2021.06.07 Java

```java
import java.util.*;
public class _99th_Sorting {
    public static void main(String[] args) {
        String[] words = { "apple", "I", "WALK", "LINE", "EVENT", "Thread", "HARD", "GROUND"};
        ArrayList<String> words_arrayList = new ArrayList<>(Arrays.asList(words));
        // = List<String> list = Arrays.asList(words)라고 해도 됨.
        Collections.sort(words_arrayList); // 대문자 우선 알파벳 순으로 정렬
        System.out.println(words_arrayList);

        Integer[] numbers = {1, 2, 3, 33, 212, 213, 5135, 341, 545345, 8787};
        ArrayList<Integer> numbers_arrayList = new ArrayList<>(Arrays.asList(numbers));
        // = List<Integer> numbers_arrayList = Arrays.asList(numbers) 라고 해도 됨.
        Collections.sort(numbers_arrayList);
        System.out.println(numbers_arrayList);
```

〈Collection.sort〉

- 배열 정렬시키는 method.

Q. new를 사용해서 ArrayList형으로 만들면 words_arrayList를 객체화 시키는 것인데 new를 사용하지 않고 List형으로 만들어도 배열이기 때문에 객체라고 볼 수 있는 것인지?

new를 이용했을 때와 아닐 때의 차이점은 무엇인지..?

```java
Set fruits = new HashSet();
fruits.add("melon");
fruits.add("orange");
fruits.add("banana");
fruits.add("apple");
fruits.add("watermelon");
fruits.add("strawberry");

List fruitsList = new ArrayList(fruits);
fruitsList.add("List가 됐어도 ArrayList의 method 쓸 수 있음 왜냐하면..\n" +
        "List에서 미구현된 method들이 List를 implements하고 있는 ArrayList에서\n" +
        "구현되고 있기 때문. ");
Collections.sort(fruitsList);
System.out.println(fruitsList);
//sort는 List의 method이기 때문에 쓰려면 List로 변환시켜야됨.
//List는 Collection을 상속받는 interface.
```

```
"C:\Program Files\Java\jdk-16\bin\java.exe" -javaagent:C:\Users\Samuel\Ap
[EVENT, GROUND, HARD, I, LINE, Thread, WALK, apple]
[1, 2, 3, 33, 212, 213, 341, 5135, 8787, 545345]
[List가 됐어도 ArrayList의 method 쓸 수 있음 왜냐하면..
List에서 미구현된 method들이 List를 implements하고 있는 ArrayList에서
구현되고 있기 때문. , apple, banana, melon, orange, strawberry, watermelon]

Process finished with exit code 0
```

```
Type parameters: <E> – the type of elements in this list
public class ArrayList<E> extends AbstractList<E>
        implements List<E>, RandomAccess, Cloneable, java.io.Serializable
{
    @java.io.Serial
    private static final long serialVersionUID = 8683452581122892189L;
```

ArrayList는 List를 implements 한다.

```java
public interface List<E> extends Collection<E> {
    // Query Operations

    Returns the number of elements in this list. If this list co
    elements, returns Integer.MAX_VALUE.
    Returns: the number of elements in this list
    int size();

    Returns true if this list contains no elements.
    Returns: true if this list contains no elements
    boolean isEmpty();

    Returns true if this list contains the specified element.
    this list contains at least one element e such that Objec
    Params: o – element whose presence in this list is to be
    Returns: true if this list contains the specified element
    Throws: ClassCastException – if the type of the spe
            (optional)
            NullPointerException – if the specified ele
            elements (optional)
    boolean contains(Object o);
```

