Collection Framework

컬렉션(Collection) → 여러 객체를 모아놓은 것 (객체의 저장을 뜻한다)

프레임워크 (Framework) → 표준화, 정형화된 체계적인 프로그래밍 방식

컬렉션 프레임워크
→ 여러객체를 다루기 위한 표준화된 프로그래밍 방식
→ 편리하고 쉽게 다룰 수 있는 다양한 클래스를 제공
→ java.util 패키지에 포함

컬렉션 프레임워크의 주요 인터페이스로는 List, Set, Map이 있다.

Collection Framework

List
(인터페이스)

순서가 있고
중복이 허용된다

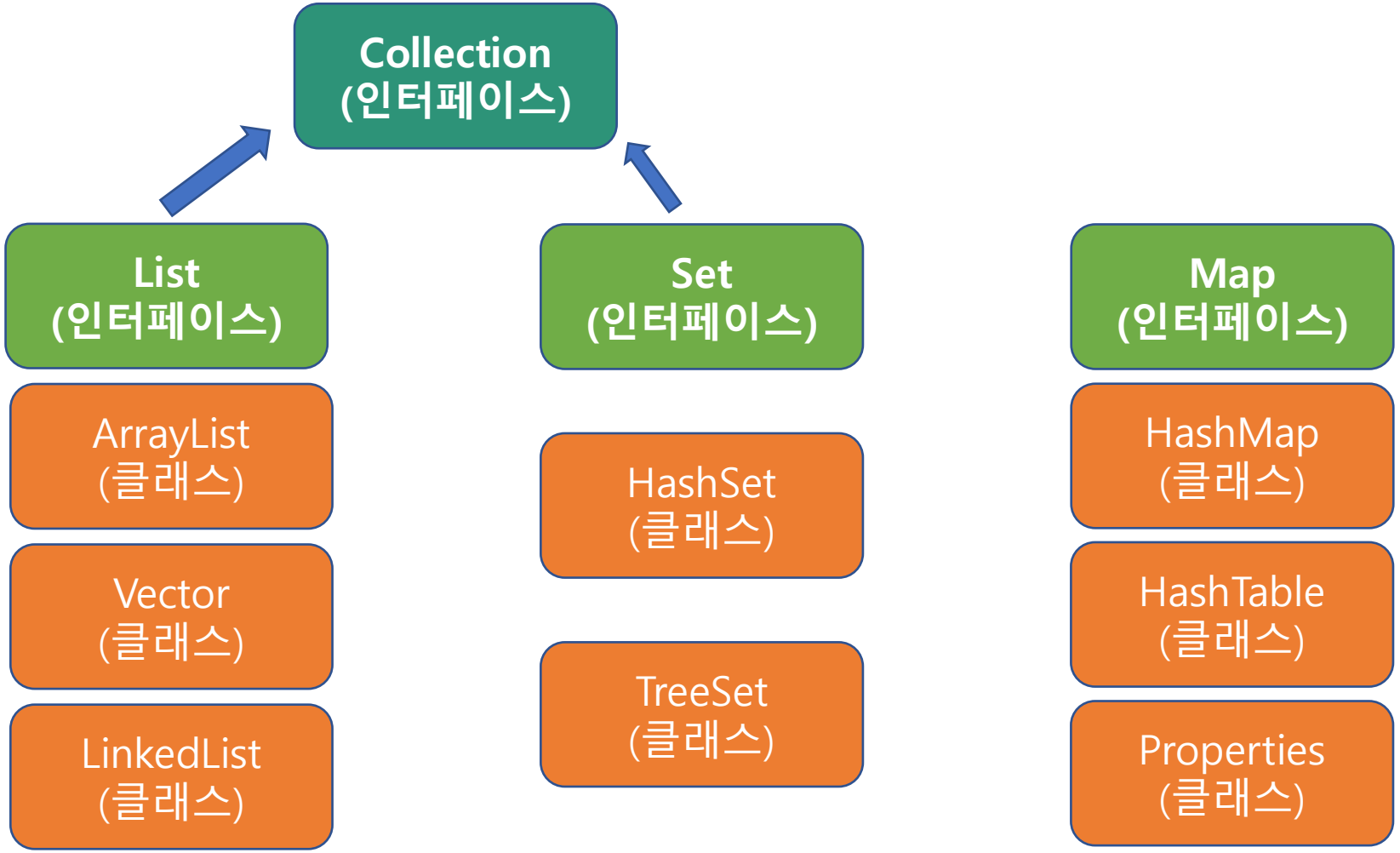
Set
(인터페이스)

순서가 없고
중복이 안된다.

