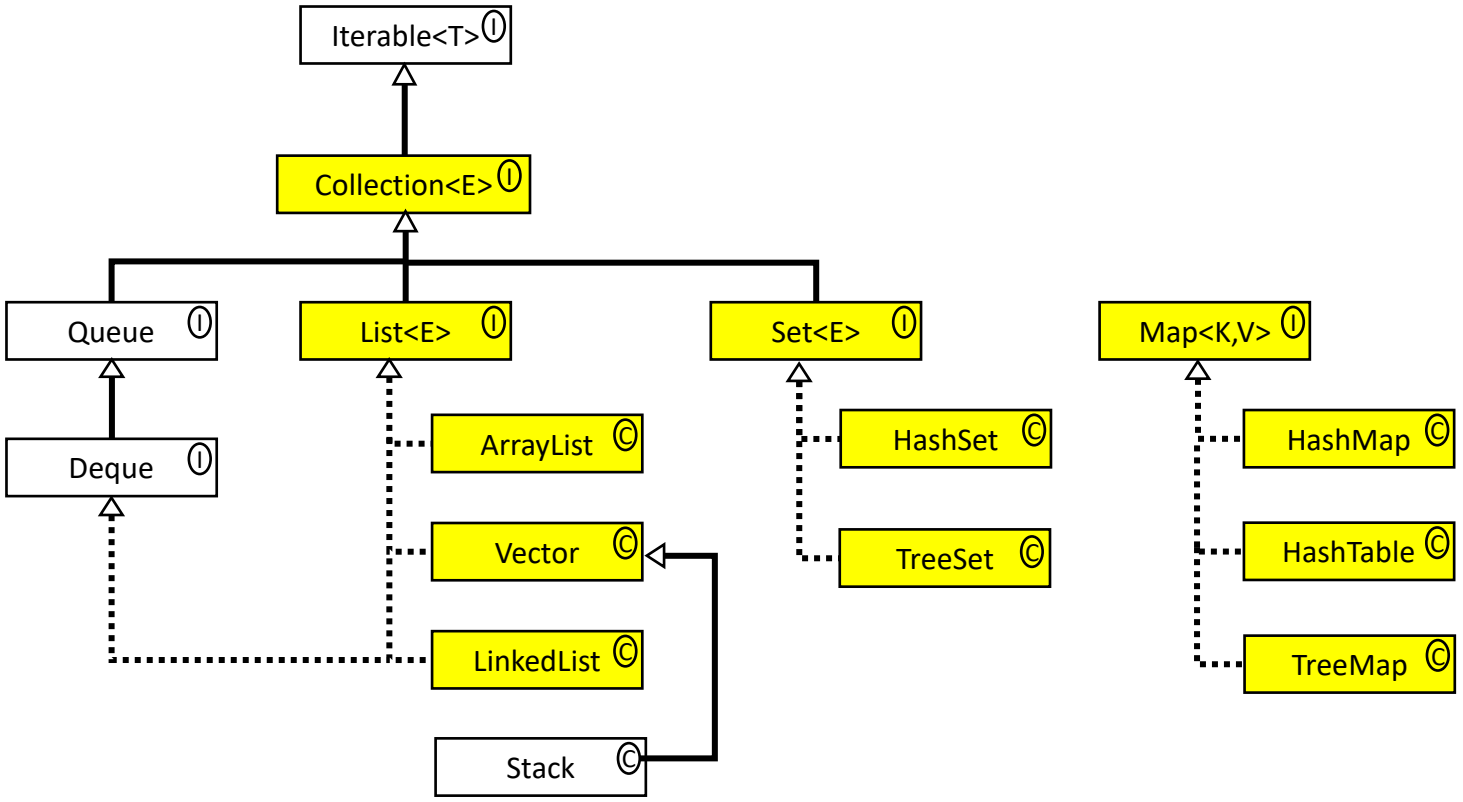
Ch08_Collection 프레임 워크

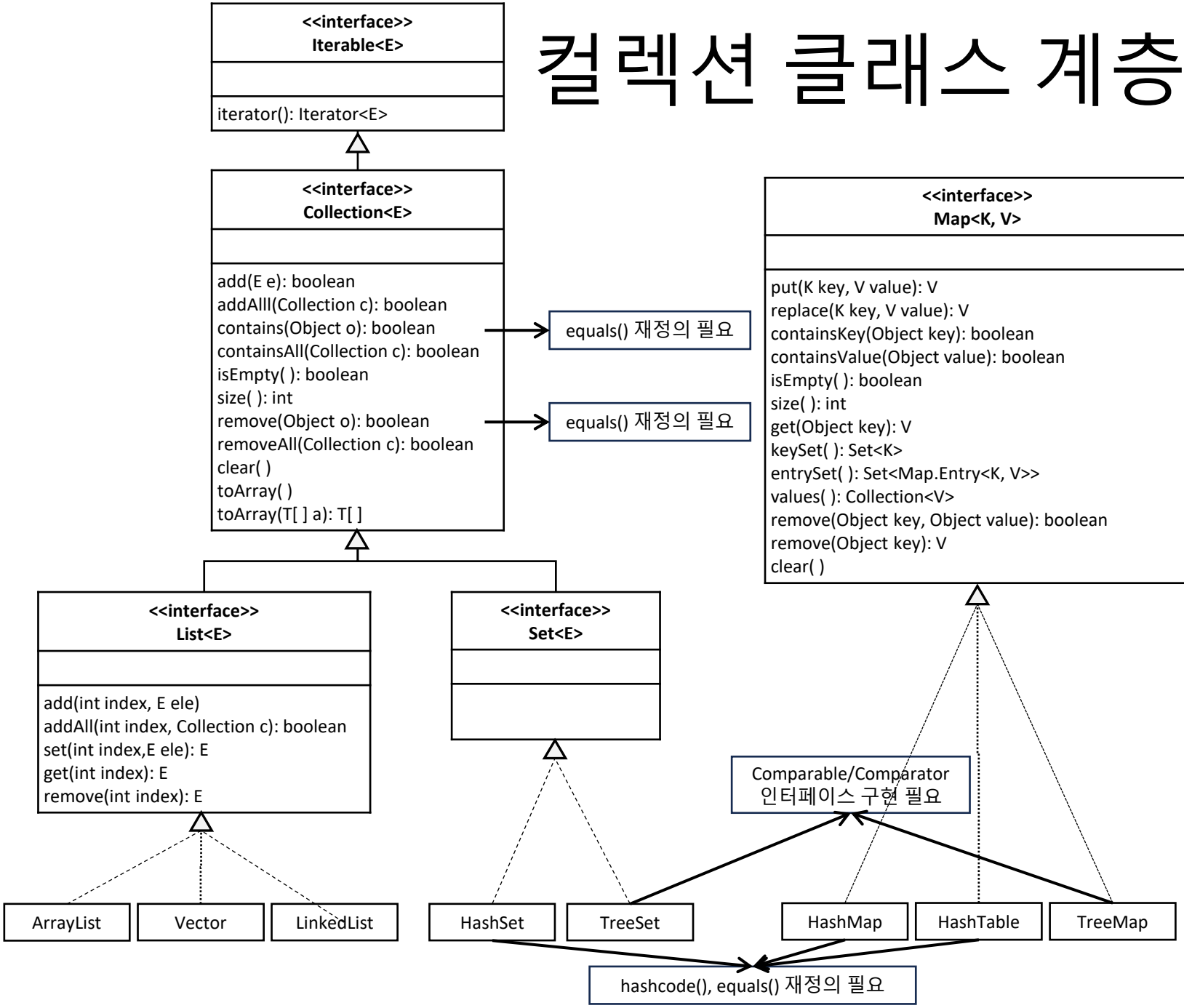
Collection Framework



Collection 특징

<table>
<tr><th colspan="4">Collection</th></tr>
<tr><td colspan="4">- 동일 타입의 원소들을 묶어 동적으로 관리하는 자료구조
- 컬렉션에서 원소는 객체만 가능</td></tr>
<tr><td rowspan="3">List</td><td rowspan="3">- 순서유지(연속적/순차적 저장)
- 중복저장 가능
= list(파이썬)</td><td>ArrayList</td><td>- 배열을 이용하여 List의 특징을 구현한 자료구조
- 배열과 같이 인덱스로 원소 관리
- 비동기화 메소드로 구성, 싱글 쓰레드 환경에 효율적
- 사용자 정의 객체에 대한 contains(), remove() 위해 equals() 재정의 필요</td></tr>
<tr><td>Vector</td><td>- ArrayList와 유사(구버전)
- 동기화 메소드들로 구성, 멀티 쓰레드 환경에 적합</td></tr>
<tr><td>LinkedList</td><td>- 비동기화 메소드로 구성, 싱글 쓰레드 환경에 효율적
- 저장방식에서 ArrayList와 Vector는 연속적인 객체 저장
- 저장방식에서 LinkedList는 불연속적으로 노드를 서로 연결한 자료구조
- 빈번한 객체 추가/삭제 수행시 효율적, 그러나 검색은 비효율적
- 사용자 정의 객체에 대한 contains(), remove() 위해 equals() 재정의 필요</td></tr>
<tr><td rowspan="2">Set</td><td rowspan="2">- 순서없음(인덱스 정보 없음)
- 중복저장 불가
