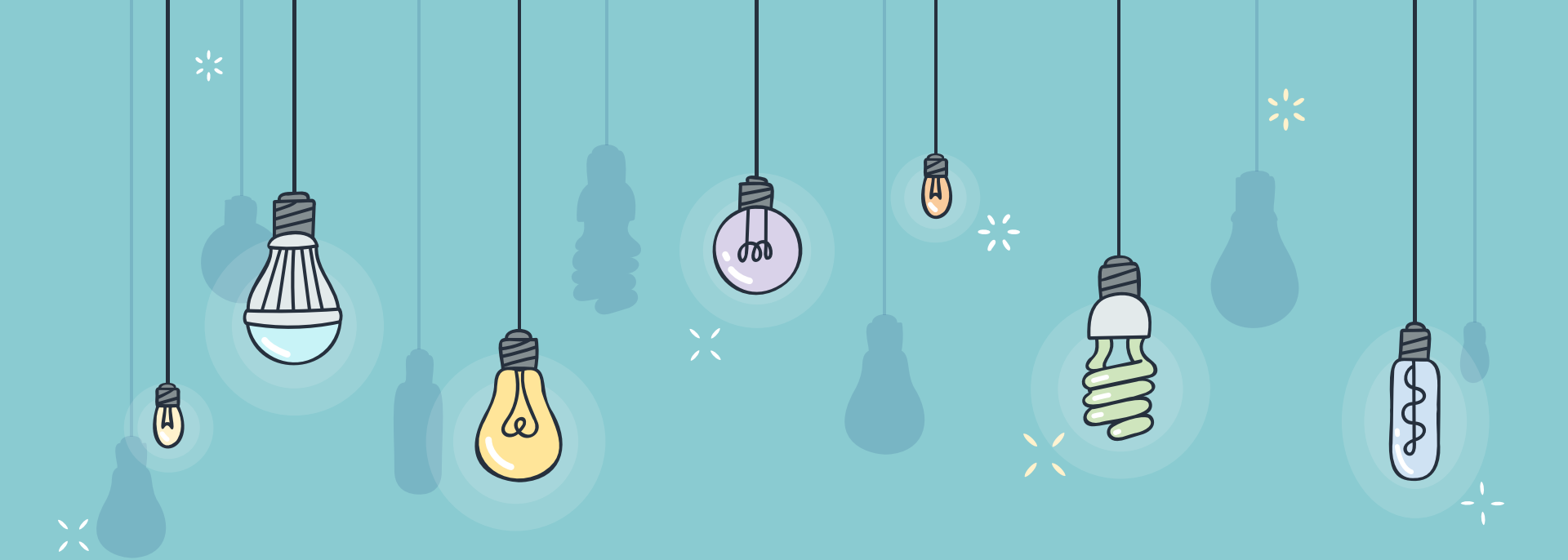
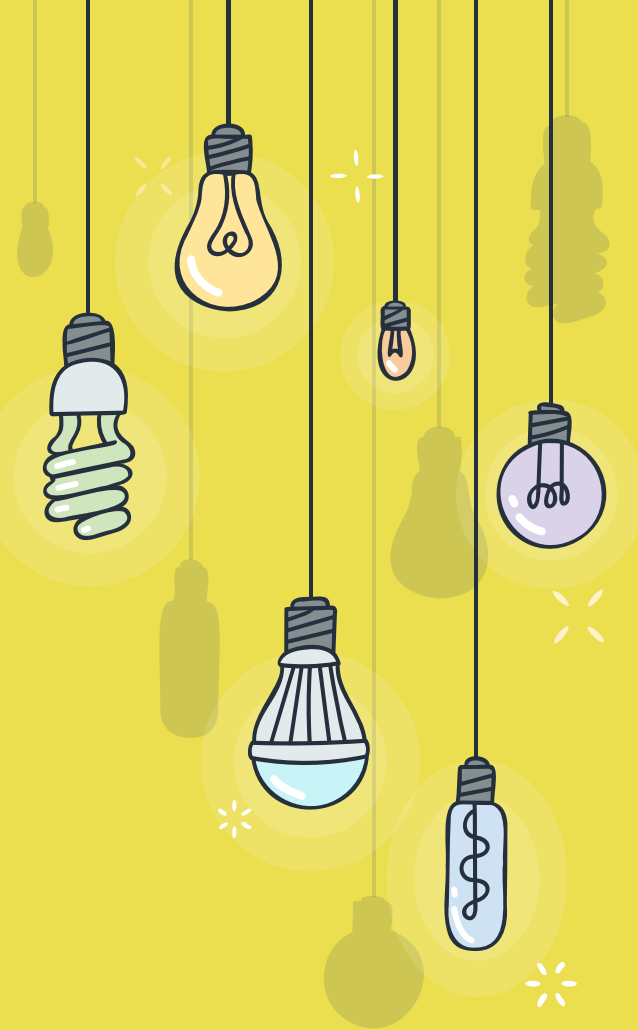


