컬렉션 프레임워크

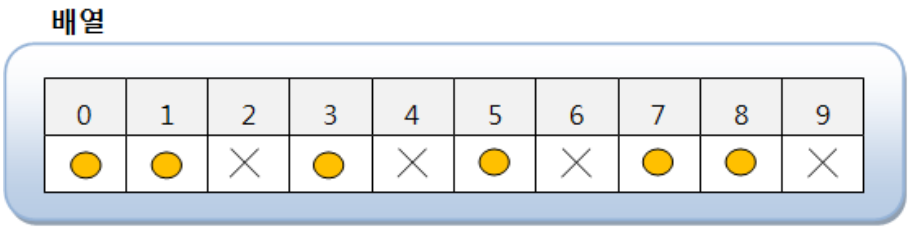
Contents

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



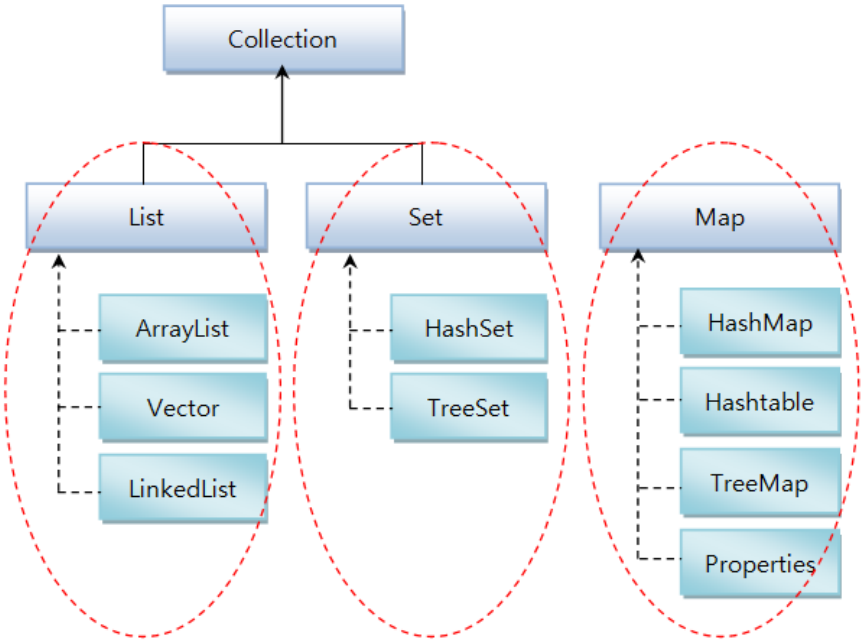
1절. 컬렉션 프레임워크 소개

❖ 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

1절. 컬렉션 프레임워크 소개

❖ 컬렉션 프레임워크의 주요 인터페이스



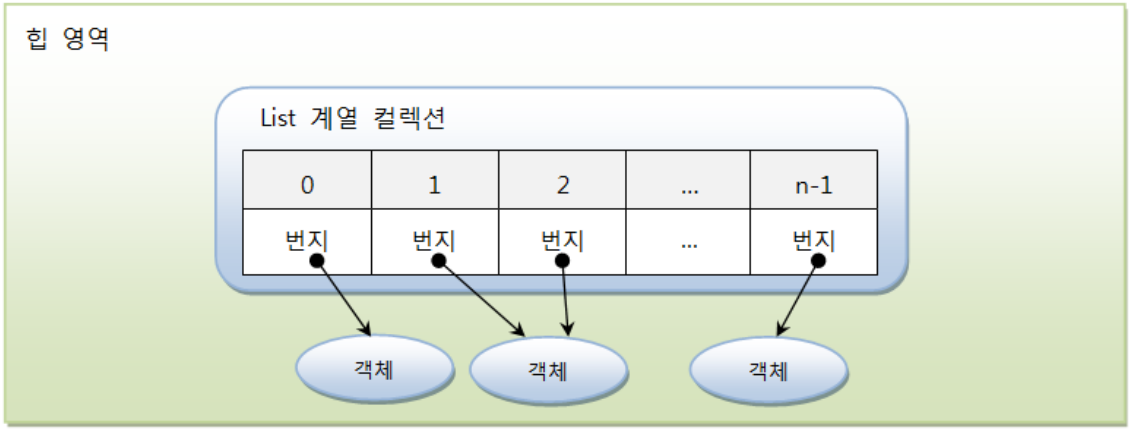
인터페이스 분류		특징	구현 클래스
Collection	List 계열	- 순서를 유지하고 저장 - 중복 저장 가능	ArrayList, Vector, LinkedList
	Set 계열	- 순서를 유지하지 않고 저장 - 중복 저장 안됨	HashSet, TreeSet
Map 계열		- 키와 값의 쌍으로 저장 - 키는 중복 저장 안됨	HashMap, Hashtable, TreeMap, Properties

