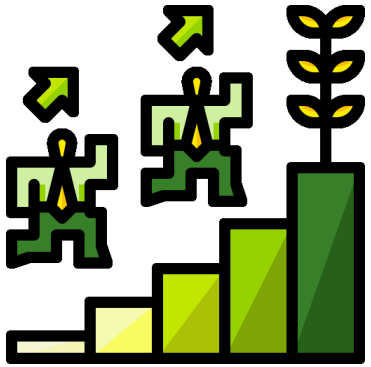


JAVA Programming

컬렉션

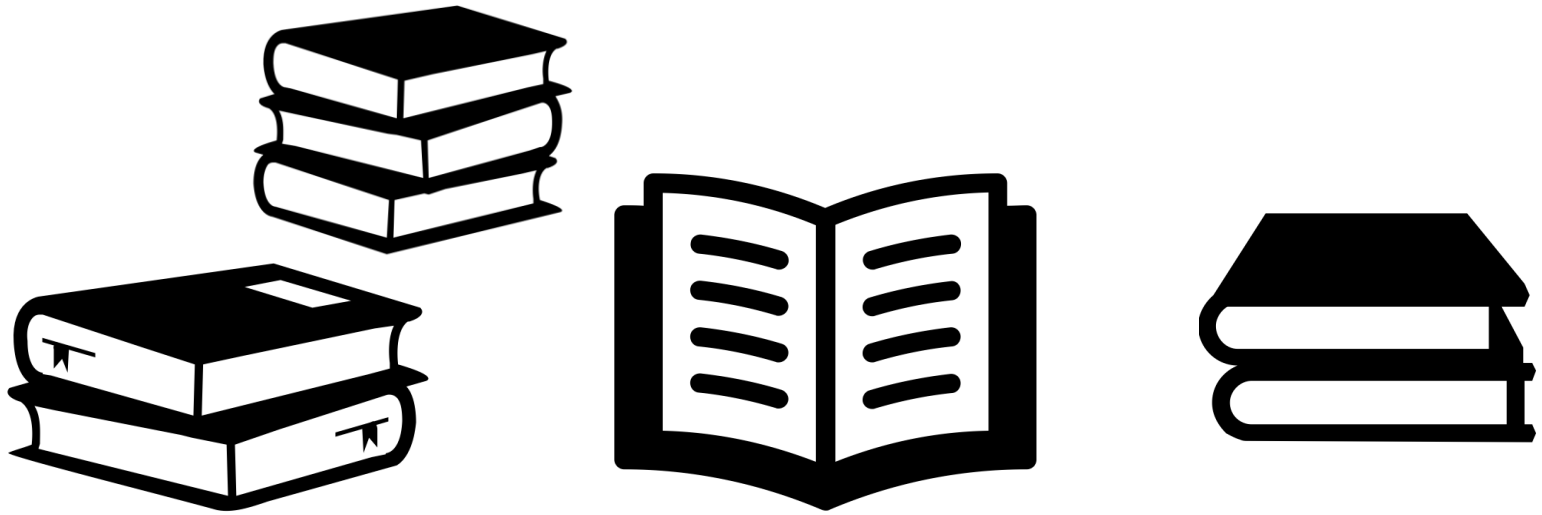
컬렉션

- 컬렉션(Collection)이란?

