15장. 컬렉션 프레임워크

이것이 자바다(http://cafe.naver.com/thisjava)

한빛미디어 Hanbit Media, Inc.

Contents

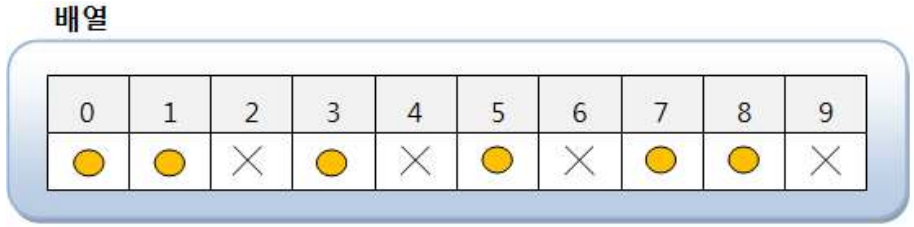
1절. 컬렉션 프레임워크 소개

❖ 컬렉션 프레임워크(Collection Framework)

- 컬렉션
  - 사전적 의미로 요소(객체)를 수집해 저장하는 것

- 배열의 문제점
  - 저장할 수 있는 객체 수가 배열을 생성할 때 결정
    → 불특정 다수의 객체를 저장하기에는 문제

  - 객체 삭제했을 때 해당 인덱스가 비게 됨
    → 낱알 빠진 옥수수 같은 배열
    → 객체를 저장하려면 어디가 비어있는지 확인해야



1절. 컬렉션 프레임워크 소개

❖ 컬렉션 프레임워크(Collection Framework)

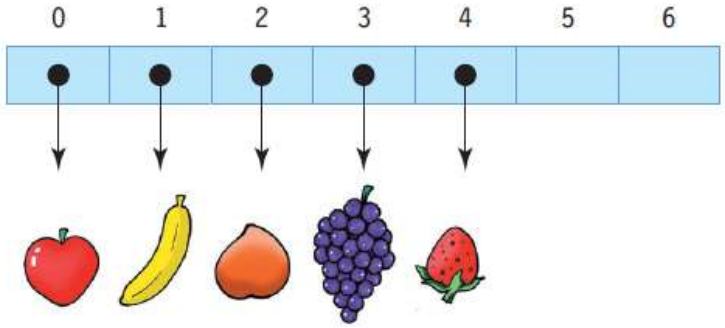
- 객체들을 효율적으로 추가, 삭제, 검색할 수 있도록 제공되는 컬렉션 라이브러리
- java.util 패키지에 포함
- 인터페이스를 통해서 정형화된 방법으로 다양한 컬렉션 클래스 이용

컬렉션(collection)의 개념

❖ 컬렉션

- **요소(element)라고 불리는 가변 개수의 객체들의 저장소**
  - 객체들의 컨테이너라고도 불림
  - 요소의 개수에 따라 크기 자동 조절
  - 요소의 삽입, 삭제에 따른 요소의 위치 자동 이동
- **고정 크기의 배열을 다루는 어려움 해소**
- **다양한 객체들의 삽입, 삭제, 검색 등의 관리 용이**

배열(array)