- set(파이썬)</td><td>HashSet</td><td>- HashMap을 이용하여 Set의 특징을 구현한 자료구조
- 입력 순서와 관계 없이 출력
- 사용자 정의 객체 동등 비교 위해 hashcode() 및 equals() 재정의 필요</td></tr>
<tr><td>TreeSet</td><td>- Set의 기본 기능에 정렬 및 검색 기능 추가 -> Set이 아닌 TreeSet으로 선언 필요
- 균형 이진 검색 트리(binary search tree) 형태로 여러 노드가 서로 연결된 컬렉션
- 입력 순서와 관계 없이 크기 순으로 출력
- 사용자 정의 객체 크기 비교 위해 Comparable/Comparator 인터페이스 구현 필요</td></tr>
<tr><td rowspan="3">Map</td><td rowspan="3">- 키와 데이터로 구성된 엔트리 객체 저장
- 순서없음
- 키는 중복저장 불가
- 데이터는 중복저장 가능
- 데이터 검색에 효율적
= dictionary(파이썬)</td><td>HashMap</td><td>- 배열과 해쉬 함수 이용하여 Map의 특징을 구현한 자료구조
- 비동기화 메소드로 구성, 싱글 쓰레드 환경에 효율적
- 사용자 정의 객체 동등 비교 위해 hashcode() 및 equals() 재정의 필요</td></tr>
<tr><td>HashTable</td><td>- HashMap과 유사(구버전)
- 동기화 메소드들로 구성, 멀티 쓰레드 환경에 적합</td></tr>
<tr><td>TreeMap</td><td>- 균형 이진 탐색 트리를 이용하여 Map의 특징을 구현한 자료구조
- Map의 기본 기능에 정렬 및 검색 기능 추가
- 입력 순서와 관계 없이 크기 순으로 출력
- 사용자 정의 객체 크기 비교 위해 Comparable/Comparator 인터페이스 구현 필요</td></tr>
</table>

