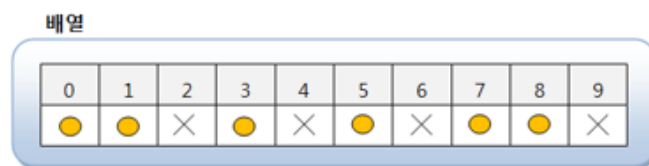


1. 컬렉션 프레임워크(Collection Framework)

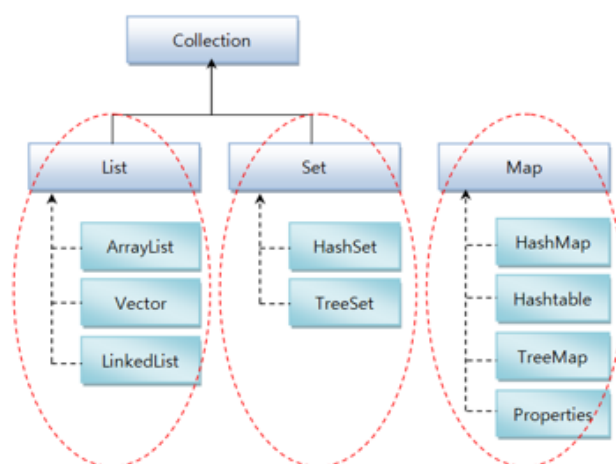
01. 컬렉션 프레임워크이란?

- 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 함



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

02. 컬렉션 프레임워크의 주요 인터페이스

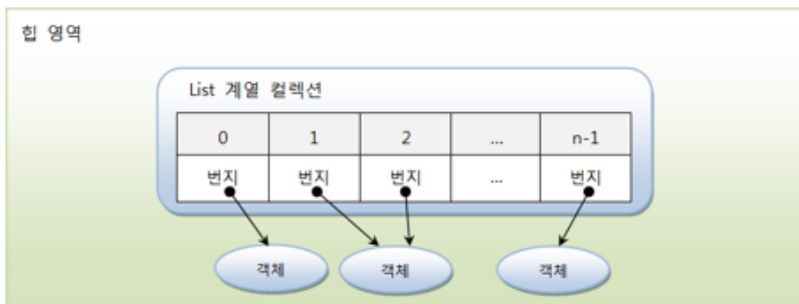


인터페이스 분류		특징	구현 클래스
Collection	List 계열	- 순서를 유지하고 저장 - 중복 저장 가능	ArrayList, Vector, LinkedList
	Set 계열	- 순서를 유지하지 않고 저장 - 중복 저장 안됨	HashSet, TreeSet
Map 계열		- 키와 값의 쌍으로 저장 - 키는 중복 저장 안됨	HashMap, Hashtable, TreeMap, Properties

2. List 컬렉션

01. List 컬렉션의 특징 및 주요 메소드

- 특징
 - 인덱스로 관리
 - 중복해서 객체 저장 가능
- 구현 클래스
 - ArrayList
 - Vector
 - LinkedList



02. List 컬렉션의 특징 및 주요 메소드

- 주요 메소드

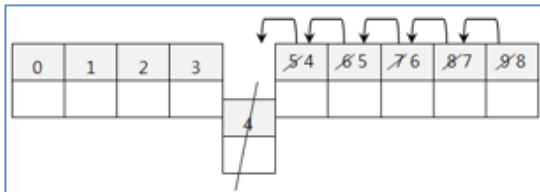
기능	메소드	설명
객체 추가	<code>boolean add(E e)</code>	주어진 객체를 맨끝에 추가
	<code>void add(int index, E element)</code>	주어진 인덱스에 객체를 추가
	<code>set(int index, E element)</code>	주어진 인덱스에 저장된 객체를 주어진 객체로 바꿈
객체 검색	<code>boolean contains(Object o)</code>	주어진 객체가 저장되어 있는지 여부
	<code>E get(int index)</code>	주어진 인덱스에 저장된 객체를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

03. ArrayList

- 저장 용량(capacity)
 - 초기 용량 : 10 (따로 지정 가능)
 - 저장 용량을 초과한 객체들이 들어오면 자동적으로 늘어남, 고정도 가능