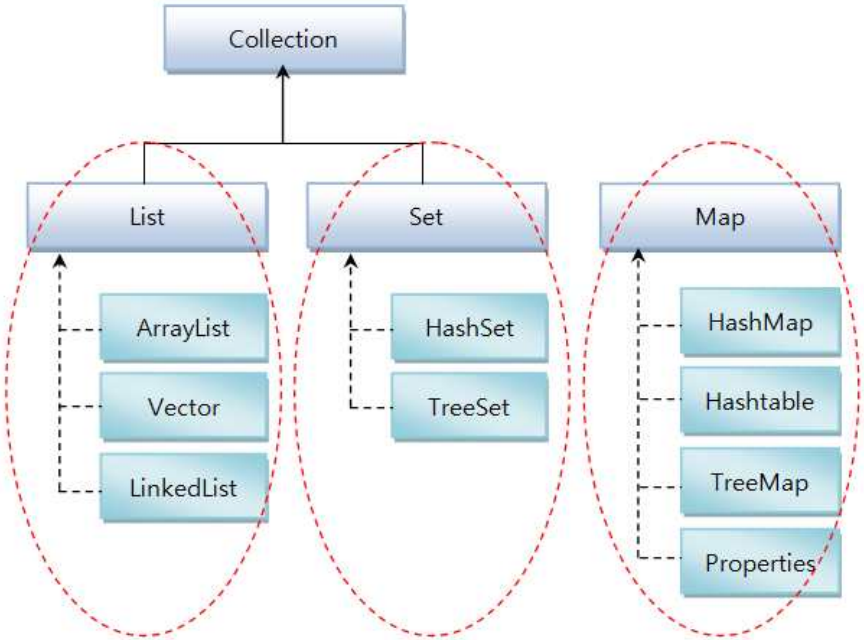
- 고정 크기 이상의 객체를 관리할 수 없다.
- 배열의 중간에 객체가 삭제되면 응용프로그램에서 자리를 옮겨야 한다.

컬렉션(collection)

- 가변 크기로서 객체의 개수를 염려할 필요 없다.
- 컬렉션 내의 한 객체가 삭제되면 컬렉션이 자동으로 자리를 옮겨준다.

1절. 컬렉션 프레임워크 소개

❖ 컬렉션 프레임워크의 주요 인터페이스



인터페이스 분류		특징	구현 클래스
Collection	List 계열	- 순서를 유지하고 저장 - 중복 저장 가능	ArrayList, Vector, LinkedList
	Set 계열	- 순서를 유지하지 않고 저장 - 중복 저장 안됨	HashSet, TreeSet
Map 계열		- 키와 값의 쌍으로 저장 - 키는 중복 저장 안됨	HashMap, Hashtable, TreeMap, Properties

2절. List 컬렉션

❖ 리스트 계열 장단점

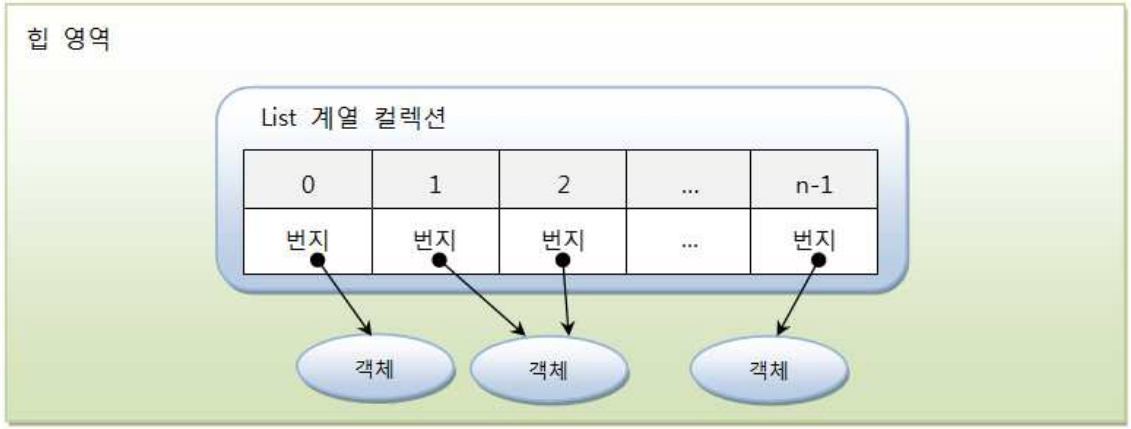
	속도	장점	단점
ArrayList		검색이 빠름	빈번한 추가 삭제시 성능이 떨어짐
Vector	가장느림	동기화 보장	속도가 느림
LinkedList	가장빠름	속도 빠름	데이터가 많아지면 검색이 느려짐