컬렉션 클래스 계층구조

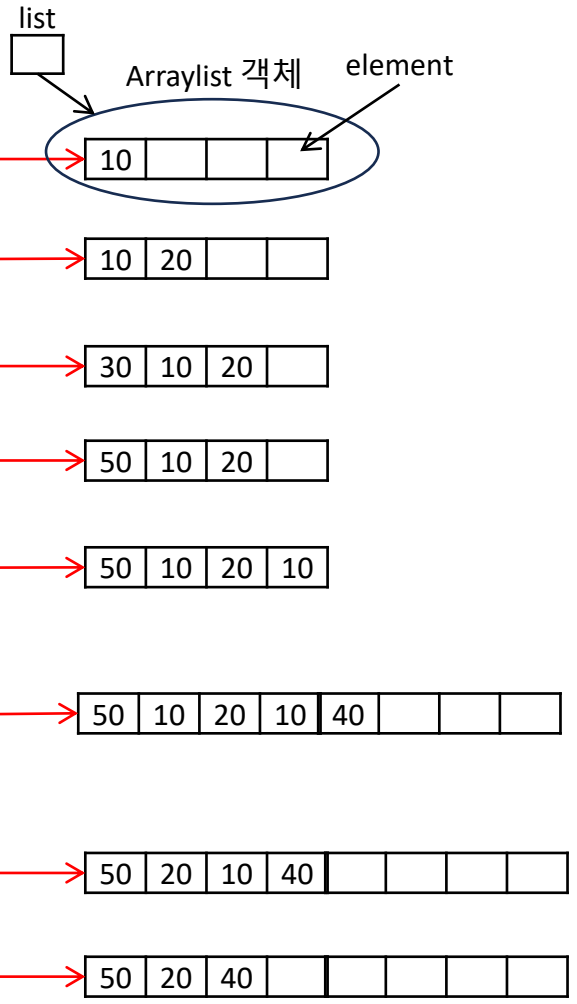

<<interface>>
Iterable<E>
iterator(): Iterator<E>
<<interface>>
Collection<E>
add(E e): boolean
addAlll(Collection c): boolean
contains(Object o): boolean
containsAll(Collection c): boolean
isEmpty( ): boolean
size( ): int
remove(Object o): boolean
removeAll(Collection c): boolean
clear( )
toArray( )
toArray(T[ ] a): T[ ]
equals() 재정의 필요
equals() 재정의 필요
<<interface>>
Map<K, V>
put(K key, V value): V
replace(K key, V value): V
containsKey(Object key): boolean
containsValue(Object value): boolean
isEmpty( ): boolean
size( ): int
get(Object key): V
keySet( ): Set<K>
entrySet( ): Set<Map.Entry<K, V>>
values( ): Collection<V>
remove(Object key, Object value): boolean
remove(Object key): V
clear( )
<<interface>>
List<E>
add(int index, E ele)
addAll(int index, Collection c): boolean
set(int index,E ele): E
get(int index): E
remove(int index): E
<<interface>>
Set<E>
Comparable/Comparator
인터페이스 구현 필요
ArrayList
Vector
LinkedList
HashSet
TreeSet
HashMap
HashTable
TreeMap
hashcode(), equals() 재정의 필요

컬렉션 주요 메소드

기능(행위)	Collection메소드	List 메소드	Set 메소드	Map 메소드
객체추가	add(E e): boolean	add(int index, E element)		put(K key, V value): V
	addAll(Collection c): boolean	addAll(int index, Collection c): boolean		