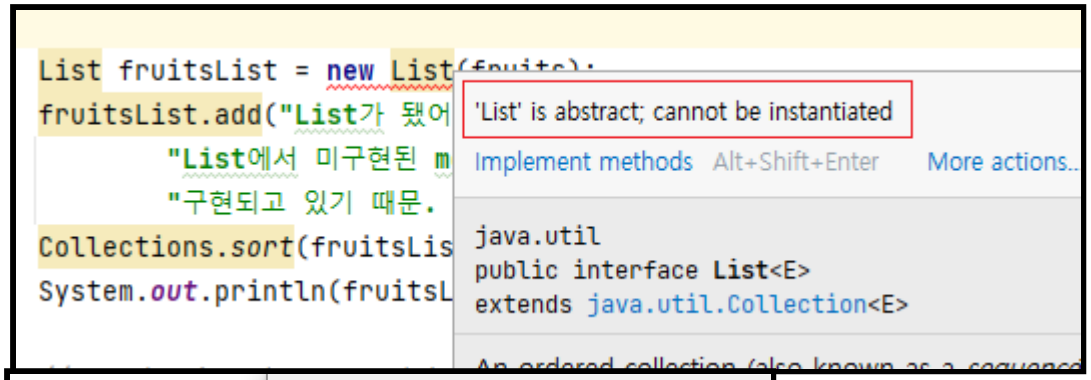
List는 Collection을 상속받음.

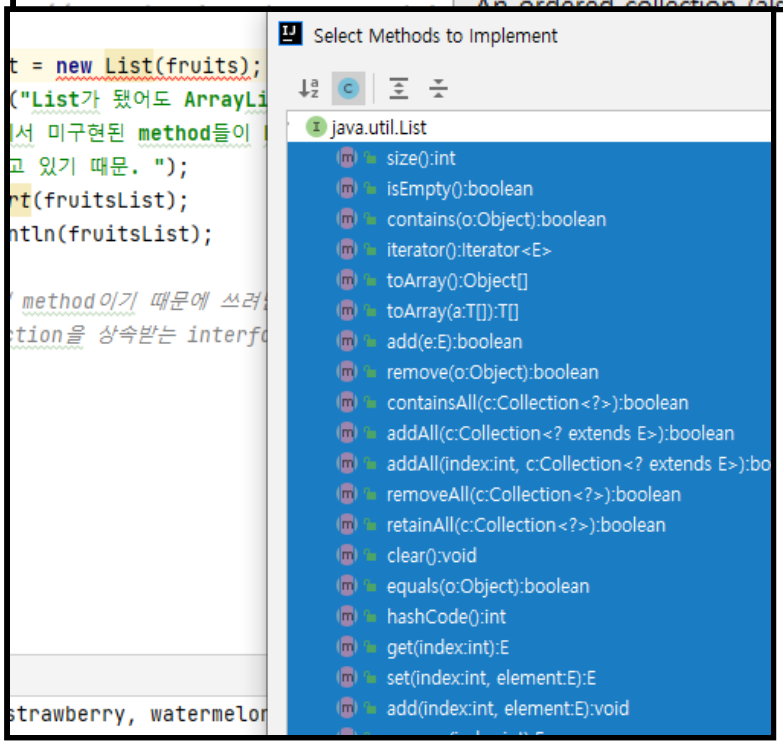
List들의 method들은 미구현되어있음. ›› 추상화 필요.

›› ArrayList에서 구현(추상화)되어 있음.

new ArrayList에서 Array를 지우면

"List는 추상적이다. 설명될 수 없다." 라는 error가 뜸. ›› 즉 추상화 필요.





'implement methods' 눌러보면

저거 다 추상화시키라고 함..

그래서 이미 추상화되어있는

ArrayList를 사용.

```java
public class _2nd_HashSetEdu {
    public static void main(String[] args) {
        Set<String> food = new HashSet<String>();
        // = HashSet<String> food = new HashSet<String>();
        food.add("우유");
```

Set과 HashSet의 관계도 같은 이치.

HashSet은 Interface 'Set'을 implements한다.

```java
public interface Set<E> extends Collection<E> {
    // Query Operations

    Returns the number of elements in this set (its cardinal
    MAX_VALUE elements, returns Integer.MAX_VALUE.
    Returns: the number of elements in this set (its cardina
    int size();

    Returns true if this set contains no elements.
    Returns: true if this set contains no elements
    boolean isEmpty();

    Returns true if this set contains the specified element.
    this set contains an element e such that Objects.equ
    Params: o – element whose presence in this set is to b
    Returns: true if this set contains the specified element
    Throws: ClassCastException – if the type of the sp
            (optional)
            NullPointerException – if the specified el
            elements (optional)
    boolean contains(Object o);
```

