s\n", (String)v.elementAt( index: 3));
25        System.out.printf("v의 4번째 : %d\n", (BigInteger)v.elementAt( index: 4));
26        //특정 인덱스의 데이터를 지운다.
27        v.remove( index: 0);
28    }
29 }
```

```
vector 의 크기 :4
insertElementAt 이후 size 5
setElementAt 이후 size 5
v의 0번째 : A
v의 1번째 : 3.141592
v의 2번째 : ABC
v의 3번째 : 이런것도 되니?
v의 4번째 : 3333
```

2. Set 인터페이스

중복을 허용하지 않고 저장순서가 유지되지 않는다.

1) HashSet : 데이터가 중복한다면 false를 반환함으로 중복처리에 효과적이다.

```
1 import java.util.HashSet;
2 import java.util.Set;
3
4 public class HashSetTest {
5     public static void main(String[] args) {
6         Set<String> s = new HashSet<>();
7         String [] duplicate = {"중복", "노중복", "중복", "에스중복"};
8
9         for (String dup : duplicate){
10             if (!s.add(dup)) {System.out.println("중복된 단어: "+dup);}
11             else {System.out.println("중복 아닌 단어: " +dup);}
12         }
13         System.out.println("중복되지 않은 단어 :"+s);
14     }
15 }
```

중복 아닌 단어: 중복
중복 아닌 단어: 노중복
중복된 단어: 중복
중복 아닌 단어: 에스중복
중복되지 않은 단어 :[에스중복, 중복, 노중복]

=> is.add(dup)가 false라면(=중복) 중복된 데이터값 출력
아니면 (true) 중복되지 않은 데이터 출력
HashSet s에는 중복되지 않는 값만 들어간다.

```
1 import java.util.HashMap;
2 import java.util.Map;
3
4 public class MapBasedDuplicateCheckTest {
5     public static void main(String[] args) {
6         Map<String, Integer> m = new HashMap<>();
7         String[] sample = {"to", "be", "or", "not", "to", "be", "is", "a", "problem", "to"};
8
9         for (String s : sample){
10             Integer freq = m.get(s);
11             m.put (s, (freq==null)? 1 : freq+1);
12         }
13
14         System.out.println("Map size : " +m.size());
15         System.out.println("Map - to ? "+m.containsKey("to") );
16         System.out.println(m.isEmpty());
17         System.out.println(m);
18     }
19 }
```

Map size : 7
Map - to ? true
false
{a=1, not=1, be=2, or=1, problem=1, is=1, to=3}

3. Map 인터페이스

key와 value를 쌍으로 묶어서 저장하는 컬렉션 클래스를 구현하는데 사용된다.

키는 중복이 허용되지 않지만 데이터값은 중복을 허용한다.

기존의 데이터와 중복된 키와 값을 저장하면 기존값은 사라지고 마지막 값이 저장됨

4. Map.Entry 인터페이스

Map인터페이스의 내부 인터페이스이다.

*getKey() : Entry의 key객체를 반환한다.

*getValue() : Entry의 value객체를 반환한다.

```
4 public class MapTest {
5     public static void main(String[] args) {
6         Map<Integer, Test> test = new HashMap<>();
7         //put을 통해서 Map(key, value) 형식으로 저장이 가능하다.
8         //key값 1에 new Test()로 만들어진 객체가 value로 저장된다.
9         test.put(1, new Test());
10        // key값 2에 new Test() 객체가 저장된다.
11        test.put(2, new Test());
12        test.put(3, new Test());
13        System.out.println(test);
14
15        test.remove((2));
16        System.out.println("remove()이후 : "+test);
17
18        //put()의 또다른 특성으로 같은 key에 새로운 값을 넣으면 덮어쓰기 됨
19        test.put(3, new Test());
20        //get을 통해서 특정 key 값을 가져온다.
21        System.out.println("test.put(3)이후 : "+ test.get(3));
22
23        for(Map.Entry<Integer, Test> t : test.entrySet()){
24            Integer key = t.getKey();
25            Test value = t.getValue();
26            System.out.println("key = "+ key + ", value ="+value);
27        }
```

```
{1=Test{num1=3, num2=7}, 2=Test{num1=4, num2=8}, 3=Test{num1=5, num2=9}}
remove()이후 : {1=Test{num1=3, num2=7}, 3=Test{num1=5, num2=9}}
test.put(3)이후 : Test{num1=6, num2=10}
key = 1, value =Test{num1=3, num2=7}
key = 3, value =Test{num1=6, num2=10}
Process finished with exit code 0
```