  여러 개의 다양한 데이터들을 쉽고 효과적으로 처리할 수 있도록 표준화 된 방법을 제공하는 클래스들의 집합

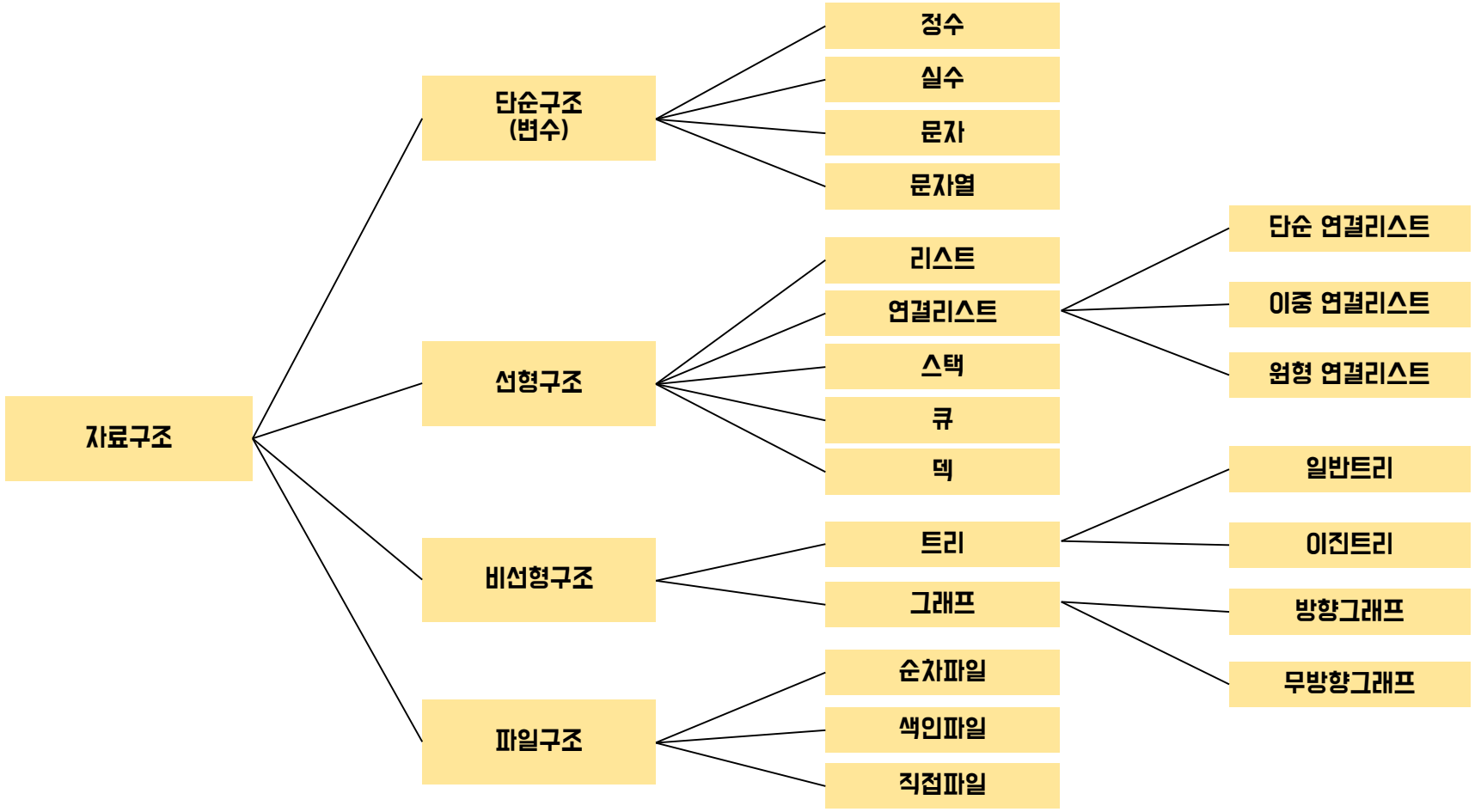
  (데이터를 효율적으로 저장하는 **자료구조**와 데이터를 처리하는 **알고리즘**이 미리 구현되어 있음)