Interface 'Set' 역시 Interface 'List'처럼 미구현화된 method들의 구현(추상화)이 필요함.
HashSet에서 그 것들이 추상화되어 있다.

```java
public class HashSet<E>
    extends AbstractSet<E>
    implements Set<E>, Cloneable, java.io.Serializable
{
    @java.io.Serial
    static final long serialVersionUID = -5024744406713321676L;

    private transient HashMap<E,Object> map;

    // Dummy value to associate with an Object in the backing Map
    private static final Object PRESENT = new Object();

    Constructs a new, empty set; the backing HashMap instance has default initial capacity (16) and load
    factor (0.75).
    public HashSet() {
        map = new HashMap<>();
    }

    Constructs a new set containing the elements in the specified collection. The HashMap is created
    with default load factor (0.75) and an initial capacity sufficient to contain the elements in the specified
    collection.
    Params: c – the collection whose elements are to be placed into this set
    Throws: NullPointerException – if the specified collection is null
    public HashSet( @NotNull @Flow(sourceIsContainer = true, targetIsContainer = true) Collection<? extends E> c) {
        map = new HashMap<>(Math.max((int) (c.size()/.75f) + 1, 16));
        addAll(c);
    }
```

```
HashSet<String> food = new Set<String>();
 // = HashSet<String> food = new HashSet<String>();
```

따라서 이런 코드는 당연히 error

```
public static void main(String[] args) {
    Set<String> food = new Set<String>();
     // = HashSet<String> food = new HashSet<String>();
    food.add("우유");
```

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Set;

public class _99th_Sorting {
```

```
import java.util.*;

public class _99th_Sorting {
```

utility class들 import 하다보니까 4개에서 5개로 넘어갈 때 저렇게 *로 표시됨. 많이 import하면 그런듯?

〈Quiz.60〉

```java
public class _3rd_Quiz60 {
    public static void main(String[] args) {
     /*
     아무 숫자 10개 설정.
     이 난수들을 가지고 100개의 data를 마구잡이로 생성한다.
     각각의 데이터들이 몇 개씩 중복되었는지 프로그래밍.
     그리고 이 정보들을 정렬
     */
        // HashSet이용하는 방법으로 풀이
        // 아무 숫자 10개를 int[]에 setting 하고
        // 그 배열을 HashSet으로 만든다.
        // 그리고 HashMap을 이용해서 key 값에 10개의 숫자들을 setting하고
        // value 초기값을 0으로 하고 중복될 때마다 1씩 증가(++cnt)하는 method를 만들고
        // 그 method를 100번 돌린다.