Map
(인터페이스)

순서가 없고
중복은
Key는 X
Value는 O

Collection Framework



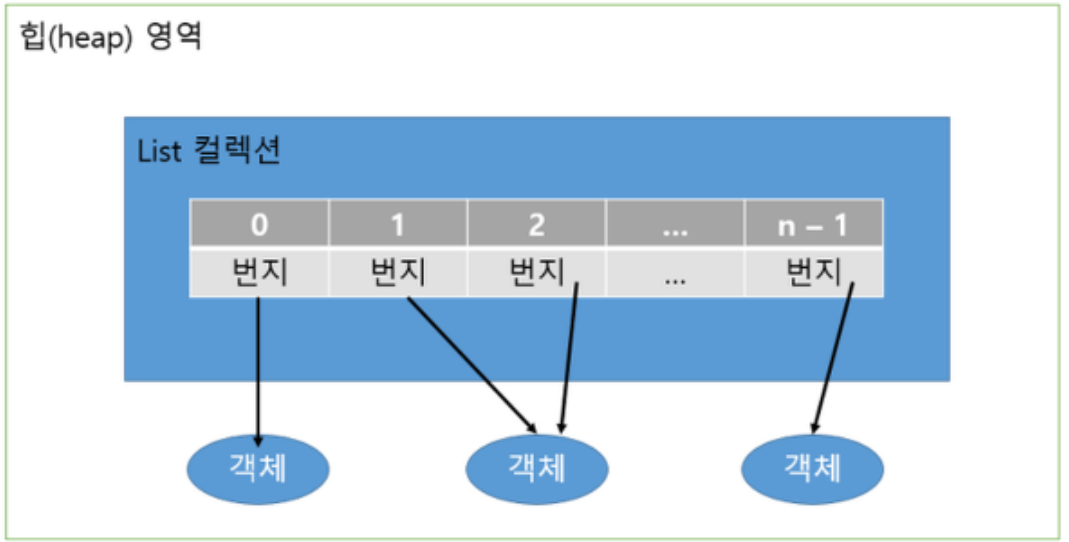
List 컬렉션

List 컬렉션은 배열과 비슷하게 객체를 인덱스로 관리한다.

저장용량이 자동으로 증가하며, 객체를 저장할 때 자동 인덱스가 부여된다.

(추가, 삭제, 검색을 위한 다양한 메소드 제공)

List 컬렉션은 객체 자체를 저장하는 것이 아니라 객체의 번지를 참조한다. 그렇기 때문에 동일한 객체를 중복 저장할 수 있다.



List 컬렉션

List 컬렉션에 저장된 모든 객체를 대상으로 하나씩 가져와 처리하고 싶다면 인덱스를 이용하는 방법(일반 for문)과 향상된 for문을 사용하는 방법이 있다.

```
List<Integer> list = new ArrayList<Integer>();

for (int i = 0; i < list.size(); i++) {
    list.get(i);
}

for(int i : list) {
    System.out.println(i);
}
```

