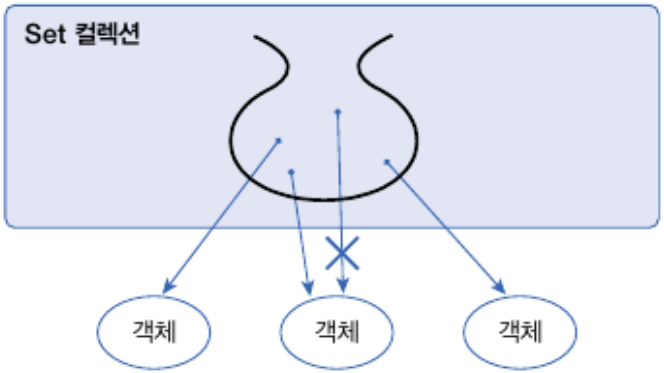
Set 컬렉션

▣ Set 컬렉션

- Set 컬렉션은 구술 주머니와 같다. 동일한 구슬을 두 개 넣을 수 없으며, 들어갈(저장할) 때와 나올(찾을) 때의 순서가 다를 수도 있기 때문이다.

- Set 컬렉션에는 HashSet, LinkedHashSet, TreeSet 등이 있는데, Set 컬렉션에서 공통적으로 사용 가능한 Set 인터페이스의 메소드를 보면, 인덱스로 관리하지 않기 때문에 인덱스를 매개 값으로 갖는 메소드가 없다.



가능	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

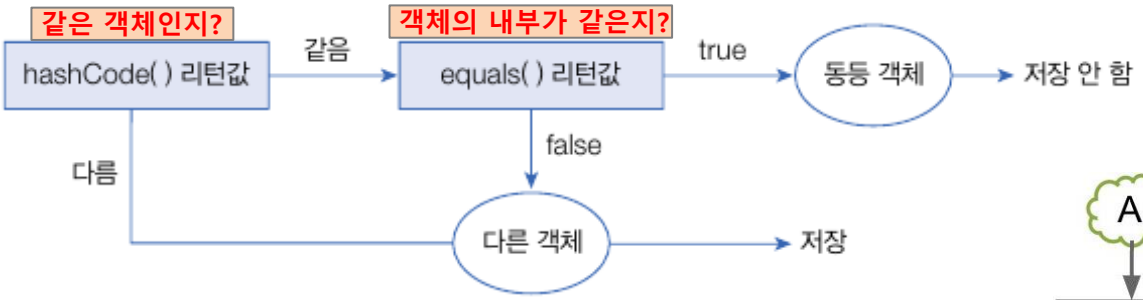
◎ HashSet

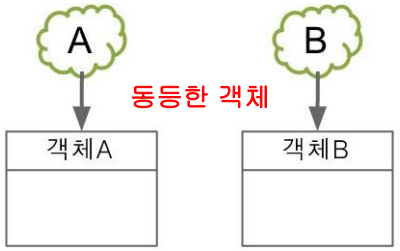
- boolean add(Object o)시 저장할 객체(o)의 hashCode() 와 equals() 를 자동 호출한다.

- HashSet은 동일한 객체는 중복 저장하지 않는다. 여기서 동일한 객체란 동등 객체를 말한다.

- HashSet은 다른 객체라도 hashCode() 메소드의 리턴값이 같고, equals() 메소드가 true를 리턴하면 동일한 객체라고 판단하고 중복 저장하지 않는다.

- 문자열을 HashSet에 저장할 경우는 같은 문자열을 갖는 String 객체는 동등한 객체로 간주한다. 같은 문자열이면 hashCode()의 리턴값이 같고, equals()의 리턴값이 true가 나오기 때문이다.



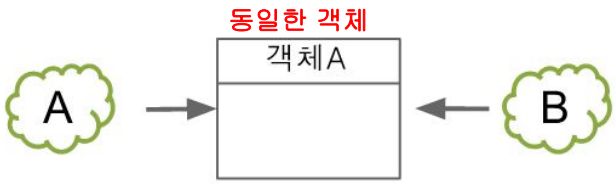


hashCode()

- 두 객체가 같은 객체인지 확인하는 Method입니다.

equals()

- 두 객체의 내용이 같은지 확인하는 Method입니다.



- Set 컬렉션은 인덱스로 객체를 검색해서 가져오는 메소드가 없다.
- 대신 객체를 한 개씩 반복해서 가져와야 하는데, 여기에는 두 가지 방법이 있다.