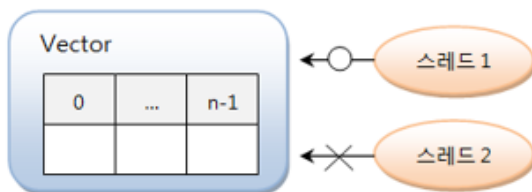
- 객체 제거
 - 바로 뒤 인덱스부터 마지막 인덱스까지 모두 앞으로 1씩 당겨짐



04. Vector

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

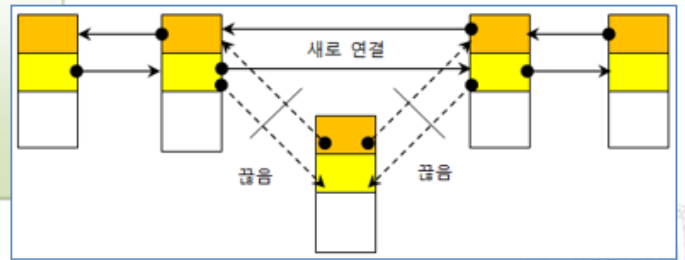
- 특징
 - Vector는 스레드 동기화(synchronization)
 - 복수의 스레드가 동시에 Vector에 접근해 객체를 추가, 삭제하더라도 스레드에 안전(thread safe)



05. LinkedList

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

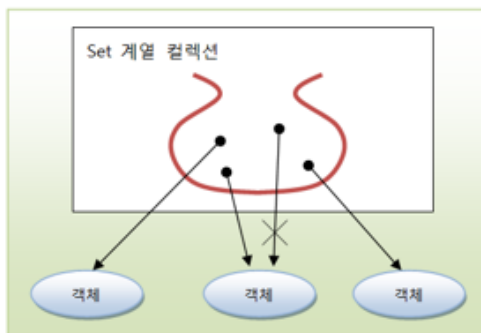
- 특징
 - 인접 참조를 링크해서 체인처럼 관리
 - 특정 인덱스에서 객체를 제거하거나 추가하게 되면 바로 앞뒤 링크만 변경
 - 빈번한 객체 삭제와 삽입이 일어나는 곳에서는 ArrayList보다 좋은 성능



3. Set 컬렉션

01. 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



- 주요 메소드

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

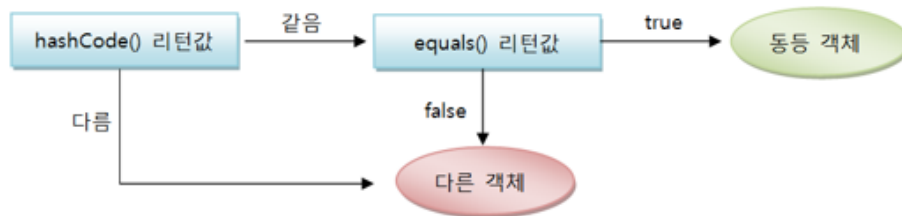
- 전체 객체 대상으로 한 번씩 반복해 가져오는 반복자(iterator) 제공

- 인덱스로 객체를 검색해서 가져오는 메소드 없음

02. HashSet

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

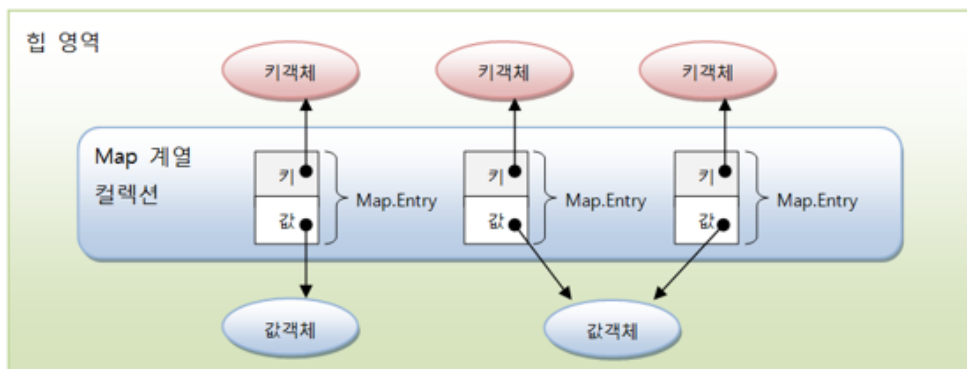
- 특징
 - 동일 객체 및 동등 객체는 중복 저장하지 않음
 - 동등 객체 판단 방법