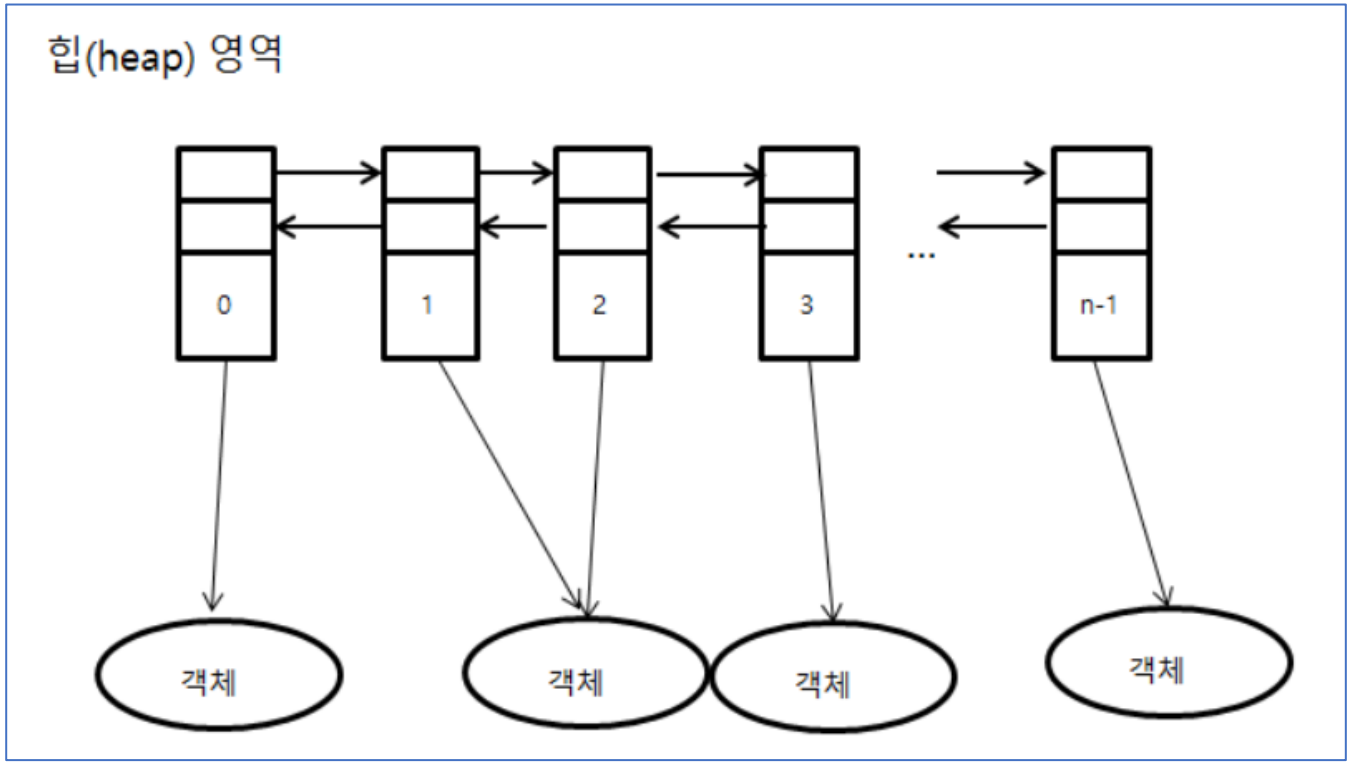
ArrayList<E>

ArrayList는 List 인터페이스의 대표적인 구현 클래스이다.

List 인터페이스를 구현함으로 저장순서가 유지, 중복이 허용된다.