        int[] testSet = {100, 200, 5, 15, 22, 50, 1245, 23331, 77777, 77};
        Duplicate_check dc = new Duplicate_check(testSet);
        dc.allocRandomDuplicate( num: 100);
        System.out.println("MAP: " + dc.getDuplicateCheckMap());
```

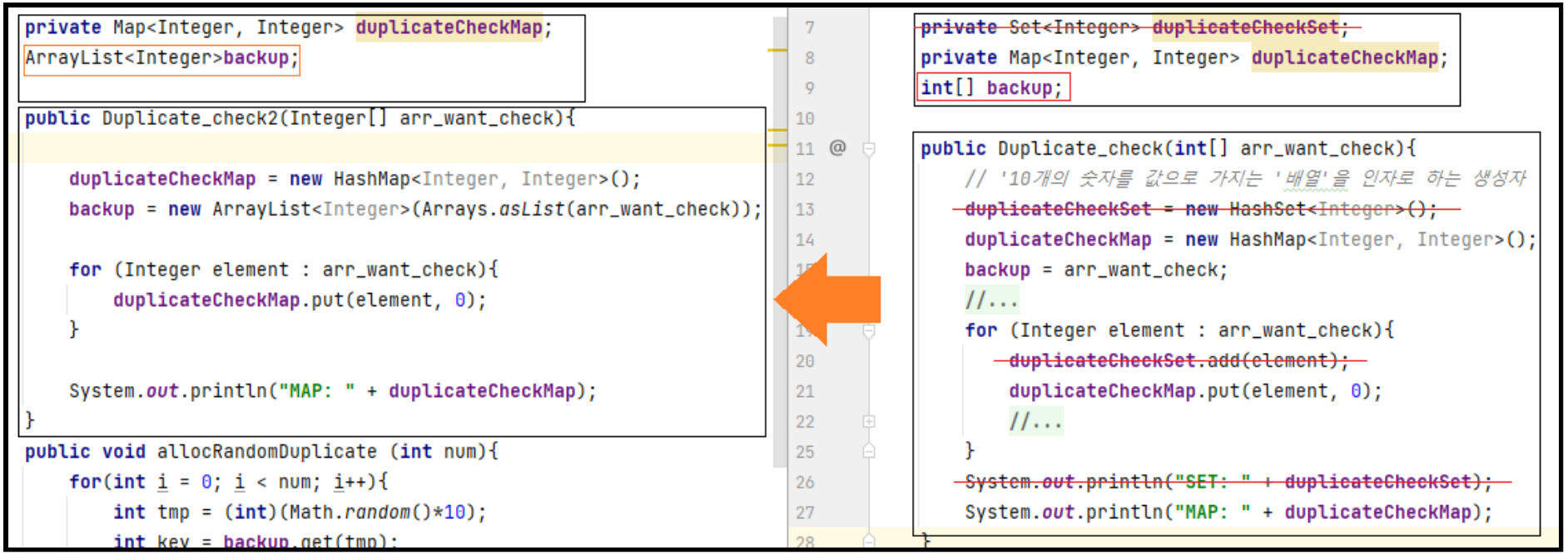
```java
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;
class Duplicate_check{

    private Set<Integer> duplicateCheckSet;
    private Map<Integer, Integer> duplicateCheckMap;
    int[] backup;

    public Duplicate_check(int[] arr_want_check){
        // '10개의 숫자를 값으로 가지는 '배열'을 인자로 하는 생성자
        duplicateCheckSet = new HashSet<Integer>();
        duplicateCheckMap = new HashMap<Integer, Integer>();
        backup = arr_want_check;
        // 인자로 받은 배열을 백업할 수 있는 'backup' 배열을 만든다면
        // 굳이 HashSet을 이용할 필요가 있을까??
        // HashSet 없이 푸는 방법 >> 3rd_Quiz_withoutHashSet

        for (Integer element : arr_want_check){
            duplicateCheckSet.add(element);
            duplicateCheckMap.put(element, 0);
            // HashSet과 HashMap setting
            // HashMap의 key값은 10개의 숫자들로 setting되고
            // 그 value의 초기값을 0으로 setting한다.
        }
        System.out.println("SET: " + duplicateCheckSet);
        System.out.println("MAP: " + duplicateCheckMap);
    }
```

```java
public void allocRandomDuplicate (int num){
    // num번을 실행시켜서 중복이 몇 번 나오는지 알아보는 method.
    for(int i = 0; i < num; i++){
        int tmp = (int)(Math.random()*10);
        int key = backup[tmp];
        // backup에 저장되어 있는 숫자를 random으로 뽑는 code
        // 뽑힌 수를 int key에 대입.

        System.out.printf("%6d", key);
        if (i % 20 == 19){ // 20개씩 나열 후 줄바꾸기
            System.out.println();
        }
        //if(duplicateCheckSet.contains(key)){
        // if문을 굳이 돌릴 필요가 없는 것 같다.

            int cnt = duplicateCheckMap.get(key);
            duplicateCheckMap.put(key, ++cnt);
        // 지금까지 count되어 있는 value값을 불러와서(초기에는 0부터 시작)
        // +1을 해주고 다시 그 값을 해당 key에 맞는 value에 대입.
        }
    }
public Map<Integer, Integer> getDuplicateCheckMap() { return duplicateCheckMap; }
```

중복되는 횟수