JAVA

컬렉션 프레임워크



1. List 컬렉션

2. Set 컬렉션

3. Map 컬렉션

4. 검색 기능 컬렉션

컬렉션 프레임워크

- 컬렉션
  - 여러 객체들을 모아 놓은 것
- 프레임웍
  - 표준화, 정형화된 체계적인 프로그래밍 방식
- 컬렉션 프레임워크(Collection Framework)
  - 객체들을 효율적으로 추가, 삭제, 검색할 수 있도록 제공되는 컬렉션 라이브러리
  - java.util 패키지에 포함
  - 인터페이스를 통해 정형화된 방법으로 다양한 컬렉션 클래스 이용

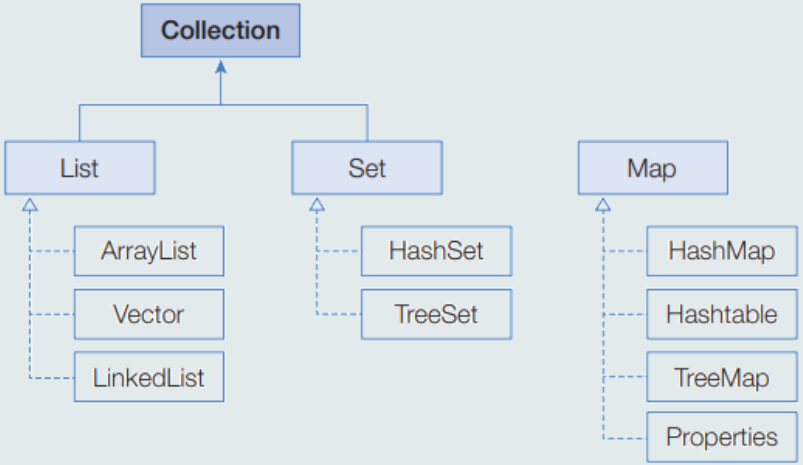
컬렉션 프레임워크

- 배열의 문제점
  - 저장할 수 있는 객체 수가 배열을 생성할 때 결정
  - 객체 삭제했을 때 해당 인덱스가 비게 됨
    - 객체를 저장하려면 어디가 비어있는지 확인해야 함

0	1	2	3	4	5	6	7	8	9
●	X	●	X	●	●	●	X	X	●

[배열]

컬렉션 프레임워크



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

List 컬렉션

- List인터페이스 메서드
  - 인덱스로 관리
    - 객체를 저장하면 인덱스가 부여되고 인덱스로 객체를 검색, 삭제할 수 있는 기능을 제공
  - 중복해서 객체 저장 가능