2절. List 컬렉션

❖ List 컬렉션의 특징 및 주요 메소드

- 특징
  - 인덱스로 관리
  - 중복해서 객체 저장 가능

- 구현 클래스
  - ArrayList
  - Vector
  - LinkedList



2절. List 컬렉션

❖ List 컬렉션의 특징 및 주요 메소드

▪ 주요 메소드

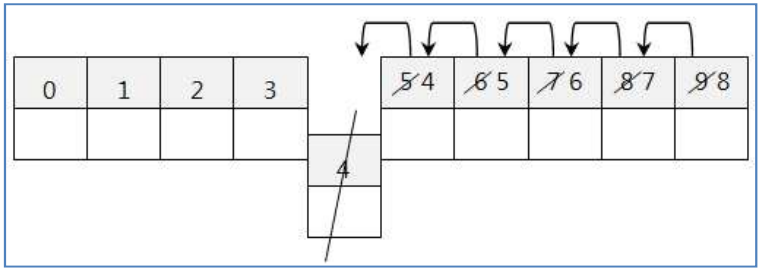
기능	메소드	설명
객체 추가	boolean add(E e)	주어진 객체를 맨끝에 추가
	void add(int index, E element)	주어진 인덱스에 객체를 추가
	set(int index, E element)	주어진 인덱스에 저장된 객체를 주어진 객체로 바꿈
객체 검색	boolean contains(Object o)	주어진 객체가 저장되어 있는지 여부
	E get(int index)	주어진 인덱스에 저장된 객체를 리턴
	isEmpty()	컬렉션이 비어 있는지 조사
	int size()	저장되어있는 전체 객체수를 리턴
객체 삭제	void clear()	저장된 모든 객체를 삭제
	E remove(int index)	주어진 인덱스에 저장된 객체를 삭제
	boolean remove(Object o)	주어진 객체를 삭제

2절. List 컬렉션

❖ ArrayList (p.725~729)

- ❖ 벡터와 달리 스레드 동기화 기능 없음
  - 저장 용량(capacity)
    - 초기 용량 : 10 (따로 지정 가능)
    - 저장 용량을 초과한 객체들이 들어오면 자동적으로 늘어남. 고정도 가능

  - 객체 제거
    - 바로 뒤 인덱스부터 마지막 인덱스까지 모두 앞으로 1씩 당겨짐



예제

```
import java.util.*;
public class ArrayListExample {
    public static void main(String[] args) {
        List<String> list = new ArrayList<String>();

        list.add("Java");
        list.add("JDBC");
        list.add("Servlet/JSP");
        list.add(2, "Database");
        list.add("iBATIS");

        int size = list.size();
        System.out.println("총 객체수: " + size);
        System.out.println();

        String skill = list.get(2);
        System.out.println("2: " + skill);
        System.out.println();

        for(int i=0; i<list.size(); i++) {
            String str = list.get(i);
            System.out.println(i + ":" + str);
        }
        System.out.println();

        list.remove(2);
        list.remove(2);
        list.remove("iBATIS");

        for(int i=0; i<list.size(); i++) {
            String str = list.get(i);
            System.out.println(i + ":" + str);
        }
    }
}
```

실습

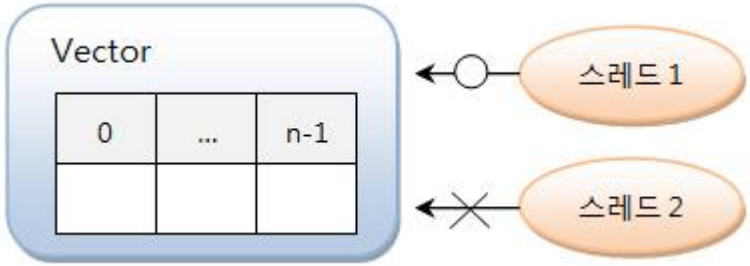
- 스트링 “푸” -” 르” -” 른” -” 바” -” 다” 가 들어 있는 ArrayList를 만드시오
  - ArrayList 출력
  - 르를 삭제
  - ArrayList 출력
  - 야 추가
  - ArrayList 출력