2절. List 컬렉션

❖ 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

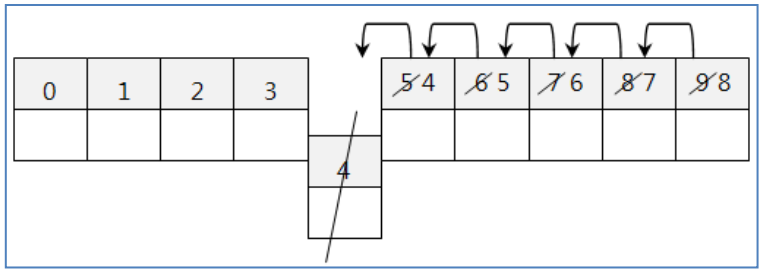
기능	메소드	설명
객체 추가	boolean add(E e)	주어진 객체를 맨끝에 추가
	void add(int index, E element)	주어진 인덱스에 객체를 추가
	set(int index, E element)	주어진 인덱스에 저장된 객체를 주어진 객체로 바꿈
객체 검색	boolean contains(Object o)	주어진 객체가 저장되어 있는지 여부
	E get(int index)	주어진 인덱스에 저장된 객체를 리턴
	isEmpty()	컬렉션이 비어 있는지 조사
	int size()	저장되어있는 전체 객체수를 리턴
객체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

- 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

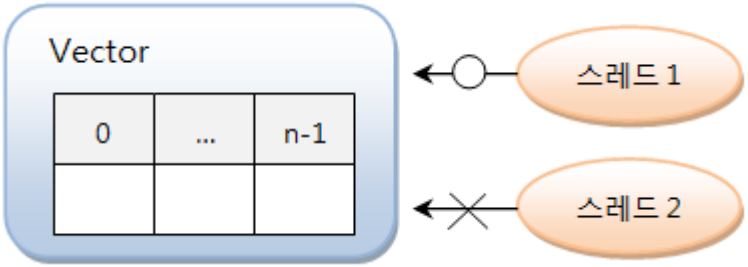


2절. List 컬렉션

❖ Vector

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

- ArrayList와 동일한 내부 구조를 가짐
- 특징
  - Vector는 동기화(synchronization)된 메소드로 구성
    - 복수의 스레드가 동시에 Vector에 접근할 수 없음
    - 멀티 스레드 환경에서 안전하게 객체를 추가, 삭제 가능