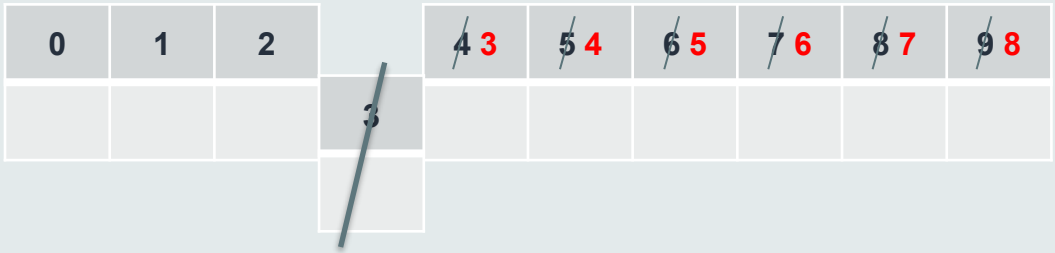
List 컬렉션

List인터페이스 메서드

기능	메소드	설명
객체 추가	boolean add(E e)	주어진 객체를 맨 끝에 추가
	void add(int index, E element)	주어진 인덱스에 객체를 추가
	set(int index, E element)	주어진 인덱스의 객체를 새로운 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( )	저장되어 있는 전체 객체 수를 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

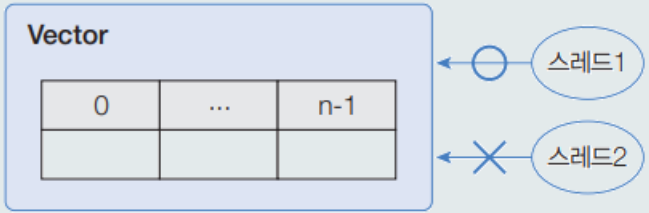
List 컬렉션

- ArrayList
  - 저장 용량(capacity)
  - 초기 용량 : 10 (따로 지정 가능)
- 저장 용량을 초과한 객체들이 들어오면 자동적으로 늘어남. 고정도 가능
- 객체 제거
  - 제거된 객체 바로 뒤 인덱스부터 마지막 인덱스까지 모두 앞으로 1씩 당겨짐



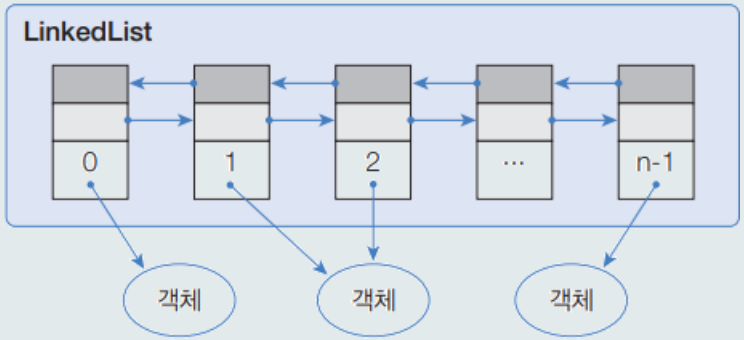
LIST 컬렉션

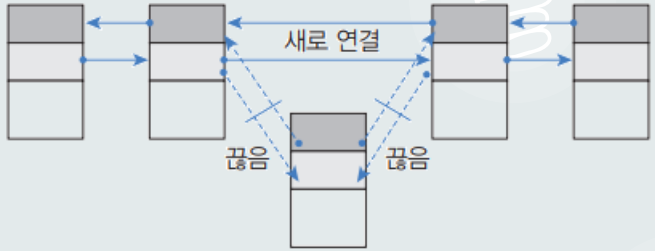
- Vector
  - 사용은 ArrayList와 동일
  - 동기화된 메소드로 구성
    - 멀티 스레드가 동시에 Vector() 메소드를 실행할 수 없음
    - 멀티 스레드 환경에서는 안전하게 객체를 추가 또는 삭제할 수 있음
    - ArrayList는 멀티 스레드 동기화를 위한 시간 소모가 없기 때문에 Vector 보다 속도가 빠르므로 단일 스레드 응용에는 더 효과적임