2절. List 컬렉션

❖ Vector

```
List<E> list = new Vector<E>();
```

- 특징
  - Vector는 스레드 동기화(synchronization)
    - 복수의 스레드가 동시에 Vector에 접근해 객체를 추가, 삭제하더라도 스레드에 안전(thread safe)



예제

```
import java.util.List;

public class VectorExample {
    public static void main(String[] args) {
        List<String> list = new Vector<String>();

        list.add("홍길동");
        list.add("김철수");
        list.add("박순이");
        list.add("홍경이");
        list.add("최루탄");

        list.remove(2);
        list.remove(3);

        for(int i=0; i<list.size(); i++) {
            String str = list.get(i);
            System.out.println(i+" : "+str);
        }
    }
}
```

실습

- 스트링형 1,2,3,4,5,6 가 들어 있는 Vector 객체 를 만드시오
  - Vector 출력
  - 2,4 제거
  - 10, 20, 30 추가
  - Vector 출력

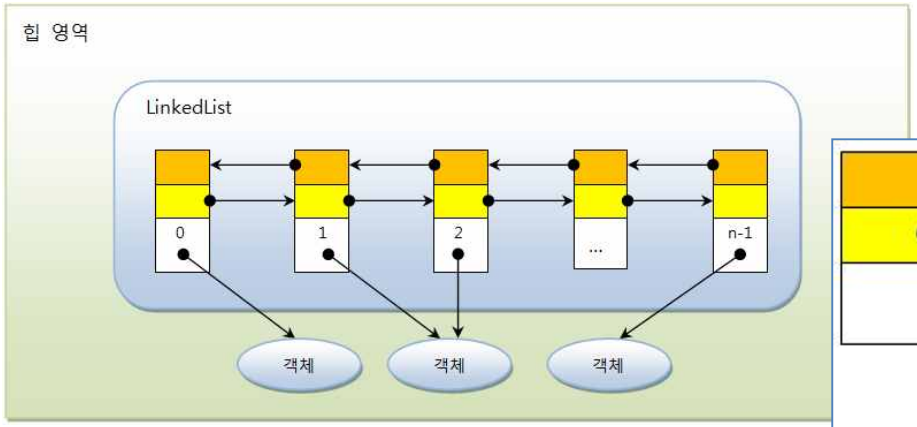
2절. List 컬렉션

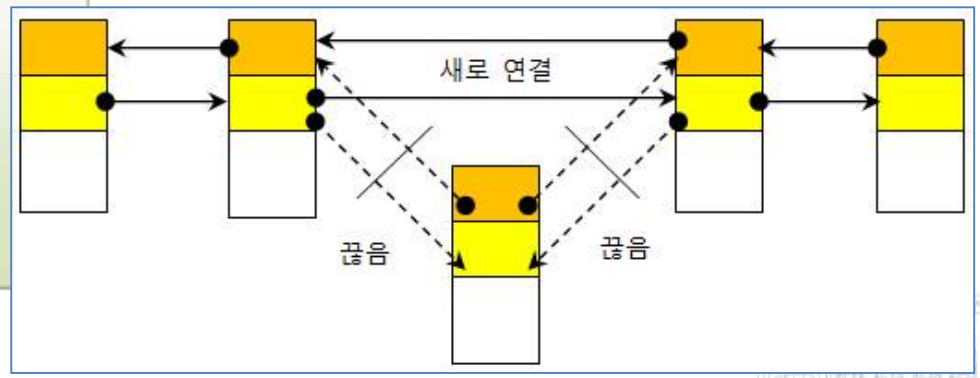
❖ LinkedList

```
List<E> list = new LinkedList<E>();
```

▪ 특징

- 인접 참조를 링크해서 체인처럼 관리
- 특정 인덱스에서 객체를 제거하거나 추가하게 되면 바로 앞뒤 링크만 변경
- 빈번한 객체 삭제와 삽입이 일어나는 곳에서는 ArrayList보다 좋은 성능





