(디지털컨버전스)스마트 콘텐츠와 웹 융합응용SW개발자 양성과정

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

2021년 6월 7일 질문노트

[김중연]

```java
//숫자값 2400, 5000, 그리고 아무 숫자나 8개 정도 추가한다.
//이 난수들(총 8개)을 가지고 1000개의 데이터를 마구잡이로 생성한다.
//각각의 데이터들이 몇 개씩 중복 되었는지 프로그래밍 해보도록 하자!
//그리고 이 정보들을 정렬해보자
import java.util.*;

class FrequencyChecker {
    Set<Integer> frequencySet; //전역변수로 지정만하고 값은 생성자에서 초기화한다
    Map<Integer, Integer> frequencyMap;
    int[] backUp;  // frequencySet을 직접 key에 대입하면 에러걸려서 만듬

    public FrequencyChecker (int[] arr) {  //생성자에 값을 초기화 해준다  int[] arr에 testSet을 대입한다
        frequencySet = new HashSet<Integer>();
        frequencyMap = new HashMap<Integer, Integer>();  //HashMap의 특징은 (Key, Value)를 입력하고 key값으로 value값을 얻는다

        backUp = arr;

        for (Integer elem : arr) {  //frequencySet에 인덱스값을  frequencyMap의 key값에 인덱스값을 지정한다
            frequencySet.add(elem);
            frequencyMap.put(elem, 0); //put은 HashMap의 (Key, Value)을 입력하는 역할을 한다
        }

        System.out.println("frequencySet: " + frequencySet);  //frequencySet 값이 제대로 입력됐는지 확인한다
        System.out.println("frequencyMap: " + frequencyMap);  //frequencyMap 값이 제대로 입력됐는지 확인한다
    }

    public void allocRandomFrequency (int num) {  //지역변수 num을 만든다
        for (int i = 0; i < num; i++) {
            int tmp = (int) (Math.random() * 10);
            int key = backUp[tmp];
```

Set과 Map을 사용하는
코드를 연습할 필요가 있다.

```
            System.out.printf("%6d", key);  //6은 간격을 6칸 띄우는 것을 의미한다

            // 0~19: 20개
            // 20~39: 20개
            if (i % 20 == 19) {  // 20개씩 나열하기
                System.out.println();
            }

            if (frequencySet.contains(key)) {
                int cnt = frequencyMap.get(key); //Map에 나왔던 인덱스 값을 cnt에 대입한다
                frequencyMap.put(key, ++cnt);  // 인덱스값이 나올때마다 +1을 하여 얼마나 나왔는지 확인한다
            }
        }
    }

    public Map<Integer, Integer> getFrequencyMap() { return frequencyMap; }
}

public class Prob60 {
    public static void main(String[] args) {
        int[] testSet = {
                2400,  5000, 1000, 200, 6000,
                77000, 434,  768,  20,  50
        };

        FrequencyChecker fc = new FrequencyChecker(testSet);

        fc.allocRandomFrequency( num: 1000);  //메소드의 소괄호안에 변수를 만든 후 값을 설정한다

        System.out.println(fc.getFrequencyMap());  //각 인덱스 값이 얼마나 나왔는지 출력한다
```

임익환씨 코드보면서 작성한 코드

```java
//숫자값 2400, 5000, 그리고 아무 숫자나 8개 정도 추가한다.
//이 난수들(총 8개)을 가지고 1000개의 데이터를 마구잡이로 생성한다.
//각각의 데이터들이 몇 개씩 중복 되었는지 프로그래밍 해보도록 하자!
//그리고 이 정보들을 정렬해보자


import java.util.Arrays;

public class NewProb60 {
    public static void main(String[] args) {
        int[] testSet = {2400, 5000, 1,2,3,4,5,6,7,8};
        int[] number = new int[1000];
        int[] count = new int[10];

        for(int i=0; i<1000; i++){
            int j=(int)(Math.random() * 10);
            number[i]= testSet[j];
            if(number[i]==testSet[j]);{
                count[j]++;  //중복되는 만큼 카운트시킴
            }
        }
        Arrays.sort(testSet);  //Arrays.sort : 전달받은 배열의 모든 요소를 오름차순으로 정렬
        for(int i=0; i<10; i++){
            System.out.println(testSet[i] +"중복개수"+count[i]);
        }
    }
}
```