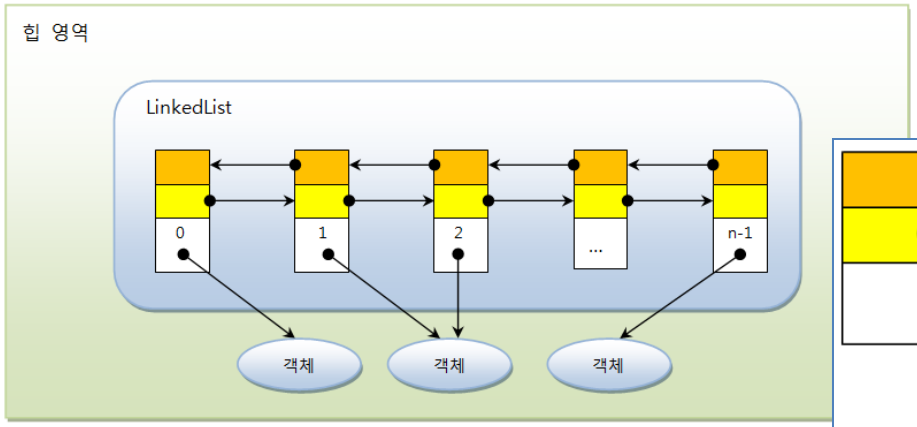


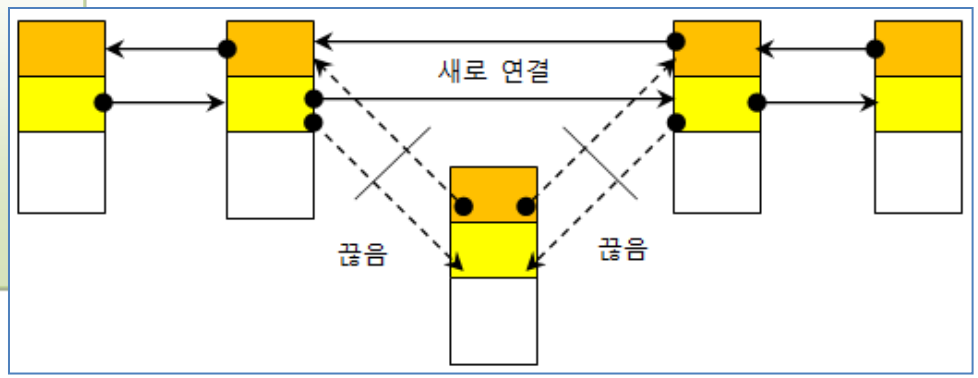
2절. List 컬렉션

❖ 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



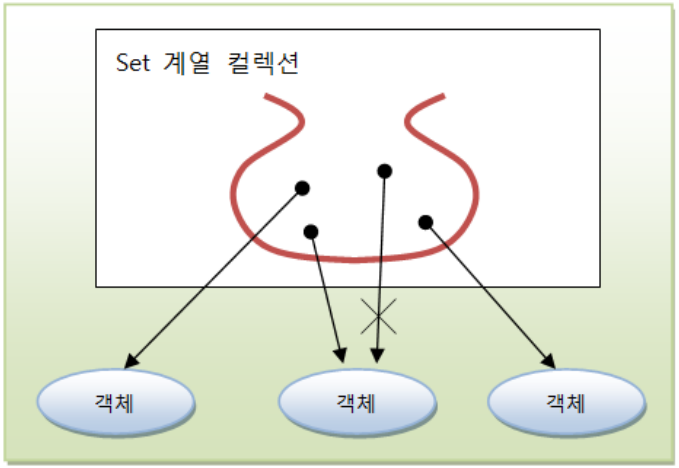


3절. Set 컬렉션

❖ 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

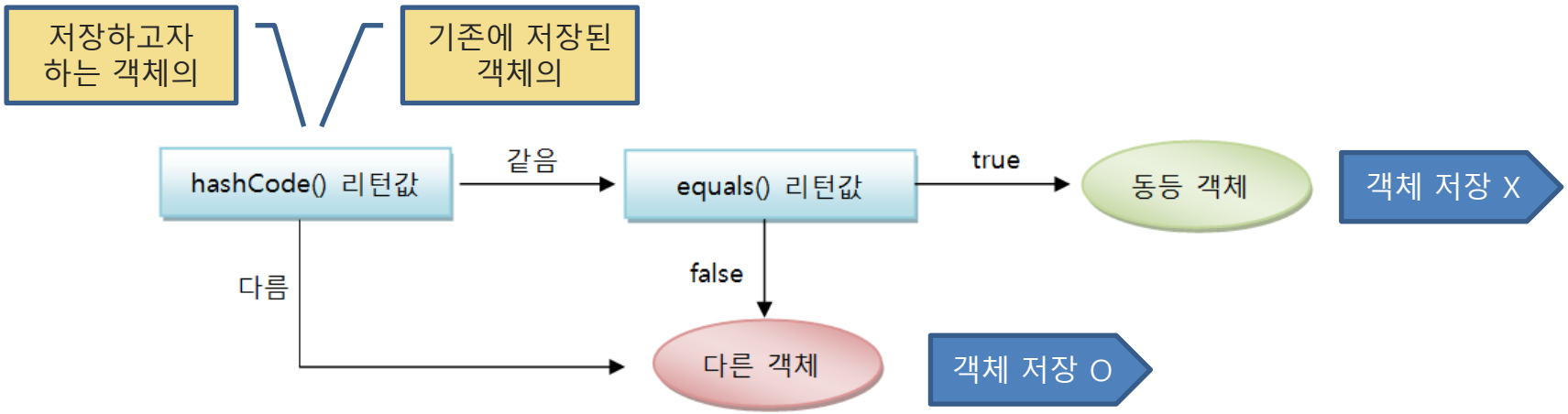
- 인덱스로 객체를 검색해서 가져오는 메소드 없음

3절. Set 컬렉션

❖ HashSet

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

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