예제

```
import java.util.ArrayList;

public class LinkedListExample {
    public static void main(String[] args) {
        List<String> list1 = new ArrayList<String>();
        List<String> list2 = new LinkedList<String>();

        long startTime;
        long endTime;

        startTime = System.nanoTime();
        for(int i=0; i<10000; i++) {
            list1.add(0, String.valueOf(i));
        }
        endTime = System.nanoTime();
        System.out.println("ArrayList 걸린시간: " + (endTime-startTime) + " ns");

        startTime = System.nanoTime();
        for(int i=0; i<10000; i++) {
            list2.add(0, String.valueOf(i));
        }
        endTime = System.nanoTime();
        System.out.println("LinkedList 걸린시간: " + (endTime-startTime) + " ns");
    }
}
```

실습

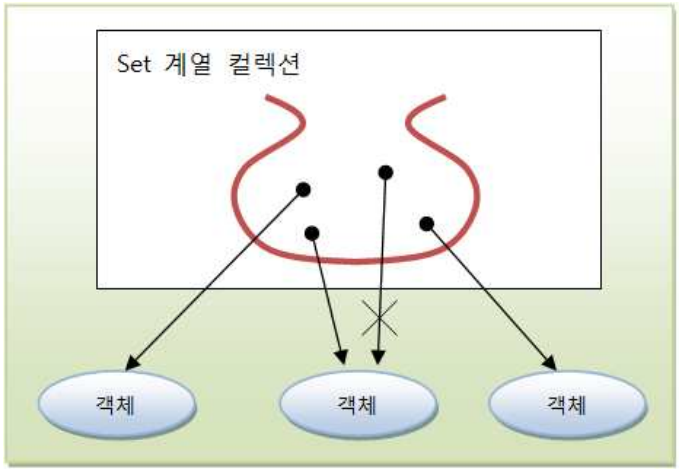
- for 문과 Math.randon() 메소드를 이용 LinkedList 에 10개의 수를 추가
- LinkedList 출력
- 1,3,5,7 번째 숫자 제거
- LinkedList 출력
- for 문과 Math.randon() 메소드를 이용 LinkedList 에 2개의 수를 추가
- LinkedList 출력

3절. Set 컬렉션

❖ Set 컬렉션의 특징 및 주요 메소드

- 특징
  - 수학의 집합에 비유
  - 저장 순서가 유지되지 않음
  - 객체를 중복 저장 불가
  - 하나의 null만 저장 가능

- 구현 클래스
  - HashSet, LinkedHashSet, TreeSet



3절. Set 컬렉션

❖ Set 컬렉션의 특징 및 주요 메소드

- 주요 메소드

기능	메소드	설명
객체 추가	boolean add(E e)	주어진 객체를 저장, 객체가 성공적으로 저장되면 true 를 리턴하고 중복 객체면 false 를 리턴
객체 검색	boolean contains(Object o)	주어진 객체가 저장되어 있는지 여부
	isEmpty()	컬렉션이 비어 있는지 조사
	Iterator<E> iterator()	저장된 객체를 한번씩 가져오는 반복자 리턴
	int size()	저장되어있는 전체 객체수 리턴
객체 삭제	void clear()	저장된 모든 객체를 삭제
	boolean remove(Object o)	주어진 객체를 삭제

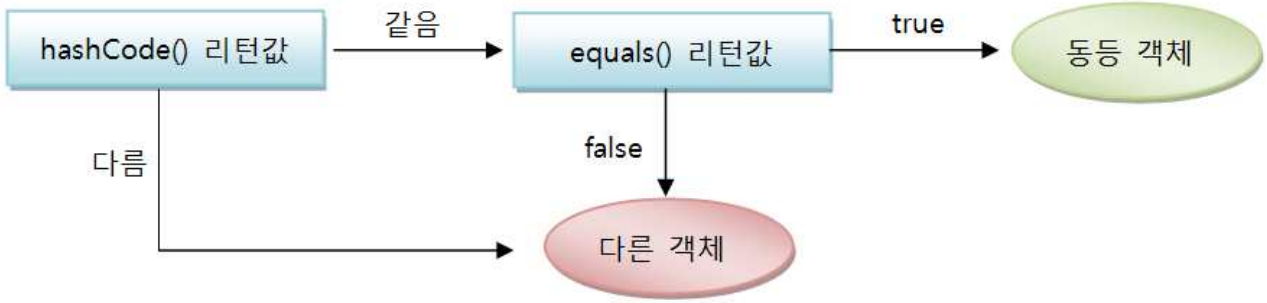
- 전체 객체 대상으로 한 번씩 반복해 가져오는 반복자(Iterator) 제공
  - 인덱스로 객체를 검색해서 가져오는 메소드 없음