- Java.util 패키지에 포함

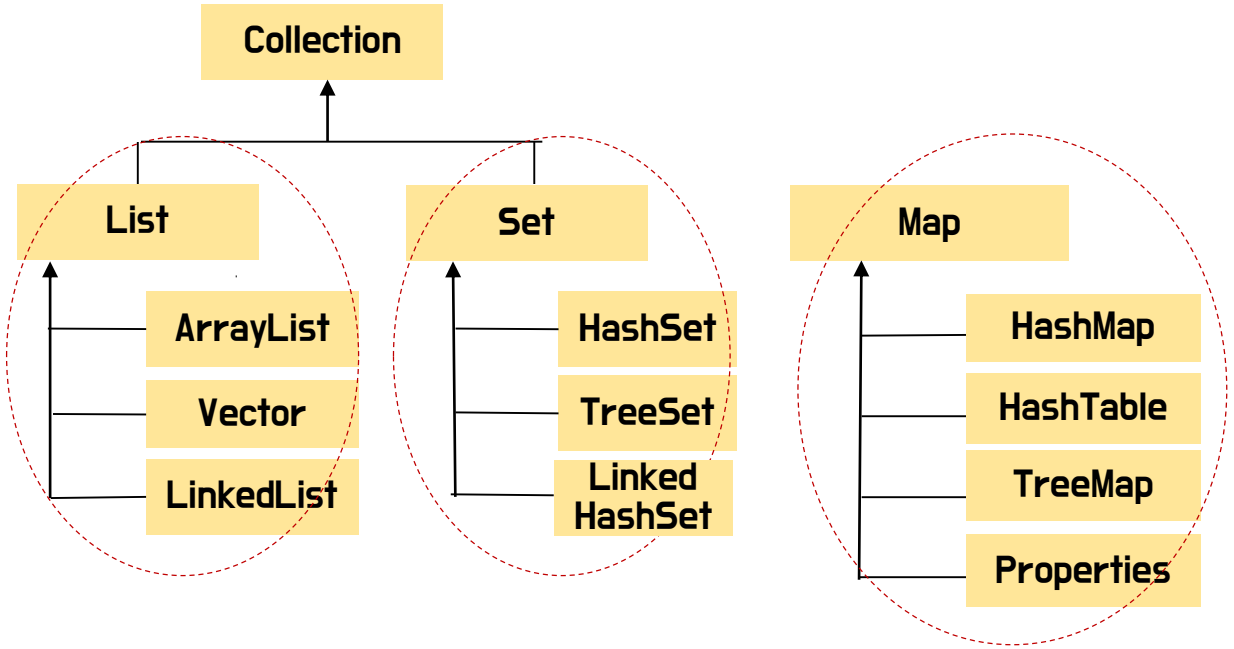


자료구조

- 데이터(자료)를 메모리에서 효율적으로 저장하기 위한 방법론



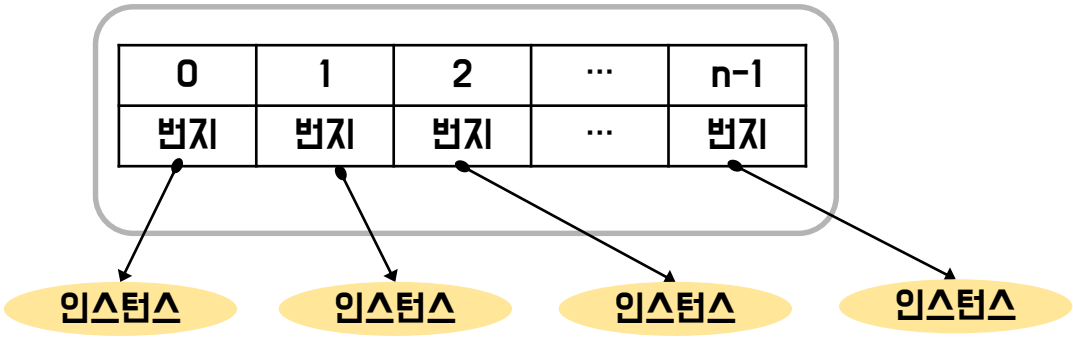
컬렉션의 주요 인터페이스

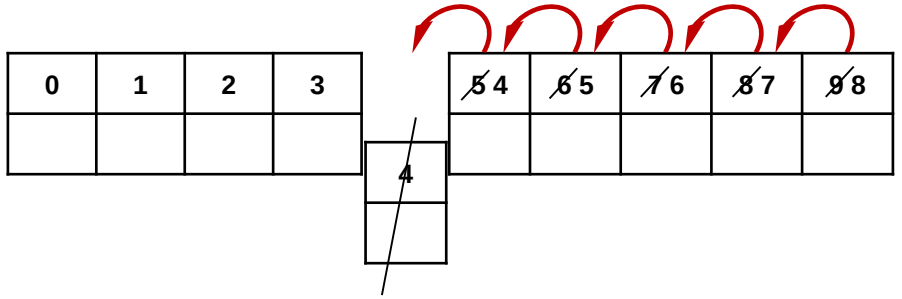


인터페이스 분류		특징	구현 클래스
Collection	List 계열	순서를 유지하고 저장 중복 저장 가능	ArrayList, Vector, LinkedList
	Set계열	순서를 유지하지 않고 저장 중복 저장 안됨	HashSet, LinkedHashSet, TreeSet
Map 계열		키와 값의 쌍으로 저장 키는 중복 저장 안됨	HashMap, HashTable, TreeMap, Properties