List 컬렉션

- LinkedList
  - 인접 객체를 체인처럼 연결해서 관리
  - 객체 삭제와 삽입이 빈번한 곳에서 ArrayList보다 유리





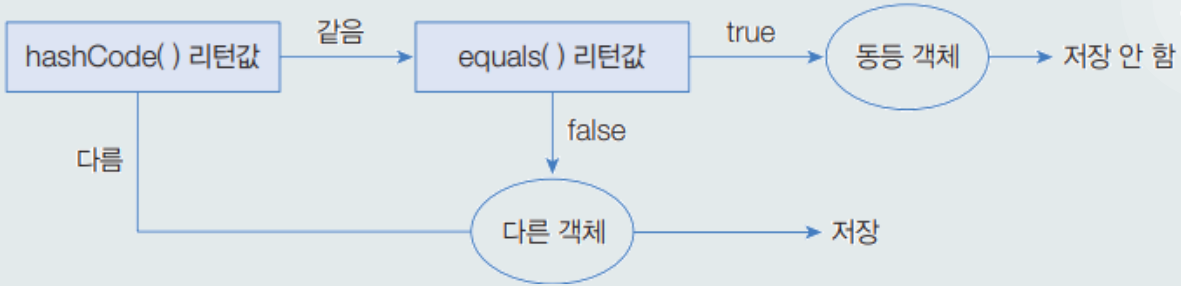
SET 컬렉션

- 수학의 집합 개념
- 저장 순서가 유지되지 않음
- 객체를 중복 저장 불가
- 하나의 null만 저장 가능

기능	메소드	설명
객체 추가	boolean add(E e)	주어진 객체를 성공적으로 저장하면 true를 리턴하고 중복 객체면 false를 리턴
객체 검색	boolean contains(Object o)	주어진 객체가 저장되어 있는지 여부
	isEmpty()	컬렉션이 비어 있는지 조사
	Iterator<E> iterator()	저장된 객체를 한 번씩 가져오는 반복자 리턴
	int size()	저장되어 있는 전체 객체 수 리턴
객체 삭제	void clear()	저장된 모든 객체를 삭제
	boolean remove(Object o)	주어진 객체를 삭제

SET 컬렉션

- HashSet
  - hashCode() 메소드의 리턴값이 같고, equals() 메소드가 true를 리턴하면 동일한 객체라고 판단하고 중복 저장하지 않음



SET 컬렉션

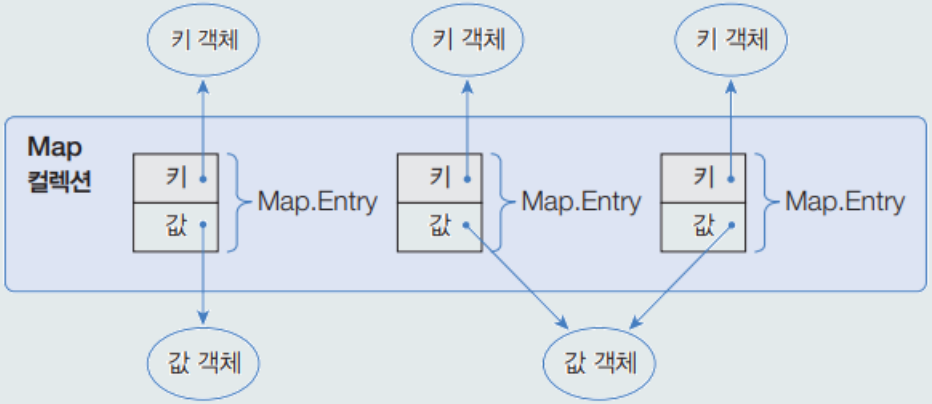
- iterator() 메소드
  - 저장된 객체를 한번씩 가져오는 반복자 리턴
  - 특정 객체만 가져올 수는 없다

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

리턴 타입	메소드명	설명
boolean	hasNext( )	가져올 객체가 있으면 true를 리턴하고 없으면 false를 리턴한다.
E	next( )	컬렉션에서 하나의 객체를 가져온다.
void	remove( )	next( )로 가져온 객체를 Set 컬렉션에서 제거한다.