4. Map 컬렉션

01. 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

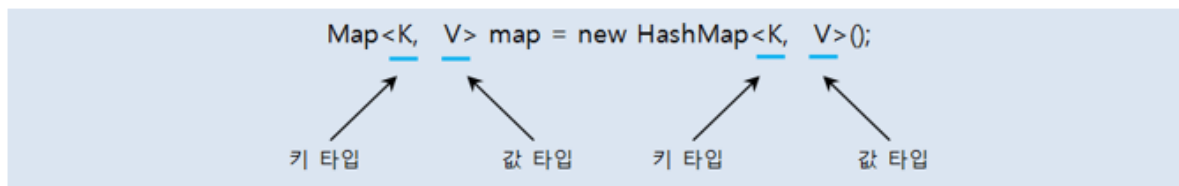


- 주요 메소드

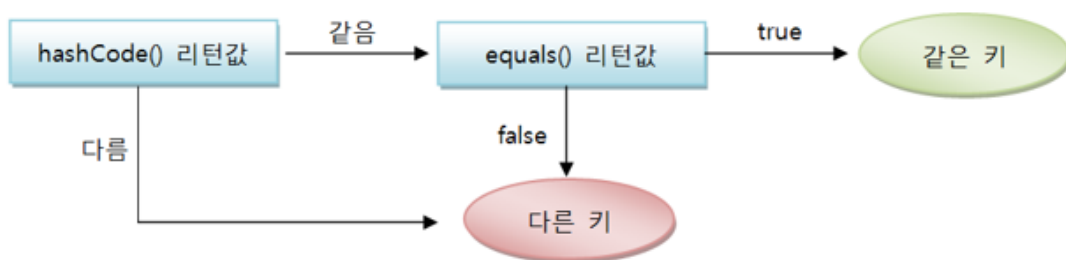
기능	메소드	설명
객체 추가	<code>V put(K key, V value)</code>	주어진 키와 값을 추가, 저장되면 값을 리턴
	<code>boolean containsKey(Object key)</code>	주어진 키가 있는지 여부
객체 검색	<code>boolean containsValue(Object value)</code>	주어진 값이 있는지 여부
	<code>Set<Map.Entry<K,V>> entrySet()</code>	키와 값의 쌍으로 구성된 모든 <code>Map.Entry</code> 객체를 <code>Set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

02. HashMap

- 특징

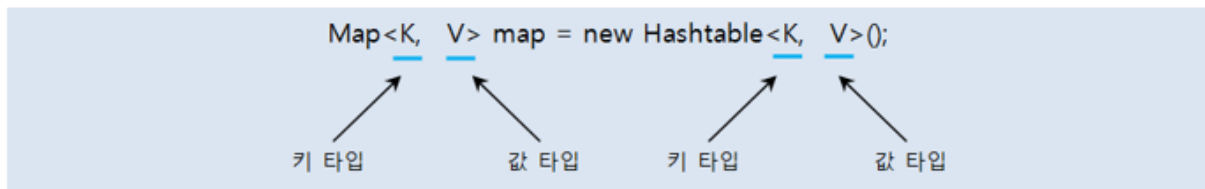


- 키 객체는 `hashCode()`와 `equals()`를 재정의해 동등 객체가 될 조건을 정해야 한다

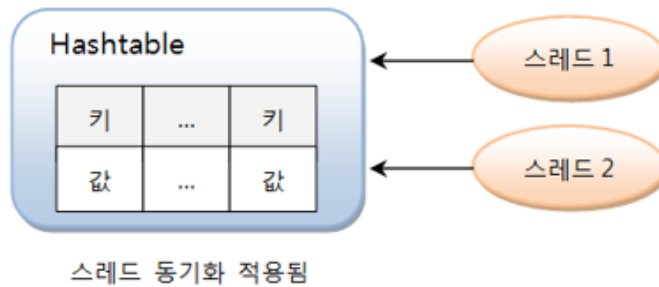


- 키 타입은 `String` 많이 사용
 - `String`은 문자열이 같은 경우 동등 객체가 될 수 있도록 `hashCode()`와 `equals()` 메소드가 재정의 되어 있기 때문