```
"C:\Program Files\Java\jdk-16\bin\java.exe" -javaagent:C:\Users\Samuel\AppData\Local\JetBrains\Toolbox\apps\IDEA-C\ch-0\2
SET: [77777, 50, 23331, 100, 5, 22, 200, 1245, 77, 15]
MAP: {77777=0, 50=0, 23331=0, 100=0, 5=0, 22=0, 200=0, 1245=0, 77=0, 15=0}
    22 23331   200    15 77777 77777   200     5   100    22 23331    77    50    77     5     5     5    22    77    50
     5  1245 23331    22 23331  1245  1245    77 77777    50    15   100   100    50 23331    77     5   100    50 23331
   100  1245    15    77 77777    77   100   200     5   200    22    15    77  1245    77    15     5    50   100    15
    22   100    22    22    22    22    15    50 23331    15    22  1245    77 77777  1245 77777    77 23331     5    50
     5    15   100 23331  1245  1245    77 77777     5 77777 77777   100   100 23331    15    77    50   200     5 23331
MAP: {77777=9, 50=9, 23331=11, 100=11, 5=12, 22=11, 200=5, 1245=9, 77=13, 15=10}
```

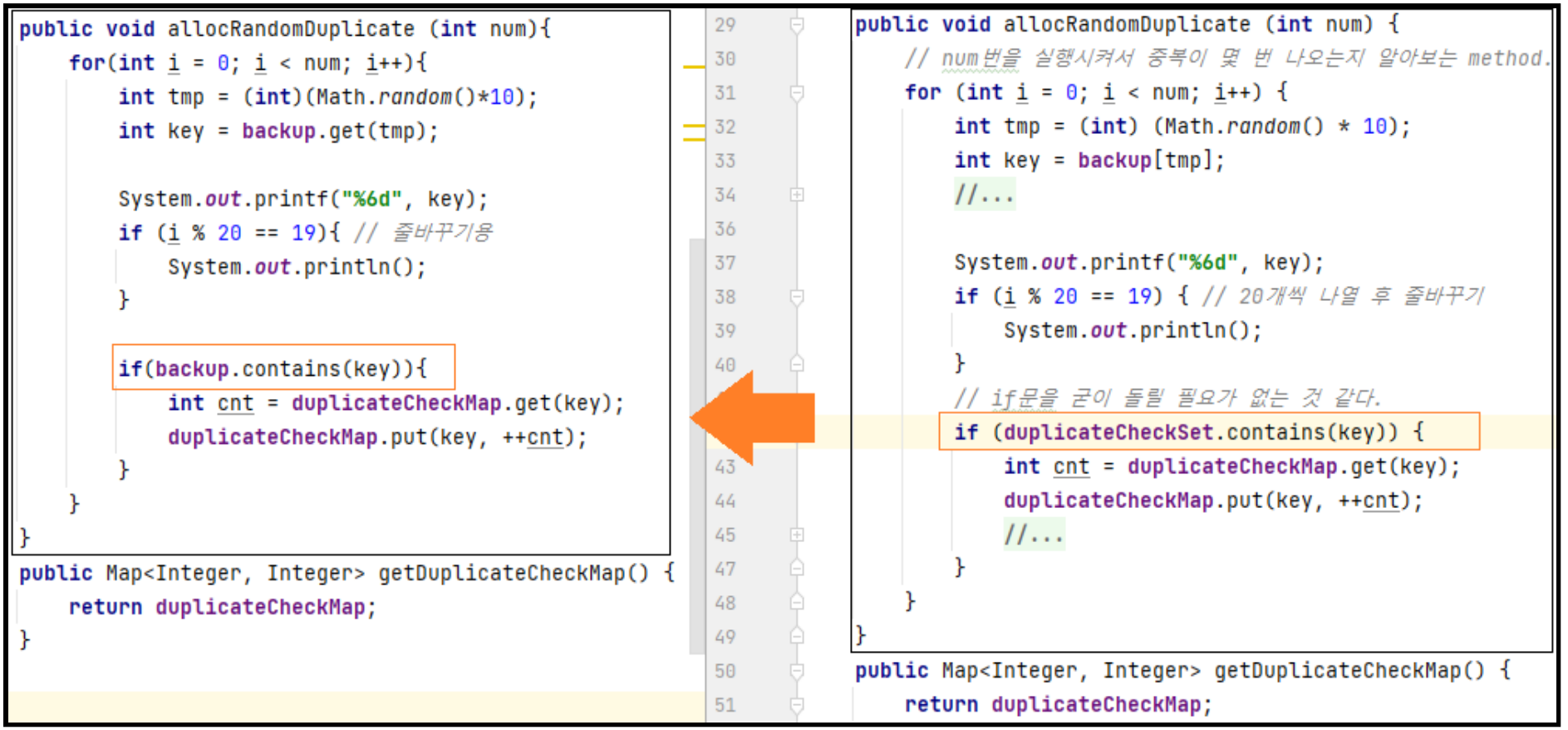
〈Quiz.60 without HashSet〉