MAP 컬렉션

- 키와 값으로 구성된 엔트리 객체를 저장
- 키는 중복 저장할 수 없지만 값은 중복 저장할 수 있음
  - 기존에 저장된 키와 동일한 키로 값을 저장하면 새로운 값으로 대치



MAP 컬렉션

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 Set에 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 Collection에 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

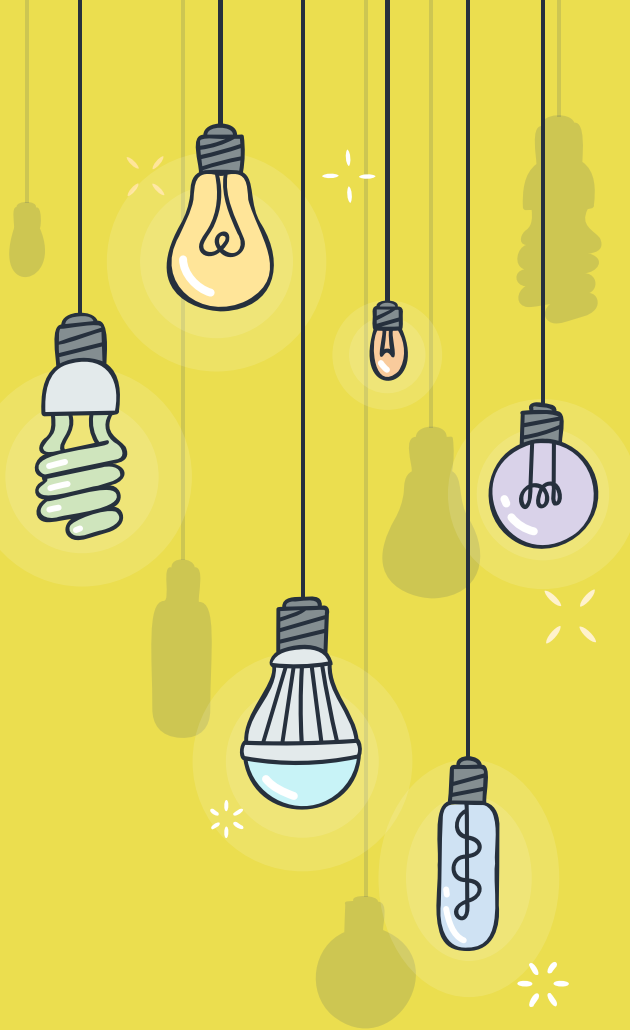
MAP 컬렉션

- HashMap
  - 키로 사용할 객체가 hashCode() 메소드의 리턴값이 같고 equals() 메소드가 true를 리턴할 경우 동일 키로 보고 중복 저장을 허용하지 않음

```
Map<K, V> map = new HashMap<K, V>();
    ↑   ↑           ↑   ↑
  키 타입 값 타입   키 타입 값 타입
```

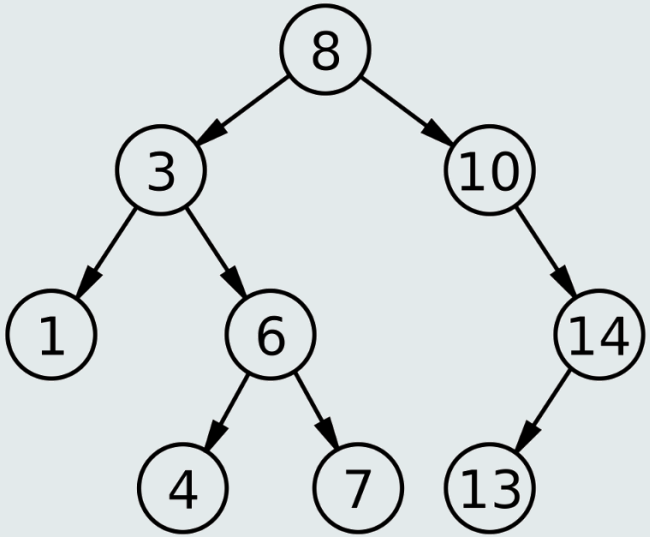
  - 키 타입은 String 많이 사용
    - String은 문자열이 같을 경우 동등 객체가 될 수 있도록 hashCode()와 equals() 메소드가 재정의되어 있기 때문