List

- 자료들을 순차적으로 나열한 자료구조로 인덱스로 관리되며, 중복해서 인스턴스 저장이 가능
- 구현 클래스 : ArrayList,Vector,LinkedList





com.greedy.section01.list.run.Application1

List

- List 계열 주요 메소드

기능	메소드	리턴타입	설명
인스턴스 추가	add(E e)	boolean	주어진 인스턴스를 맨 끝에 추가
	add(int index, E element)	void	주어진 인덱스에 인스턴스를 추가
	addAll(Collection<? extends E> c)	boolean	주어진 Collection타입 인스턴스를 리스트에 추가
	set(int index, E element)	E	주어진 인덱스에 저장된 인스턴스를 주어진 인스턴스로 바꿈
인스턴스 검색	contains(Object o)	boolean	주어진 인스턴스가 저장되어 있는지 여부
	get(int index)	E	주어진 인덱스에 저장된 인스턴스를 리턴
	iterator()	Iterator<E>	저장된 인스턴스를 한번씩 가져오는 반복자 리턴
	isEmpty()	boolean	컬렉션이 비어 있는지 조사
	size()	int	저장되어 있는 전체 인스턴스 수를 리턴
인스턴스 삭제	clear()	void	저장된 모든 인스턴스를 삭제
	remove(int index)	E	주어진 인덱스에 저장된 인스턴스를 삭제
	remove(Object o)	boolean	주어진 인스턴스를 삭제

List

- ArrayList

  가장 많이 사용되는 컬렉션 클래스

  내부적으로 배열을 이용하여 요소를 관리하며, 인덱스를 이용해 배열 요소에 접근 가능

ex) List<E> list = new ArrayList<E>();

ArrayList

* 동기화 : 하나의 자원(데이터)에 대해 여러 스레드가 접근 하려 할 때 한 시점에서 하나의 스레드만 사용할 수 있도록 하는 것

com.greedy.section01.list.run.Application2

배열의 문제점

- 한번 크기를 지정하면 변경할 수 없음
  (저장할 공간의 크기가 부족 시 에러 발생 -> 할당 시 넉넉한 크기로 할당 (메모리 낭비)
  필요에 따라 공간을 늘리거나 줄이기 힘듬)

- 배열에 기록된 데이터에 대한 중간 위치의 추가, 삭제가 불편
  (추가, 삭제 할 데이터부터 마지막 기록된 데이터까지 하나씩 뒤로 밀어내고 추가 (복잡한 알고리즘)

- 한 타입의 데이터만 저장 가능

ArrayList의 특징

- 저장하는 크기의 제약이 없음

- 추가, 삭제, 정렬 등의 기능 처리가 간단하게 해결
  (자료구조가 내장되어 있어 따로 복잡한 알고리즘 불필요)

- 여러 타입의 데이터가 저장 가능
  (다만 인스턴스만 저장할 수 있기 때문에 기본 자료형을 저장해야 할 경우 Wrapper 클래스 사용)

List

- Comparable, Comparator

	Comparable	Comparator
패키지	java.lang	java.util
사용 메소드	compareTo()	compare()
정렬	기존의 정렬기준을 구현하는데 사용	그 외 다른 여러 기준으로 정렬하고자 할 때 사용
사용법	정렬하고자 하는 인스턴스에 Comparable를 상속받아 compareTo() 메소드를 오버라이딩해 기존의 정렬 기준 재정의 → 한 개의 정렬만 가능	vo 패키지 안에 필요한 정렬 기준에 맞춘 클래스들을 생성하고 Comparator를 상속받아 compare() 메소드를 오버라이딩해 기존의 정렬 기준 재정의 → 여러 개의 정렬 가능

- Collections.sort()

Collections.sort(List<T> list) → T인스턴스에 Comparable을 상속받아 compareTo 메소드 재정의를 통해 정렬 구현
(단 한가지 기준의 정렬)