<방법1> for-each 문을 이용하는 방법

```
Set<E> set = new HashSet<>();
for(E e : set) {
   ...
}
```

<방법2> 일반 for문을 이용하는 방법

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

for (int i = 0; i < set.size(); i++) {
  System.out.print(set.get(i) + " ");
    ...
}
```

<방법3> Set 컬렉션의 iterator() 메소드로 반복자(iterator)를 얻어 객체를 하나씩 가져오는 방법

- List, Set에는 있으나, Map에는 없음(Map -> Set -> iterator( ) : iterator 객체 얻음)

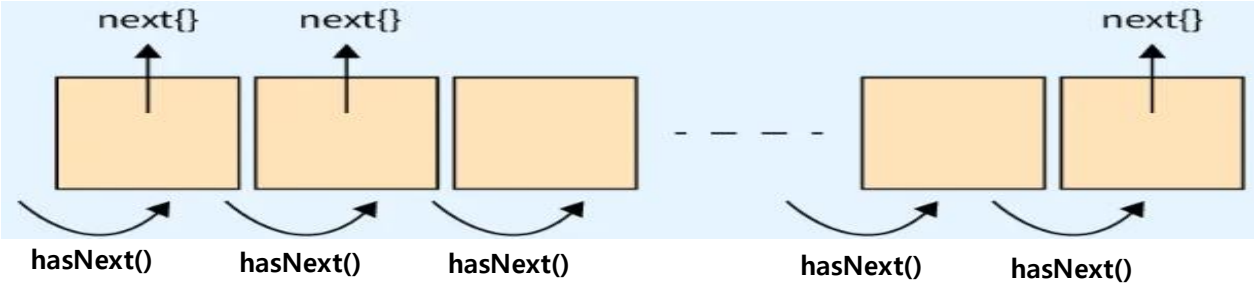
- Iterator객체는 일회용이기 때문에,
  얻은 Iterator를 사용한 경우에는, 다시 Iterator객체를 얻어서 사용해야 한다.

iterator는 Set 컬렉션의 객체를 가져오거나 제거하기 위해 다음 메소드를 제공한다.
hasNext() 메소드로 가져올 객체가 있는지 먼저 확인하고, true를 리턴할 때만 next() 메소드로 객체를 가져온다.
만약 next()로 가져온 객체를 컬렉션에서 제거하고 싶다면 remove() 메소드를 사용한다.

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.

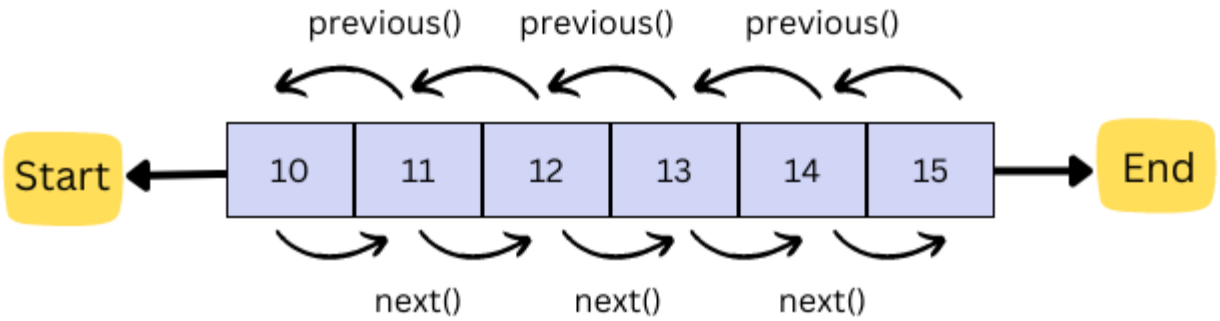
```
Set<E> set = new HashSet<>();
Iterator<E> iterator = set.iterator();
```

```
while(iterator.hasNext()) {
  E e = iterator.next();
}
```



Enumeration, Iterator, ListIterator

- 컬렉션에 저장된 데이터를 접근하는데 사용되는 인터페이스
- Enumeration은 Iterator의 구버전
- ListIterator는 Iterator의 접근성을 향상시킨 것 (단방향 → 양방향)



Iterator

메서드	설 명
boolean hasNext( )	읽어 올 요소가 남아있는지 확인한다. 있으면 true, 없으면 false를 반환한다.
Object next( )	다음 요소를 읽어 온다. next( )를 호출하기 전에 hasNext( )를 호출해서 읽어 올 요소가 있는지 확인하는 것이 안전하다.
void remove( )	next( )로 읽어 온 요소를 삭제한다. next( )를 호출한 다음에 remove( )를 호출해야한다.(선택적 기능)

Enumeration

메서드	설 명
boolean hasMoreElements( )	읽어 올 요소가 남아있는지 확인한다. 있으면 true, 없으면 false를 반환한다. Iterator의 hasNext( )와 같다.
Object nextElement( )	다음 요소를 읽어 온다. nextElement( )를 호출하기 전에 hasMoreElements( )를 호출해서 읽어올 요소가 남아있는지 확인하는 것이 안전하다. Iterator의 next( )와 같다.

ListIterator

메서드	설 명
boolean hasNext( )	읽어 올 다음 요소가 남아있는지 확인한다. 있으면 true, 없으면 false를 반환한다.
boolean hasPrevious( )	읽어 올 이전 요소가 남아있는지 확인한다. 있으면 true, 없으면 false를 반환한다.
Object next( )	다음 요소를 읽어 온다. next( )를 호출하기 전에 hasNext( )를 호출해서 읽어 올 요소가 있는지 확인하는 것이 안전하다.
Object previous( )	이전 요소를 읽어 온다. previous( )를 호출하기 전에 hasPrevious( )를 호출해서 읽어 올 요소가 있는지 확인하는 것이 안전하다.