2 검색기능 컬렉션



검색 기능 컬렉션

- 이진 트리를 이용한 계층적 구조



TreeSet

- 이진 트리를 기반으로 한 Set 컬렉션
- 검색 관련 메소드가 TreeSet에 정의되어 있음
  - Set에는 없다

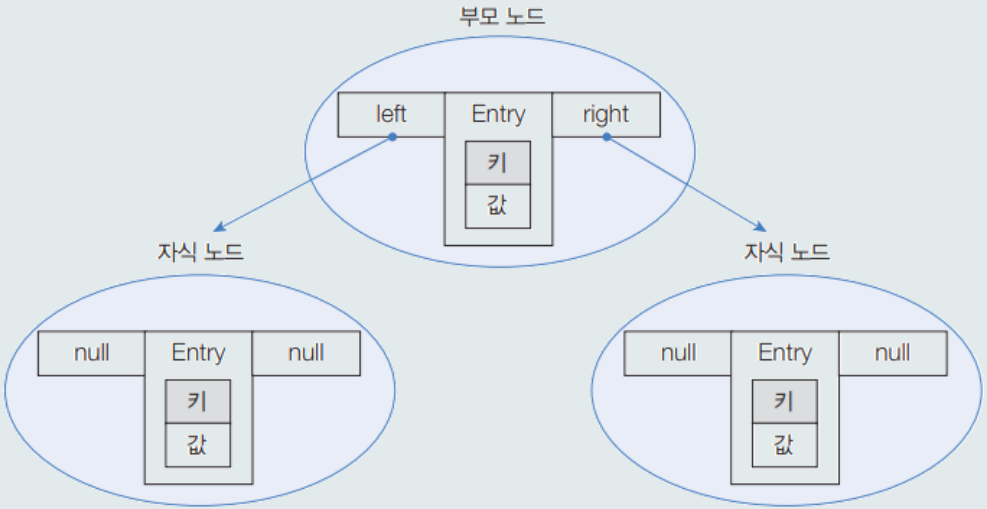
```
TreeSet<E> treeSet = new TreeSet<E>();
TreeSet<E> treeSet = new TreeSet<>();
```

리턴 타입	메소드	설명
E	first( )	제일 낮은 객체를 리턴 정렬된 순서에서 첫번째 객체
E	last( )	제일 높은 객체를 리턴 정렬된 순서에서 마지막 객체
E	lower(E e)	주어진 객체보다 바로 아래 객체를 리턴
E	higher(E e)	주어진 객체보다 바로 위 객체를 리턴
E	floor(E e)	주어진 객체와 동등한 객체가 있으면 리턴. 만약 없다면 주어진 객체의 바로 아래의 객체를 리턴
E	ceiling(E e)	주어진 객체와 동등한 객체가 있으면 리턴. 만약 없다면 주어진 객체의 바로 위의 객체를 리턴
E	pollFirst( )	제일 낮은 객체를 꺼내오고 컬렉션에서 제거함
E	pollLast( )	제일 높은 객체를 꺼내오고 컬렉션에서 제거함
NavigableSet〈E〉	headSet(E toElement, boolean inclusive)	주어진 객체보다 낮은 객체들을 NavigableSet으로 리턴. 주어진 객체 포함 여부는 두 번째 매개값에 따라 달라짐
NavigableSet〈E〉	tailSet(E fromElement, boolean inclusive)	주어진 객체보다 높은 객체들을 NavigableSet으로 리턴. 주어진 객체 포함 여부는 두 번째 매개값에 따라 달라짐
NavigableSet〈E〉	subSet(E fromElement, boolean fromInclusive, E toElement, boolean toInclusive)	시작과 끝으로 주어진 객체 사이의 객체들을 NavigableSet으로 리턴. 시작과 끝 객체의 포함 여부는 두 번째, 네 번째 매개값에 따라 달라짐