객체 수정		set(int index, E element): E		replace(K key, V value): V
객체검색	contains(Object o): boolean			containsKey(Object key): boolean
	containsAll(Collection c): boolean			containsValue(Object value): boolean
	isEmpty( ): boolean			isEmpty( ): boolean
	size( ): int			size( ): int
		get(int index): E		get(Object key): V
				keyset( ): Set<K>
				entrySet( ): Set<Map.Entry<K,V>>
				values( ): Collection<V>
객체제거	remove(Object o): boolean	remove(int index): E		remove(Object key, Object value): boolean
	removeAll(Collection c): boolean			remove(Object key): V
	clear( )			clear( )
기타	iterator( ): Iterator<E> toArray( ) toArray(T[ ] a): T[ ]			

ArrayList(Vector) - 메모리 상태

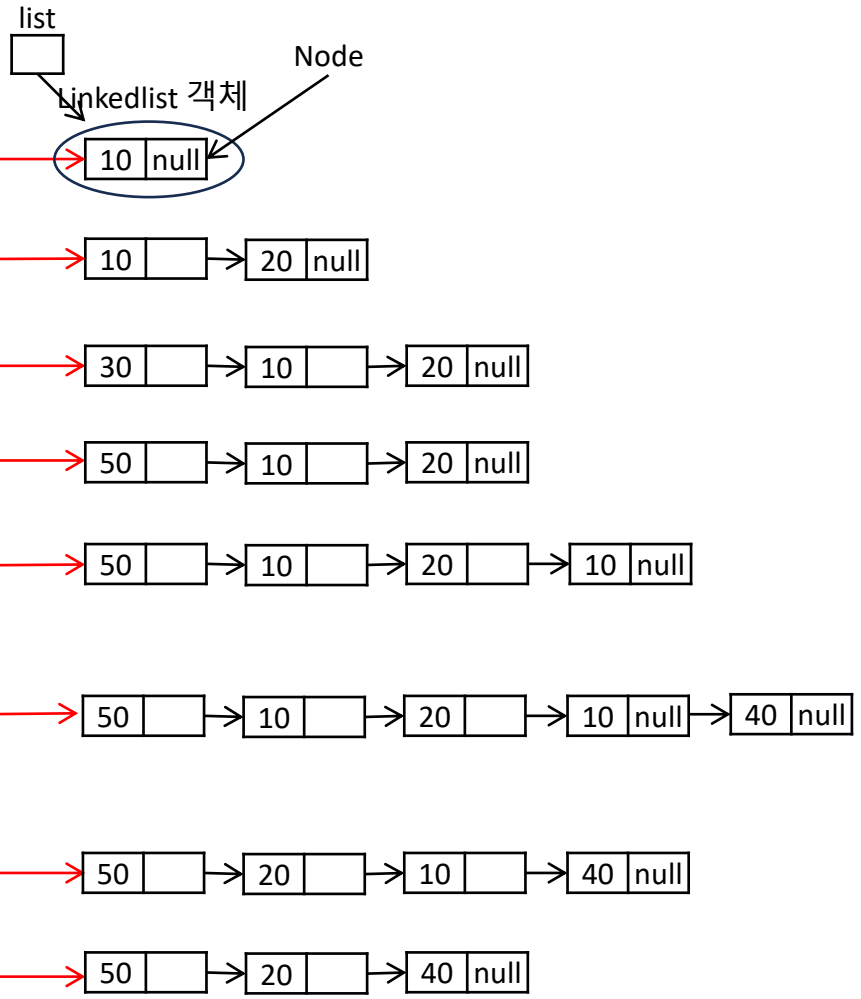
```
List<Integer> list = new ArrayList<>();
list.add(10);
System.out.println(list);
list.add(20);
System.out.println(list);
list.add(0, 30);
System.out.println(list);
list.set(0, 50);
System.out.println(list);
list.add(10);
System.out.println(list);
System.out.println(list.get(2));
list.add(40);
System.out.println(list);
System.out.println(list.contains(50));
list.remove(new Integer(10));
System.out.println(list);
list.remove(2);
System.out.println(list);
```




```
[10]
[10, 20]
[30, 10, 20]
[50, 10, 20]
[50, 10, 20, 10]
20
[50, 10, 20, 10, 40]
true
[50, 20, 10, 40]
[50, 20, 40]
```