NetworkUrl에 관한 간단한 설명

```java
import java.net.MalformedURLException;
import java.net.URL;

public class NetworkUrlTest {
    // Malform 이라는것이 악성 코드에 해당해서
    // 이상한 URL로 링크를 태워서 공격을 할 수 있기 때문에 그것에 대한 방어 조치라 보면 됨
    // www.daum.net, http://www.daum.net
    // URL을 반드시 후자로 줘야 합니다.
    // 이유는 www.daum.net 으로 하면 위와 같이 악성코드 공격이 가능함
    public static void main(String[] args) throws MalformedURLException {
        URL myURL = new URL( spec: "http://www.loanconsultant.or.kr/source/index.jsp?t=20191216");

        // Protocol: HTTP(웹 애플리케이션 전용 프로토콜입니다)
        System.out.println("Protocol = " + myURL.getProtocol());
        System.out.println("authority = " + myURL.getAuthority());
        System.out.println("host = " + myURL.getHost());
        System.out.println("port = " + myURL.getPort());
        System.out.println("path = " + myURL.getPath());
        System.out.println("query = " + myURL.getQuery());
        System.out.println("filename = " + myURL.getFile());
        System.out.println("ref = " + myURL.getRef());
    }
}

// int[] test = { 2400, 5000, 123, 123245, 23542345648, 234923, 23492334, 2349 ..... }
// 풀이 방식: 1. HashSet, 2. 라이브러리 없이 통으로 만들기, 3. 다른 Collection 사용해서
```

```java
import java.util.*;

public class SortingTest {
    public static void main(String[] args) {
        String[] sample = {"I", "walk", "the", "line", "Apple", "hit", "me", "Ground", "attack", "you"};

        List<String> list = Arrays.asList(sample);

        // 정렬 법칙(대문자 우선, 그 다음 소문자)
        Collections.sort(list);

        System.out.println(list);

        Integer[] numbers = {1, 2, 3, 100, 77, 2342, 2342354, 345, 12323, 12, 4};

        List<Integer> numList = Arrays.asList(numbers);

        Collections.sort(numList);

        System.out.println(numList);

        Set fruits = new HashSet();

        fruits.add("strawberry");
        fruits.add("watermelon");
        fruits.add("grape");
        fruits.add("orange");
        fruits.add("apple");
        fruits.add("banana");

        List fruitsList = new ArrayList(fruits);
        fruitsList.add("ofcourse");

        Collections.sort(fruitsList);

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

Sort는 오름차순정렬을 하는 키워드이다

```java
public class AsynchronousPatternDoc {
}


/* 여기서 가져가야할 주요 개념
1. Thread를 활용하는 이유는 성능을 빠르게 만들기 위함이다.
2. 비동기 패턴(Asynchronous Pattern)이란 전부 Thread를 기반으로 한다.
3. 자바 스크립트 또한 Multi Thread 모델을 지원한다(자체적으로)
   (이건 최신 자바스크립트 ECMA 6 부터 서포트인것 같음) - Promise를 활용하여 증명
4. Thread를 사용할 때는 Critical Section에 대한 방어가 무엇보다도 중요하다(데이터 무결성)
5. 또한 스레드는 비동기 처리를 하기 때문에 데이터의 완전한 전송을 보장하지 못할 수도 있다.
   (말이 좀 어려운데 이 부분은 자바스크립트의 Promise를 통해 살펴볼 예정)
ex) 전화 통화: 동기 처리
    왜 ? 친구한테 전화를 걸었음. 친구가 통화 허용을 안하면 통화가 안됨
ex) 카카오톡 메시지: 비동기 처리
    왜 ? 상대방이 확인하던 안하던 난 보낸다.
    나는 니가 뭘 하던 내 할 일을 하겠다.
 */
```

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

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

        test.add(3);
        test.add(7);
        test.add(1);

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