TreeMap

- 이진 트리를 기반으로 한 Map 컬렉션
- 키와 값이 저장된 엔트리 저장

```
TreeMap<K, V> treeMap = new TreeMap<K, V>();
TreeMap<K, V> treeMap = new TreeMap<>();
```



Comparable과 Comparator

- TreeSet과 TreeMap의 자동 정렬
  - TreeSet의 객체와 TreeMap의 키는 저장과 동시에 자동 오름차순 정렬
  - TreeSet과 TreeMap은 정렬을 위해 java.lang.Comparable을 구현한 객체를 요구
    - Integer, Double, String은 모두 Comparable 인터페이스를 구현한 클래스
    - Comparable을 구현하지 않을 경우 저장하는 순간 ClassCastException이 발생

Comparable과 Comparator

- 사용자 정의 객체를 정렬하고 싶을 경우
  - 사용자 정의 클래스가 Comparable을 구현

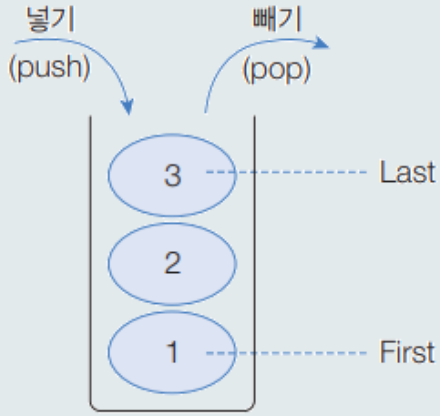
리턴타입	메소드	설명
int	compareTo(T o)	주어진 객체와 같으면 0 리턴 주어진 객체보다 적으면 음수 리턴 주어진 객체보다 크면 양수 리턴

  - TreeSet, TreeMap 생성시 Comparator 구현 객체 제공

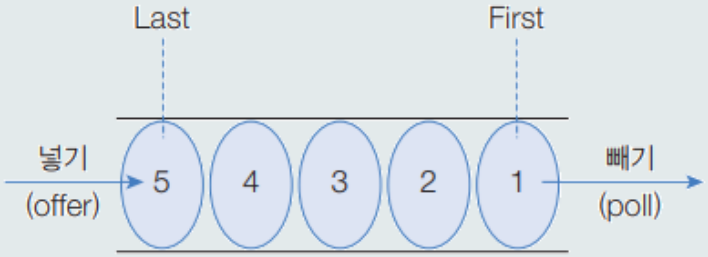
리턴타입	메소드	설명
int	compare(T o1, T o2)	o1과 o2가 동등하면 0리턴 o1이 o2보다 앞에 오게 하려면 음수 리턴 o1이 o2보다 뒤에 오게 하려면 양수 리턴

LIFO와 FIFO 컬렉션

- 후입선출(LIFO):
  - 나중에 넣은 객체가 먼저 빠져나가는 구조
- 선입선출(FIFO):
  - 먼저 넣은 객체가 먼저 빠져나가는 구조
- LIFO 자료구조를 제공 => 스택 클래스
- FIFO 자료구조를 제공=> 큐 인터페이스



스택(LIFO)



큐(FIFO)

STACK

Stack

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

리턴 타입	메소드	설명
E	push(E item)	주어진 객체를 스택에 넣는다.
E	pop( )	스택의 맨 위 객체를 빼낸다.

Queue

```
Queue<E> queue = new LinkedList<E>();
Queue<E> queue = new LinkedList<>();
```

리턴 타입	메소드	설명
boolean	offer(E e)	주어진 객체를 큐에 넣는다.
E	poll( )	큐에서 객체를 빼낸다.

Thanks!

- Any questions?