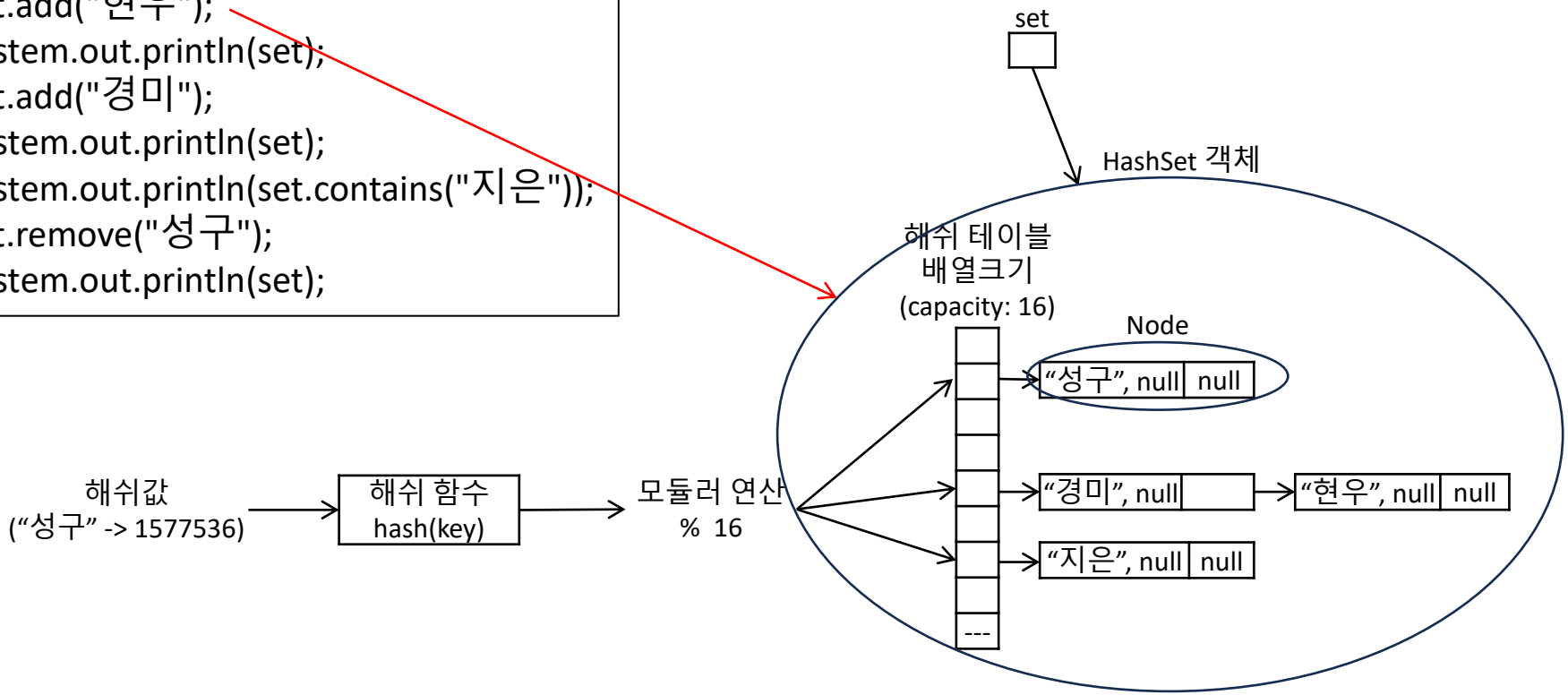
LinkedList - 메모리 상태

```
List<Integer> list = new LinkedList<>();
list.add(10);
System.out.println(list);
list.add(20);
System.out.println(list);
list.add(0, 30);
System.out.println(list);
list.set(0, 50);
System.out.println(list);
list.add(10);
System.out.println(list);
System.out.println(list.get(2));
list.add(40);
System.out.println(list);
System.out.println(list.contains(50));
list.remove(new Integer(10));
System.out.println(list);
list.remove(2);
System.out.println(list);
```