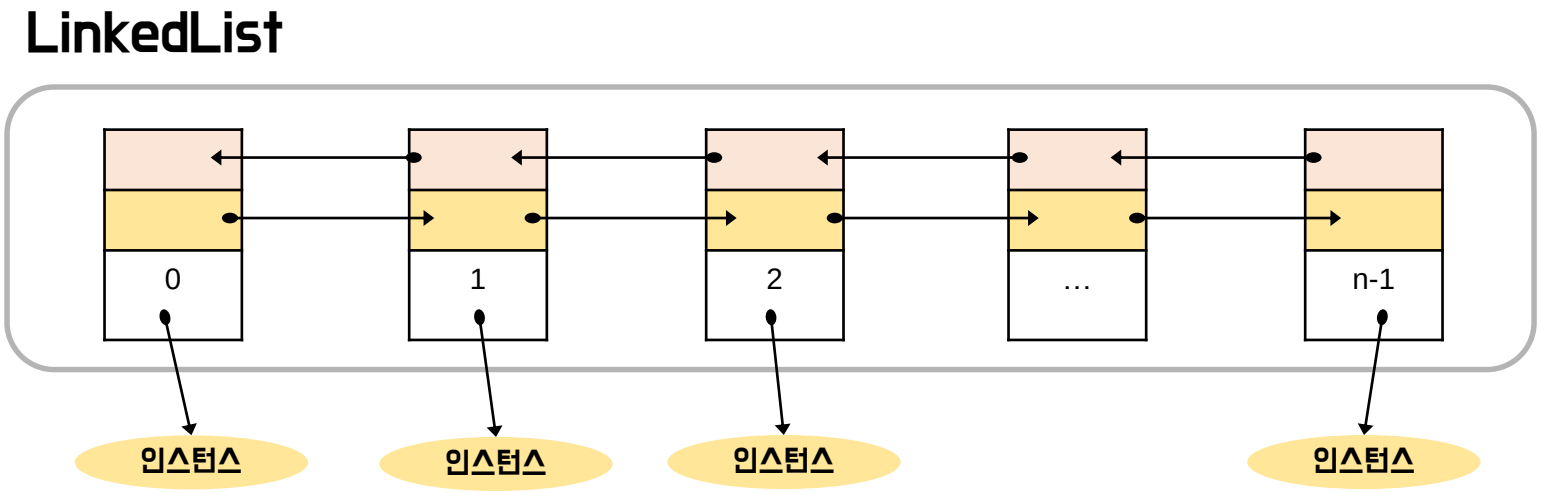
Collections.sort(List<T> list, Comparator<T> c) → 지정한 Comparator클래스에 의한 정렬
(여러 기준의 정렬)

List

- LinkedList

  인접 참조를 링크해서 체인처럼 관리

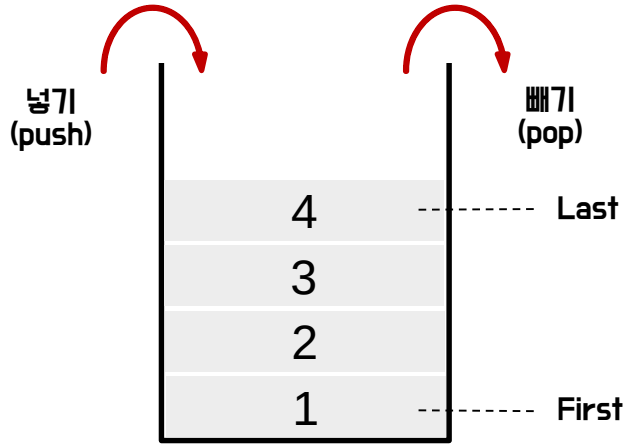
  (특정 인덱스에서 인스턴스를 제거하거나 추가하게 되면 바로 앞/뒤 링크만 변경하면 되기 때문에 인스터스 삭제와 삽입이 빈번하게 일어나는 곳에서는 ArrayList보다 성능이 뛰어남)



com.greedy.section01.list.run.Application3

Stack

- stack은 제한적으로 접근할 수 있는 나열 구조로 데이터를 저장
  후입선출(LIFO - Last Input First Out)방식의 자료 구조



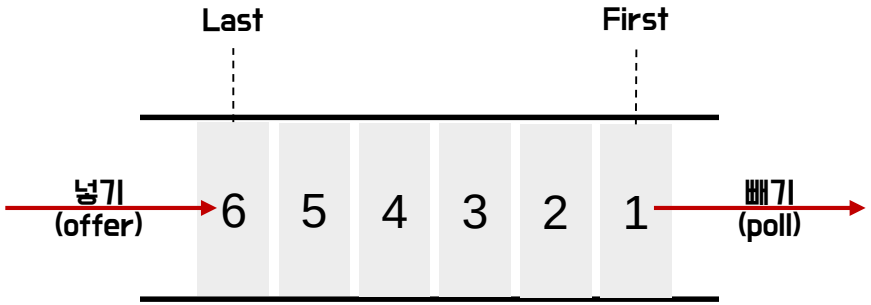
리턴 타입	메소드	설명
E	push(E item)	주어진 인스턴스를 스택에 넣는다
E	peek()	스택의 맨 위 인스턴스를 가져온다. 인스턴스를 스택에서 제거하지 않는다.
E	pop()	스택의 맨 위의 인스턴스를 가져온다. 인스턴스를 스택에서 제거한다.