03. hashtable



- 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)



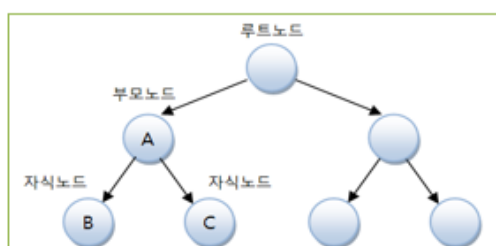
04. Properties

- 특징
 - 키와 값을 String 타입으로 제한한 Map 컬렉션
 - Properties는 프로퍼티(~.properties)파일을 읽어 들일 때 주로 사용
- 프로퍼티(~.properties) 파일
 - 옵션 정보, 데이터베이스 연결 정보, 국제화(다국어)정보를 기록
 - 텍스트 파일로 활용
 - 애플리케이션에서 주로 변경이 잦은 문자열을 저장
 - 유지 보수를 편리하게 만들어 줌
 - 키와 값이 = 기호로 연결되어 있는 텍스트 파일
 - ISO 8859-1 문자셋으로 저장
 - 한글은 유니코드(Unicode)로 변환되어 저장

5. 검색 기능을 강화시킨 컬렉션

01. 검색 기능을 강화시킨 컬렉션(계층 구조 활용)

- TreeSet, TreeMap -> 이진트리(binary tree) 사용하기 때문에 검색 속도 향상

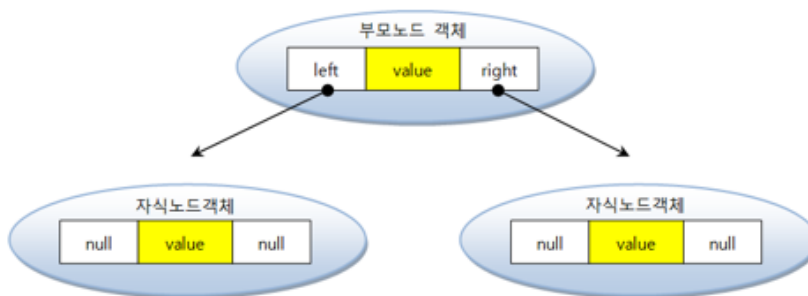


02. 이진 트리 구조

- 부모 노드와 자식 노드로 구성
 - 왼쪽 자식 노드 : 부모 보다 작은 값
 - 오른쪽 자식 노드 : 부모 보다 큰 값
- 정렬 쉬움
 - 올림 차순 : [왼쪽노드->부모노드->오른쪽노드]
 - 내림 차순 : [오른쪽노드->부모노드->왼쪽노드]

03. TreeSet

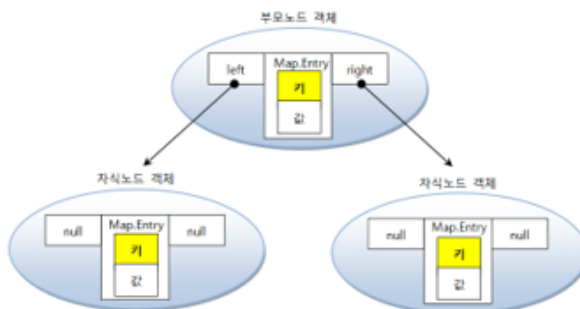
- 특징
 - 이진 트리(binary tree)를 기반으로 한 Set 컬렉션
 - 왼쪽과 오른쪽 자식 노드를 참조하기 위한 두 개의 변수로 구성



- 주요 메소드
 - 특정 객체를 찾는 메소드 : first(), last(), lower(), higher(), ...
 - 정렬 메소드 : descendingIterator(), descendingSet()
 - 범위 검색 메소드 : headSet(), tailSet(), subSet()

04. TreeMap

- 특징
 - 이진 트리(binary tree)를 기반으로 한 Map 컬렉션
 - 키와 값이 저장된 Map.Entry를 저장
 - 왼쪽과 오른쪽 자식 노드를 참조하기 위한 두 개의 변수로 구성