3절. Set 컬렉션

❖ HashSet (p.736~739)

```
Set<E> set = new HashSet<E>();
```

- **특징**
  - **동일 객체 및 동등 객체는 중복 저장하지 않음**
  - **동등 객체 판단 방법**



예제

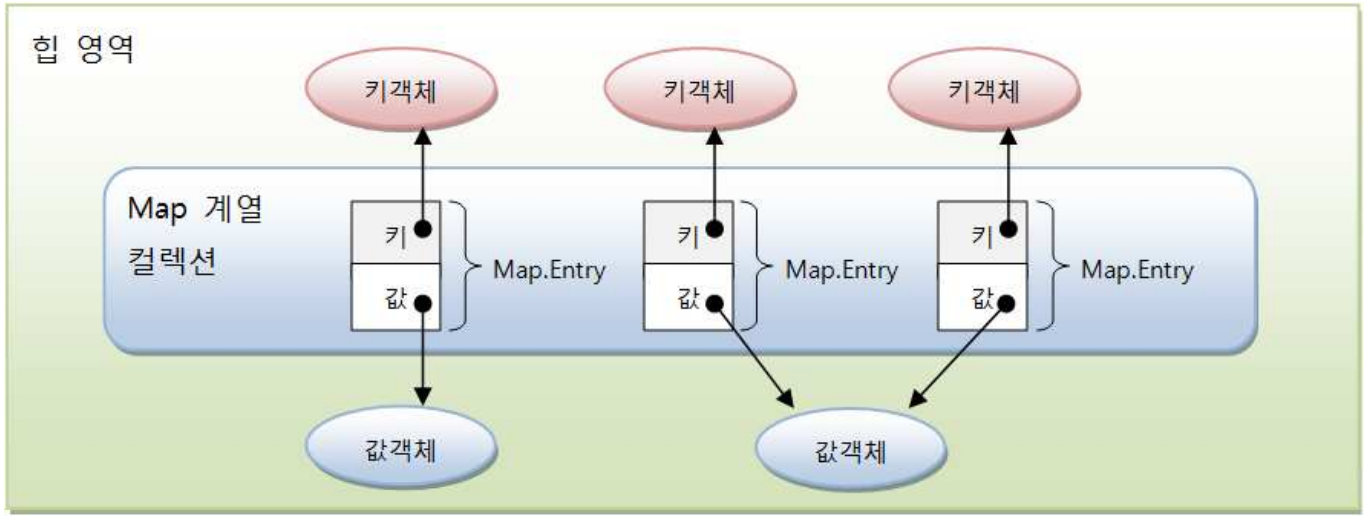
```
import java.util.*;

public class HashSetExample1 {
    public static void main(String[] args) {
        Set<String> set = new HashSet<String>();

        set.add("Java");
        set.add("JDBC");
        set.add("Servlet/JSP");
        set.add("Java");
        set.add("iBATIS");

        int size = set.size();
        System.out.println("총 객체수: " + size);

        Iterator<String> iterator = set.iterator();
        while(iterator.hasNext()) {
            String element = iterator.next();
            System.out.println("\t" + element);
        }

        set.remove("JDBC");
        set.remove("iBATIS");

        System.out.println("총 객체수: " + set.size());

        for(String element : set) {
            System.out.println("\t" + element);
        }

        set.clear();
        if(set.isEmpty()) { System.out.println("비어 있음"); }
    }
}
```