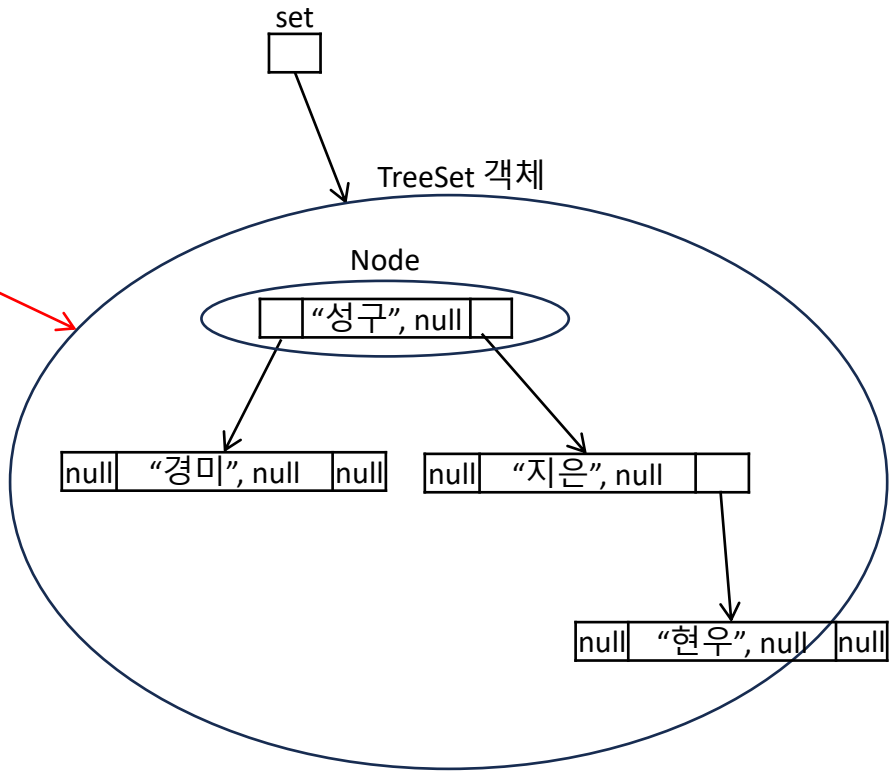
HashSet - 메모리 상태

```
Set<String> set = new HashSet<>();
set.add("성구");
System.out.println(set);
set.add("경미");
set.add("지은");
set.add("현우");
System.out.println(set);
set.add("경미");
System.out.println(set);
System.out.println(set.contains("지은"));
set.remove("성구");
System.out.println(set);
```