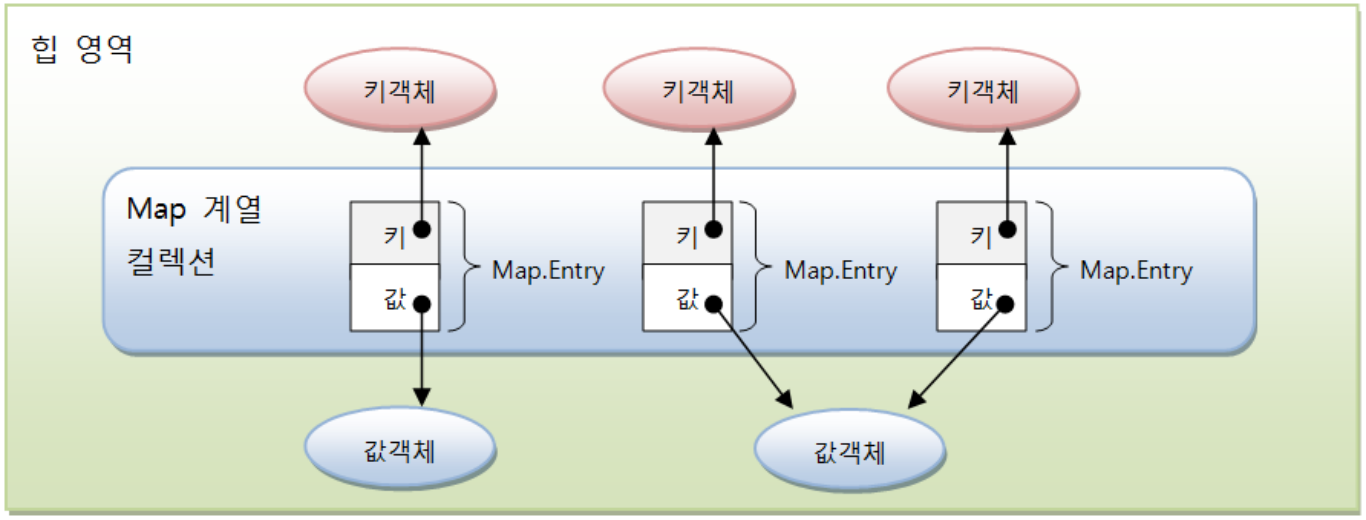


4절. Map 컬렉션

❖ 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

❖ 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

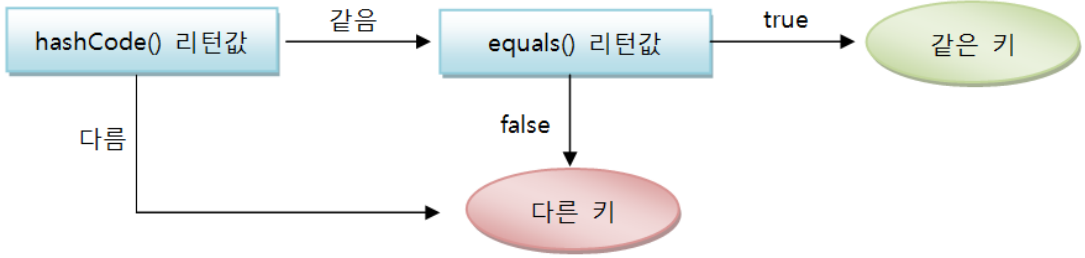
❖ HashMap

▪ 특징

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

키 타입 — 값 타입 — 키 타입 — 값 타입

- 키 객체가 hashCode()의 리턴값이 같고, equals() 메소드가 true를 리턴할 경우, 동일 키로 보고 중복 저장을 하지 않음



- 키 타입은 String 타입을 많이 사용
  - String은 문자열이 같을 경우 동등 객체가 될 수 있도록 hashCode()와 equals() 메소드가 재정의되어 있기 때문