- 주요 메소드
 - 단일 노드 객체를 찾는 메소드 : firstEntry(), lastEntry(), lowerEntry(), higherEntry(), ...
 - 정렬 메소드 : descendingKeySet(), descendingMap()
 - 범위 검색 메소드 : headMap(), tailMap(), subMap()

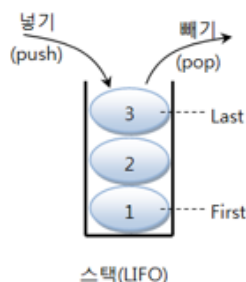
05. Comparable과 Comparator

- TreeSet과 TreeMap의 자동 정렬
 - TreeSet의 객체와 TreeMap의 키는 저장과 동시에 자동 오름차순 정렬
 - 숫자(Integer, Double)타입일 경우에는 값으로 정렬
 - 문자열(String) 타입일 경우에는 유니코드로 정렬
 - TreeSet과 TreeMap은 정렬 위해 java.lang.Comparable을 구현 객체를 요구
 - Integer, Double, String은 모두 Comparable 인터페이스 구현
 - Comparable을 구현하고 있지 않을 경우에는 저장하는 순간 ClassCastException 발생

6. LIFO와 FIFO 컬렉션

01. Stack 클래스

```
Stack<E> stack = new Stack<E>();
```

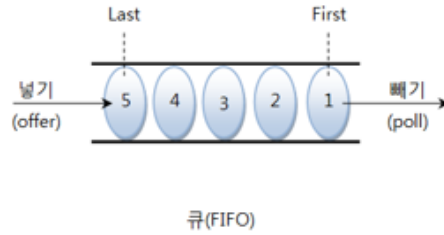


- 특징
 - 후입선출(LIFO : Last In First Out) 구조
 - 응용 예 : JVM 스택 메모리
- 주요 메소드

리턴타입	메소드	설명
E	push(E item)	주어진 객체를 스택에 넣는다.
E	peek()	스택의 맨위 객체를 가져온다. 객체를 스택에서 제거하지는 않는다.
E	pop()	스택의 맨위 객체를 가져온다. 객체를 스택에서 제거한다.

02. Queue 인터페이스

```
Queue queue = new LinkedList();
```



- 특징
 - 선입선출(FIFO:First In First Out)
 - 응용 예 : 작업 큐, 메시지 큐, ...
 - 구현 클래스 : LinkedList
- 주요 메소드

리턴타입	메소드	설명
boolean	offer(E e)	주어진 객체를 넣는다.
E	peek()	객체 하나를 가져온다. 객체를 큐에서 제거하지 않는다.
E	poll()	객체 하나를 가져온다. 객체를 큐에서 제거한다.

7. 동기화된(Synchronized) 컬렉션

01. 비 동기화된 컬렉션을 동기화된 컬렉션으로 래핑

- Collections의 synchronizedXXX() 메소드 제공
 - 772페이지의 예제 그림으로 쉽게 이해하도록!!!

리턴 타입	메소드(매개 변수)	설명
List<T>	synchronizedList(List<T> list)	List를 동기화된 List로 리턴
Map<K,V>	synchronizedMap(Map<K,V> m)	Map을 동기화된 Map으로 리턴
Set<T>	synchronizedSet(Set<T> s)	Set을 동기화된 Set으로 리턴

8. 병렬 처리를 위한 컬렉션

01. 동기화(Synchronized) 컬렉션의 단점

- 하나의 스레드가 요소 처리할 때 전체 잠금 발생
 - 다른 스레드는 대기 상태
 - 멀티 스레드가 병렬적으로 컬렉션의 요소들을 빠르게 처리할 수 없음

02. 컬렉션 요소를 병렬처리하기 위해 제공되는 컬렉션

- ConcurrentHashMap
 - 부분(segment) 잠금 사용
 - 처리하는 요소가 포함도니 부분만 잠금
 - 나머지 부분은 다른 스레드가 변경 가능하게 -> 부분 잠금
- ConcurrentLinkedQueue
 - 락-프리(lock-free) 알고리즘을 구현한 컬렉션
 - 잠금 사용하지 않음
 - 여러 개의 스레드가 동시에 접근하더라도 최소한 하나의 스레드가 성공하도록(안전하게 요소를 저장하거나 얻도록) 처리