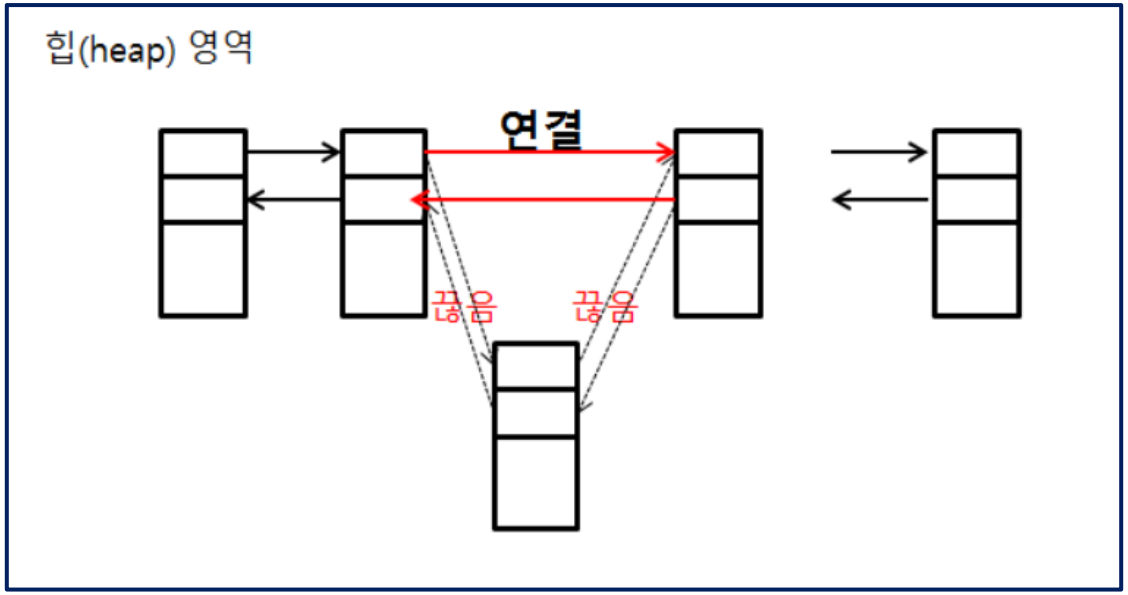
LinkedList<E>

LinkedList는 List구현클래스이므로 ArrayList와 사용방법은 똑같은데, 내부구조는 다르다. LinkedList는 인접 참조를 링크해서 체인처럼 관리한다.



LinkedList<E>

LinkedList에서 특정 인덱스의 객체를 삽입, 제거하면 앞뒤 링크만 변경된다.



ArrayList는 중간 인덱스의 객체를 제거하면 뒤에 있는 객체의 인덱스가 1씩 앞으로 당겨지기 때문에 빈번한 객체 삭제 / 삽입이 일어나는 곳에서는 LinkedList가 좋은 성능을 발휘한다.

Iterator (반복자)

컬렉션 프레임워크에서 저장된 요소를 읽어오는 방법을 표준화 한 것 (인터페이스)

컬렉션 프레임워크에 대해 공통으로 사용이 가능하고 사용법이 간단하다.

List, Set계열에 구현(Map은 없음)

주로 읽기 전용으로 사용

메소드	설명
hasNext()	가져올 객체가 있으면 true를 리턴하고 없으면 false를 리턴한다.
next()	컬렉션에서 다음 하나의 객체를 가져온다.
remove()	next()로 읽어 온 요소를 삭제한다.

Iterator (반복자)

Iterator에서 하나의 객체를 가져올 때는 next() 메소드를 사용한다.

next() 메소드를 사용하기 전에 먼저 가져올 객체가 있는지 확인하는 것이 좋다.

hasNext()메소드는 가져올 객체가 있으면 true를 리턴하고 객체가 없으면 false를 리턴한다. 따라서 true가 리턴될 때 next()메소드를 사용하는것이 바람직하다.

```
while(itr.hasNext()) {
    String str = itr.next();
    System.out.println(str);
}
```

Set 컬렉션

List 컬렉션은 객체의 저장 순서를 유지하지만, Set컬렉션은 저장 순서가 유지되지 않는다. 또한 객체를 중복해서 저장할 수 없다. Set 컬렉션은 수학의 집합에 비유될 수 있다. 집합은 순서와 상관없고 중복이 허용되지 않기 때문이다. 그리고 들어갈 때의 순서와 나올때의 순서가 다르다.

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

set.add("홍길동");
set.add("성춘향");
set.add("홍길동"); //  중복X
```

Set 컬렉션

Set 컬렉션은 인덱스로 저장되지 않기 때문에 인덱스로 객체를 검색해서 가져오는 메소드가 없다. 대신, 전체 객체를 대상으로 한번씩 반복해서 가져오는 반복자(Iterator)를 제공한다.

```
Set<String> set = new HashSet<>();
set.add("홍길동");
set.add("이순신");    //객체 추가
set.remove("홍길동"); //객체삭제
```

```
Iterator<String> itr = set.iterator();
```

HashSet

HashSet은 Set 인터페이스의 구현 클래스이다. HashSet은 객체들을 순서 없이 저장하고 동일한 객체는 중복 저장하지 않는다. HashSet이 판단하는 동일한 객체란 꼭 같은 인스턴스를 뜻하지는 않는다.

HashSet은 객체를 저장하기 전에 먼저 객체의 hashCode() 메소드를 호출해서 해시코드를 얻어낸다. 그리고 이미 저장되어 있는 객체들의 해시코드와 비교한다.
만약 동일한 해시코드가 있다면 다시 equals() 메소드로 두 객체를 비교해서 true가 나오면 동일한 객체로 판단하고 중복 저장을 하지 않는다.

HashSet

```
Set<String> set = new HashSet<>();
set.add("홍길동");
set.add("이순신");
set.add("홍길동");    //"홍길동"은 한번만 저장됨.

Iterator<String> itr = set.iterator();

while(itr.hasNext()) {
    String str = itr.next();
    System.out.println(str);
}
```

```
홍길동
이순신
```