Stack

com.greedy.section01.list.run.Application4

Queue

- queue는 선형 메모리 공간에 데이터를 저장
- 선입선출(FIFO - First Input First Out)방식의 자료구조



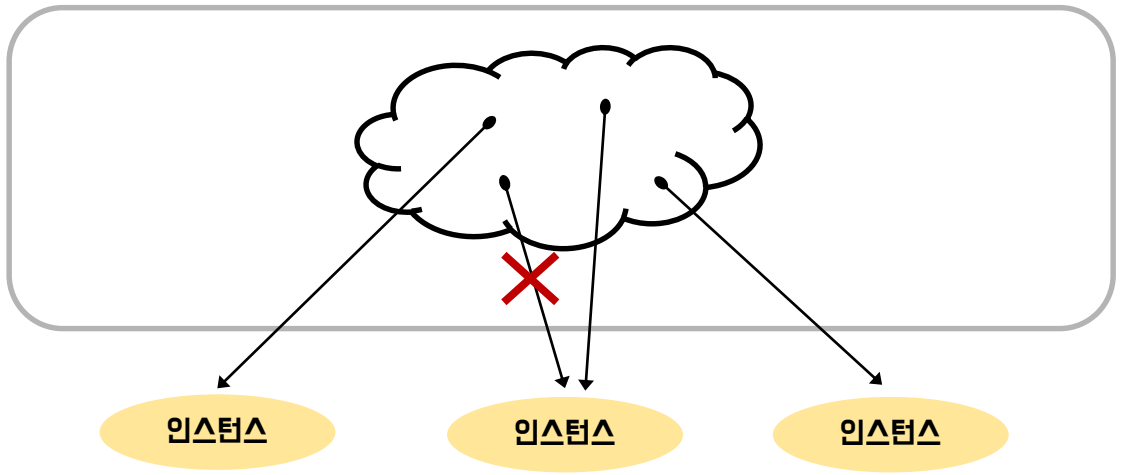
리턴 타입	메소드	설명
Boolean	offer(E e)	주어진 인스턴스를 넣는다.
E	peek()	인스턴스를 하나 가져온다. 인스턴스를 큐에서 제거하지 않는다.
E	Poll()	인스턴스를 하나 가져온다. 인스턴스를 큐에서 제거한다.

Queue

com.greedy.section01.list.run.Application5

Set

- 저장 순서가 유지되지 않고, 중복 인스턴스도 저장하지 못하게 하는 자료구조
  (null값도 중복하지 않게 하나의 null만 저장)
- 구현 클래스 : HashSet,LinkedHashSet,TreeSet



Set

com.greedy.section02.set.run.Application1

Set

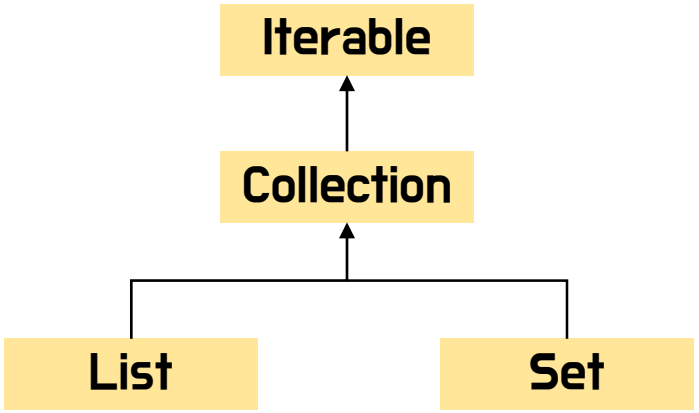
- Set 계열 주요 메소드

기능	메소드	리턴타입	설명
인스턴스 추가	add(E e)	boolean	주어진 인스턴스를 맨 끝에 추가
	addAll(Collection<? extends E> c)	boolean	주어진 Collection타입 인스턴스를 리스트에 추가
인스턴스 검색	contains(Object o)	boolean	주어진 인스턴스가 저장되어 있는지 여부
	iterator()	Iterator<E>	저장된 인스턴스를 한번씩 가져오는 반복자 리턴
	isEmpty()	boolean	컬렉션이 비어 있는지 조사
	size()	int	저장되어 있는 전체 인스턴스 수를 리턴
인스턴스 삭제	clear()	void	저장된 모든 인스턴스를 삭제
	remove(Object o)	boolean	주어진 인스턴스를 삭제