```
private Map<Integer, Integer> duplicateCheckMap;
ArrayList<Integer>backup;

public Duplicate_check2(Integer[] arr_want_check){

    duplicateCheckMap = new HashMap<Integer, Integer>();
    backup = new ArrayList<Integer>(Arrays.asList(arr_want_check));

    for (Integer element : arr_want_check){
        duplicateCheckMap.put(element, 0);
    }

    System.out.println("MAP: " + duplicateCheckMap);
}
public void allocRandomDuplicate (int num){
    for(int i = 0; i < num; i++){
        int tmp = (int)(Math.random()*10);
        int key = backup.get(tmp);
```

```
7   private Set<Integer> duplicateCheckSet;
8   private Map<Integer, Integer> duplicateCheckMap;
9   int[] backup;
10
11  public Duplicate_check(int[] arr_want_check){
12      // '10개의 숫자를 값으로 가지는 '배열'을 인자로 하는 생성자
13      duplicateCheckSet = new HashSet<Integer>();
14      duplicateCheckMap = new HashMap<Integer, Integer>();
15      backup = arr_want_check;
16      //...
17      for (Integer element : arr_want_check){
20          duplicateCheckSet.add(element);
21          duplicateCheckMap.put(element, 0);
22          //...
25      }
26      System.out.println("SET: " + duplicateCheckSet);
27      System.out.println("MAP: " + duplicateCheckMap);
28  }
```

allocRandomDuplicate()에서 contains()를 사용하기 위해 backup을 단순 배열이 아니라 ArrayList형으로 만듦



```java
public void allocRandomDuplicate (int num){
    for(int i = 0; i < num; i++){
        int tmp = (int)(Math.random()*10);
        int key = backup.get(tmp);

        System.out.printf("%6d", key);
        if (i % 20 == 19){ // 줄바꾸기용
            System.out.println();
        }

        if(backup.contains(key)){
            int cnt = duplicateCheckMap.get(key);
            duplicateCheckMap.put(key, ++cnt);
        }
    }
}
public Map<Integer, Integer> getDuplicateCheckMap() {
    return duplicateCheckMap;
}
```

```java
public void allocRandomDuplicate (int num) {
    // num번을 실행시켜서 중복이 몇 번 나오는지 알아보는 method.
    for (int i = 0; i < num; i++) {
        int tmp = (int) (Math.random() * 10);
        int key = backup[tmp];
        //...

        System.out.printf("%6d", key);
        if (i % 20 == 19) { // 20개씩 나열 후 줄바꾸기
            System.out.println();
        }
        // if문을 굳이 돌릴 필요가 없는 것 같다.
        if (duplicateCheckSet.contains(key)) {
            int cnt = duplicateCheckMap.get(key);
            duplicateCheckMap.put(key, ++cnt);
            //...
        }
    }
}
public Map<Integer, Integer> getDuplicateCheckMap() {
    return duplicateCheckMap;
```

정리하다보니 위에 첫번째 풀이 정리처럼 if문을 쓸 이유가 없기 때문에

굳이 backup을 ArrayList로 만들 이유도 없긴했었지만.. 이왕 한 김에 정리.

(= 그냥 backup을 원래 풀이처럼 일반 배열로 두고 if문 만 지우면 됨)

>> Main 부분은 똑같기 때문에 따로 정리 x

〈Quiz.61 Thread〉 - 강사님 풀이 듣고 20210608.pdf에 정리.