Map<K, V> 컬렉션 클래스들

Map<K, V> 컬렉션 클래스들

- Key-Value 방식의 데이터 저장과 HashMap<K, V> 클래스
  - 캐비넷에 서류철을 보관할 때 해당 서류철을 쉽게 찾을 수 있도록 서류철의 특정 위치에 서류의 정보나 이름 써 넣으며, 이는 Key와 Value가 하나의 쌍을 이루는 데이터 저장 방식이라 할 수 있음
  - Key는 실질적 데이터가 아니며 데이터 Value를 찾는 지표
  - 데이터 Value를 저장할 때 Key를 함께 저장하므로, Key는 중복될 수 없음
  - 즉, Key는 지표이므로 중복될 수 없지만 Key만 다르면 Value는 중복되어도 상관 없음

```
import java.util.HashMap;

public class HashMapCollection {
    public static void main(String[] args) {
        HashMap<Integer, String> map = new HashMap<>();
        // 데이터 저장
        map.put(45, "Brown");
        map.put(37, "James");
        map.put(23, "Martin");
        // 데이터 검색
        System.out.println("23번 : " + map.get(23));
        System.out.println("37번 : " + map.get(37));
        System.out.println("45번 : " + map.get(45));
        // 데이터 삭제
        map.remove(37);
        System.out.println("37번 : " + map.get(37));
    }
}
```

  - Key와 Value 모두 인스턴스
  - HashSet<E>는 해쉬 알고리즘을 기반으로 구현되어 있으며, HashMap<K, V> 역시 해쉬 알고리즘을 기반으로 구현되어 있어 정렬 상태를 유지하지 않음

Map<K, V> 컬렉션 클래스들

- HashMap<K, V>의 순차적 접근 방법
  - Iterable<T> 인터페이스를 구현하지 않아 for-each문 등을 통해 순차 접근 불가능
  - 대신 "public Set<K> keySet()" 메소드를 통해 Key를 모두 Set<E> 인스턴스로 반환하며 이를 통해 순차 접근 가능

```
public class HashMapIteration {
    public static void main(String[] args) {
        HashMap<Integer, String> map = new HashMap<>();
        // 데이터 저장
        map.put(45, "Brown");
        map.put(37, "James");
        map.put(23, "Martin");

        // Key만 담고 있는 컬렉션 인스턴스 생성
        Set<Integer> ks = map.keySet();

        // 전체 Key 출력
        for (Integer n : ks) {
            System.out.println(n.toString());
        }
        System.out.println();

        // 전체 Value 출력
        for (Integer n : ks) {
            System.out.println(map.get(n).toString());
        }
        System.out.println();

        // 전체 Value 출력
        for(Iterator<Integer> itr = ks.iterator(); itr.hasNext();) {
            System.out.println(map.get(itr.next()));
        }
    }
}
```

  - Set<E>는 Iterable<E>를 상속하므로 for-each문 등을 통해 순차 접근 진행

Map<K, V> 컬렉션 클래스들

- TreeMap<K, V>의 순차적 접근 방법
  - TreeSet<E>이 트리 자료구조를 기반으로 구현되어 있어서 정렬 상태를 유지하듯이 TreeMap<K, V> 역시 트리 자료구조를 기반으로 구현되어 있어서 정렬 상태를 유지

```
public class TreeMapIteration {
    public static void main(String[] args) {
        TreeMap<Integer, String> map = new TreeMap<>();

        // 데이터 저장
        map.put(45, "Brown");
        map.put(37, "James");
        map.put(23, "Martin");

        // Key만 담고 있는 컬렉션 인스턴스 생성
        Set<Integer> ks = map.keySet();

        // 전체 Key 출력
        for (Integer n : ks) {
            System.out.println(n.toString());
        }
        System.out.println();

        // 전체 Value 출력
        for (Integer n : ks) {
            System.out.println(map.get(n).toString());
        }
        System.out.println();

        // 전체 Value 출력
        for(Iterator<Integer> itr = ks.iterator(); itr.hasNext();) {
            System.out.println(map.get(itr.next()));
        }
    }
}
```