* 전체 인스턴스 대상으로 한 번씩 반복해서 가져오는 반복자(Iterator)를 제공 인덱스로 인스턴스에 접근할 수 없음

Iterator

- 컬렉션에 저장된 요소를 접근하는데 사용되는 인터페이스
  List와 Set계열에서만 사용
  (Map의 경우 Set 또는 List화 시켜서 iterator()를 사용)



Iterator<E>	boolean hasNext()	앞에서부터 검색
	E next()	
ListIterator<E>	boolean hasNext()	앞에서부터 검색
	E next()	
	boolean hasPrevious()	뒤에서부터 검색
	E previous()	

주요 메소드

Set

- HashSet

  Set에 인스턴스를 저장할 때 hash함수를 사용하여 처리 속도가 빠름

  동일 인스턴스 뿐 아니라 동등 인스턴스도 중복하여 저장하지 않음

  (동일 : 완전히 같은 인스턴스)

  (동등 : 다른 인스턴스이지만 특정한 기준들의 속성 값이 같음)

- LinkedHashSet

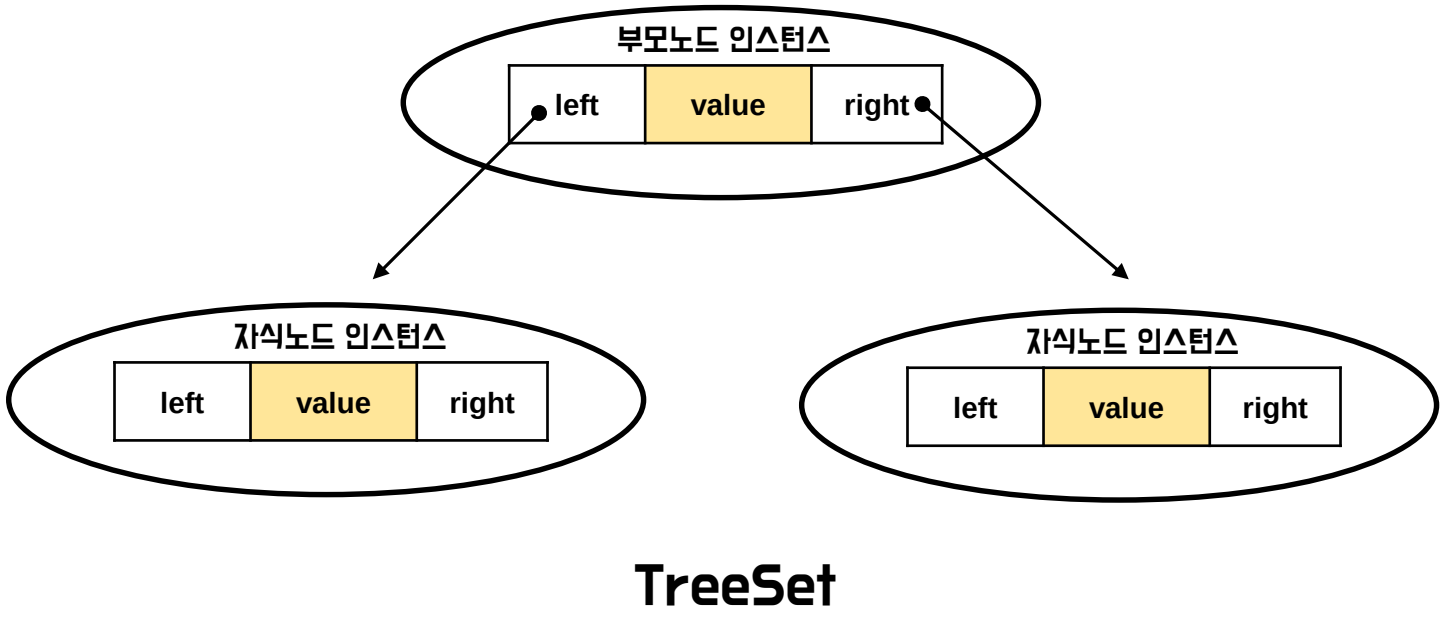
  HashSet과 거의 동일하지만 Set에 추가되는 순서를 유지함

com.greedy.section02.set.run.Application2

Set

- TreeSet

  이진 트리를 기반으로 한 Set컬렉션으로, 왼쪽과 오른쪽 자식 노드를 참조하기 위한 두 개의 변수로 구성

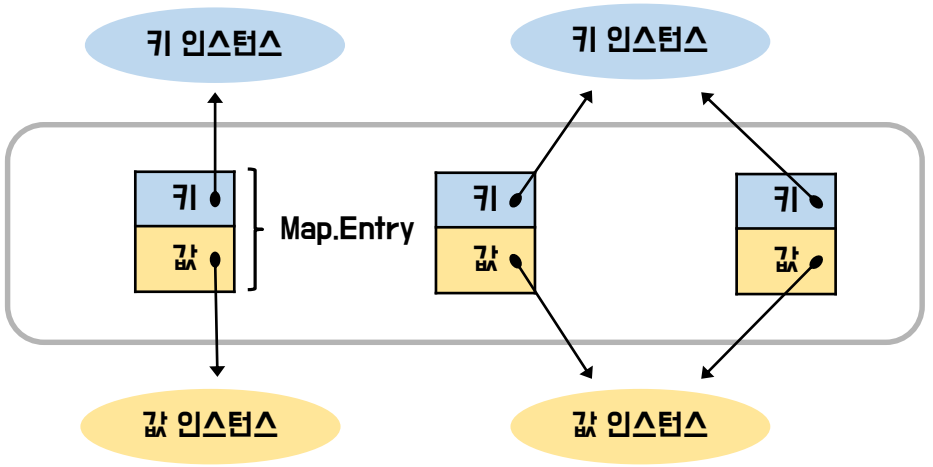


TreeSet

com.greedy.section02.set.run.Application3

Map

- 키(key)와 값(value)으로 구성되어 있으며, 키와 값은 모두 인스턴스
- 키는 중복 저장을 허용하지 않고(Set방식), 값은 중복 저장 가능(List방식)
- 키가 중복되는 경우, 기존에 있는 키에 해당하는 값을 덮어 씀
- 구현 클래스 : HashMap, HashTable, LinkedHashMap, Properties, TreeMap



Map

com.greedy.section03.map.run.Application1

Map

- Map 계열 주요 메소드

기능	메소드	리턴타입	설명
인스턴스 추가	put(K key, V value)	V	주어진 키와 값을 추가. 저장이 되면 값을 리턴