4절. Map 컬렉션

❖ Map 컬렉션의 특징 및 주요 메소드

- 특징
  - 키(key)와 값(value)으로 구성된 Map.Entry 객체를 저장하는 구조
  - 키와 값은 모두 객체
  - 키는 중복될 수 없지만 값은 중복 저장 가능

- 구현 클래스
  - HashMap, Hashtable, LinkedHashMap, Properties, TreeMap



4절. Map 컬렉션

HashMap	Hashtable
동기화 미지원	동기화 지원
HashMap 대비 빠름	HashMap 보다는 느림
키와 값으로 null이 허용	키와 값으로 null이 허용되지 않음

4절. Map 컬렉션

❖ Map 컬렉션의 특징 및 주요 메소드

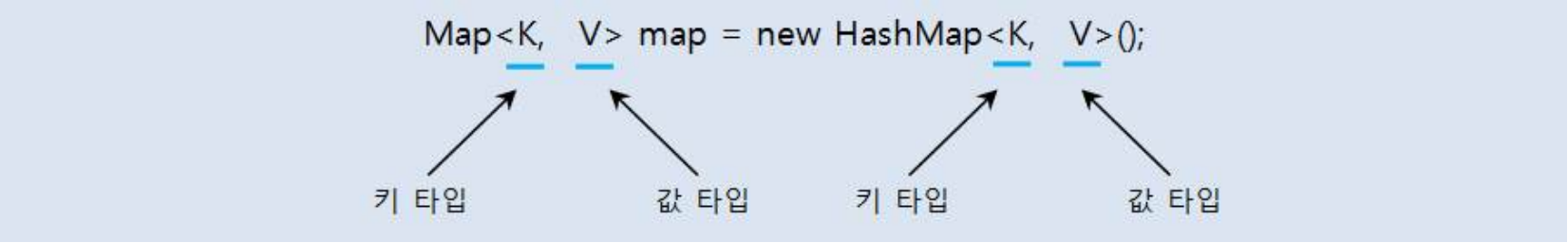
▪ 주요 메소드

기능	메소드	설명
객체 추가	V put(K key, V value)	주어진 키와 값을 추가, 저장이 되면 값을 리턴
객체 검색	boolean containsKey(Object key)	주어진 키가 있는지 여부
	boolean containsValue(Object value)	주어진 값이 있는지 여부
	Set<Map.Entry<K,V>> entrySet()	키와 값의 쌍으로 구성된 모든 Map.Entry 객체를 Set 에 담아서 리턴
	V get(Object key)	주어진 키의 값을 리턴
	boolean isEmpty()	컬렉션이 비어있는지 여부
	Set<K> keySet()	모든 키를 Set 객체에 담아서 리턴
	int size()	저장된 키의 총 수를 리턴
	Collection<V> values()	저장된 모든 값 Collection 에 담아서 리턴
객체 삭제	void clear()	모든 Map.Entry(키와 값)를 삭제
	V remove(Object key)	주어진 키와 일치하는 Map.Entry 삭제, 삭제가 되면 값을 리턴

4절. Map 컬렉션

❖ HashMap (p.742~745)

- 특징




```
Map<String, String> dic = new Hashtable<String, String>();
```

- 동기화 지원안함

예제


```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;

public class HashMapExample1 {
        public static void main(String[] args) {
                //Map 컬렉션 생성
                Map<String, Integer> map = new HashMap<String, Integer>();

                //객체 저장
                map.put("신용권", 85);
                map.put("홍길동", 90);
                map.put("동장군", 80);
                map.put("홍길동", 95);
                System.out.println("총 Entry 수: " + map.size());

                //객체 찾기
                System.out.println("₩t홍길동 : " + map.get("홍길동"));
                System.out.println();

                //객체를 하나씩 처리
                Set<String> keySet = map.keySet();
                Iterator<String> keyIterator = keySet.iterator();
```

```
while(keyIterator.hasNext()) {
                    String key = keyIterator.next();
                    Integer value = map.get(key);
                    System.out.println("₩t" + key + " : " + value);
                  }
                  System.out.println();

                  //객체 삭제
                  map.remove("홍길동");
                  System.out.println("총 Entry 수: " + map.size());

                  //객체를 하나씩 처리
                  Set<Map.Entry<String, Integer>> entrySet = map.entrySet();
                  Iterator<Map.Entry<String, Integer>> entryIterator = entrySet.iterator();
                  while(entryIterator.hasNext()) {
                    Map.Entry<String, Integer> entry = entryIterator.next();
                    String key = entry.getKey();
                    Integer value = entry.getValue();
                    System.out.println("₩t" + key + " : " + value);
                  }
                  System.out.println();

                  //객체 전체 삭제
                  map.clear();
                  System.out.println("총 Entry 수: " + map.size());
        }
}
```