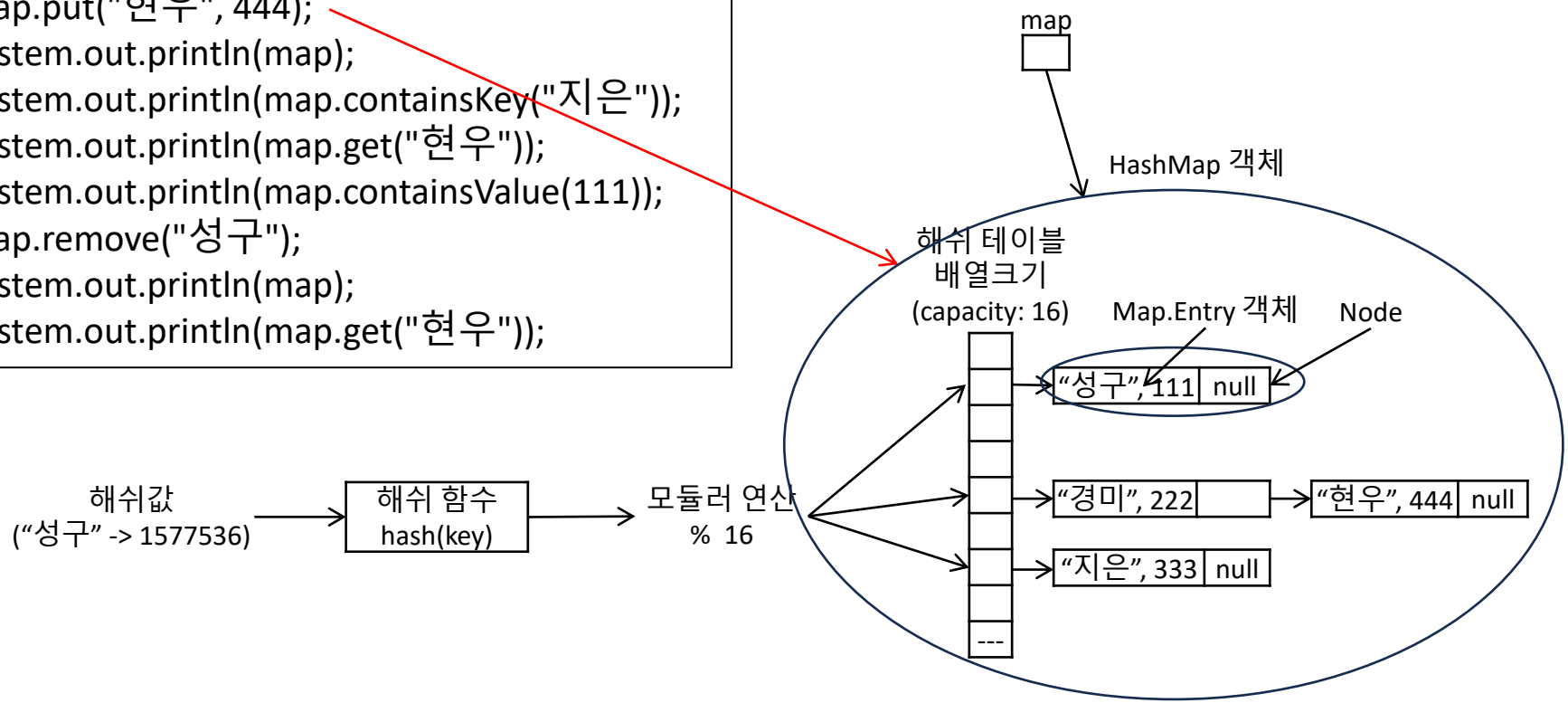
TreeSet - 메모리 상태

```
Set<String> set = new TreeSet<>();
set.add("성구");
System.out.println(set);
set.add("경미");
set.add("지은");
set.add("현우");
System.out.println(set);
set.add("경미");
System.out.println(set);
System.out.println(set.contains("지은"));
set.remove("성구");
System.out.println(set);
```



HashMap(HashTable) - 메모리 상태

```
Map<String, Integer> map = new HashMap<>();
map.put("성구", 111);
System.out.println(map);
map.put("경미", 222);
map.put("지은", 333);
map.put("현우", 444);
System.out.println(map);
System.out.println(map.containsKey("지은"));
System.out.println(map.get("현우"));
System.out.println(map.containsValue(111));
map.remove("성구");
System.out.println(map);
System.out.println(map.get("현우"));
```



TreeMap - 메모리 상태

```
Map<String, Integer> map = new TreeMap<>();
map.put("성구", 111);
System.out.println(map);
map.put("경미", 222);
map.put("지은", 333);
map.put("현우", 444);
System.out.println(map);
System.out.println(map.containsKey("지은"));
System.out.println(map.get("현우"));
System.out.println(map.containsValue(111));
map.remove("성구");
System.out.println(map);
System.out.println(map.get("현우"));
```