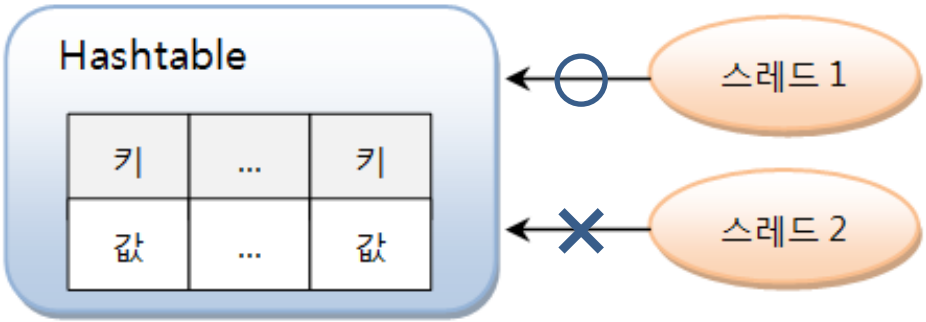
4절. Map 컬렉션

❖ Hashtable

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

키 타입 값 타입 키 타입 값 타입

- 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

실습

❖ 다양한 컬렉션 프레임워크 사용하기

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Vector;

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

        int idx = 0;

        list.add("Java");
        list.add("OOP");
        list.add("객체지향프로그래밍");

        int size = list.size();
        System.out.println("리스트 총 개수 = " + size);

        String item = list.get(2);
        System.out.println("리스트 2번 : " + item);

        System.out.println();

        for(String s : list)
            System.out.println(idx++ + " : " + s);

        System.out.println();

        list.remove(1);
        idx = 0;
        for(String s : list)
            System.out.println(idx++ + " : " + s);

        System.out.println();

        list.remove(1);
        idx = 0;
        for(String s : list)
            System.out.println(idx++ + " : " + s);

        System.out.println("----------------");

        List<String> vector = new Vector<String>();
        vector.add("홍길동");
        vector.add("이순신");
        for (String s:vector)
            System.out.println(s);

        System.out.println("----------------");

        List<String> linkedList = new LinkedList<String>();
        linkedList.add("apple");
        linkedList.add("melon");
        linkedList.add("banana");

        for(String s:linkedList)
            System.out.println(s);

        System.out.println();

        linkedList.remove("apple");
        for(String s:linkedList)
            System.out.println(s);

        System.out.println("----------------");

        Map<String, String> map = new HashMap<String, String>();
        map.put("name", "홍길동");
        map.put("gender", "남자");
        map.put("id", "admin");
        map.put("password", "1234");

        for(String s : map.keySet())
            System.out.println(s + " = " + map.get(s));

    }
}
```

실습

❖ 다양한 컬렉션 프레임워크 사용하기

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Vector;

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

        int idx = 0;

        list.add("Java");
        list.add("OOP");
        list.add("객체지향프로그래밍");

        int size = list.size();
        System.out.println("리스트 총 개수 = " + si

        String item = list.get(2);
        System.out.println("리스트 2번 : " + item

        System.out.println();

        for(String s : list)
            System.out.println(idx++ + " : " + s);

        System.out.println();

        list.remove(1);
        idx = 0;
        for(String s : list)
            System.out.println(idx++ + " : " + s);

        System.out.println();
        list.remove(1);
        idx = 0;
        for(String s : list)
            System.out.println(idx++ + " : " + s);

        System.out.println("----------------");

        List<String> vector = new Vector<String>();
        vector.add("홍길동");
        vector.add("이순신");
        for (String s:vector)
            System.out.println(s);

        System.out.println("----------------");

        List<String> linkedList = new LinkedList<String>();
        linkedList.add("apple");
        linkedList.add("melon");
```

```
Set<String> keyset = map.keySet();
Iterator<String> keyIter = keyset.iterator();

while(keyIter.hasNext()) {
    String k = keyIter.next();
    System.out.println(k + " : " + k.hashCode());
}
```

```
        Map<String, String> map = new HashMap<String, String>();
        map.put("name", "홍길동");
        map.put("gender", "남자");
        map.put("id", "admin");
        map.put("password", "1234");

        for(String s : map.keySet())
            System.out.println(s + " = " + map.get(s));

    }
}
```