실습

- **String형 HashMap 를 이용 다음 데이터를 저장하세요.**

key	value
apple	김철수
orange	박순이
mango	최순철
book	차돌이
shield	옥순자

  - **해시테이블 내용 출력**
  - **mango, book 삭제**
  - **해시테이블 내용 출력**
  - **다음의 데이터 추가**
    - chicken, 박근혜
    - mouse, 이명박

4절. Map 컬렉션

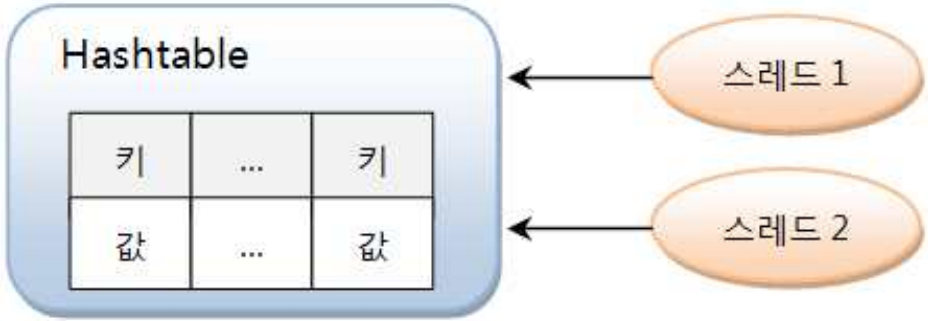
❖ Hashtable

```
Map<K, V> map = new Hashtable<K, V>();
```

키 타입 값 타입 키 타입 값 타입

```
Map<String, String> dic = new Hashtable<String, String>();
```

- 특징
  - 키 객체 만드는 법은 HashMap과 동일
  - Hashtable은 스레드 동기화(synchronization)가 된 상태
    - 복수의 스레드가 동시에 Hashtable에 접근해서 객체를 추가, 삭제 하더라도 스레드에 안전(thread safe)



스레드 동기화 적용됨

```
import java.util.*;
public class HashtableExample {
	public static void main(String[] args) {
		Map<String, String> map = new Hashtable<String, String>();
		map.put("spring", "12");
		map.put("summer", "123");
		map.put("fall", "1234");
		map.put("winter", "12345");
		Scanner scanner = new Scanner(System.in);
		while(true) {
			System.out.println("아이디와 비밀번호를 입력해주세요");
			System.out.print("아이디: ");
			String id = scanner.nextLine();
			System.out.print("비밀번호: ");
			String password = scanner.nextLine();
			System.out.println();
			if(map.containsKey(id)) {
				if(map.get(id).equals(password)) {
					System.out.println("로그인 되었습니다");
					break;
				} else {
					System.out.println("비밀번호가 불일치.");
				}
			} else {
				System.out.println("입력하신 아이디가 존재하지 않습니다");
			}
		}
	}
}
```

실습

- String형 HashTable 를 이용 다음 데이터를 저장하세요.

key	value
apple	김철수
orange	박순이
mango	최순철
book	차돌이
shield	옥순자

  - 해시테이블 내용 출력
  - mango, book 삭제
  - 해시테이블 내용 출력
  - 다음의 데이터 추가
    - chicken, 박근혜
    - mouse, 이명박

Thank You !

이것이 자바다(http://cafe.naver.com/thisjava)