인스턴스 검색	containsKey(Object key)	boolean	주어진 키가 있는지 확인하여 결과 리턴
	containsValue(Object value)	boolean	주어진 값이 있는지 확인하여 결과 리턴
	entrySet()	Set<Map.Entry<K,V>>	키와 값의 쌍으로 구성된 모든 Map.Entry 인스턴스를 set에 담아서 리턴
	get(Object key)	V	주어진 키의 값을 리턴
	isEmpty()	boolean	컬렉션이 비어있는지 여부
	keySet()	Set<K>	모든 키를 Set 인스턴스에 담아서 리턴
	size()	int	저장된 키의 총 수를 리턴
	values()	Collection<V>	저장된 모든 값을 Collection에 담아서 리턴
인스턴스 삭제	clear()	void	모든 Map.Entry를 삭제함
	remove(Object key)	V	주어진 키와 일치하는 Map.Entry 삭제. 삭제가 되면 값을 리턴한다.

Properties

- 키와 값을 String 타입으로 제한한 Map 컬렉션
- 주로 Properties는 프로퍼티(*.properties)파일을 읽어 들일 때 주로 사용

프로퍼티(*.properties)파일
- 옵션정보, 데이터베이스 연결정보, 국제화(다국어)정보를 기록하여 텍스트 파일로 활용
- 애플리케이션에서 주로 변경이 잦은 문자열을 저장하여 관리하기 때문에
  유지보수를 편리하게 만들어 줌
- 키와 값이 '='기호로 연결되어 있는 텍스트 파일로 ISO 8859-1 문자셋으로 저장되고,
  한글은 유니코드(Unicode)로 변환되어 저장

com.greedy.section03.map.run.Application2

Properties

- Properties 메소드

기능	메소드	리턴타입	설명
Properties 객체에 저장 및 가져오기	getProperty(String key)	String	Properties 인스턴스에 해당 key값에 해당하는 value값 리턴
	setProperty(String key, String value)	Object	Properties 객체에 해당 key값과 value값이 세트로 저장
파일 입출력	store(OutputStream out, String comments)	void	바이트 스트림으로 저장된 정보를 파일에 출력 저장
	store(Writer writer, String comments)	void	문자 스트림으로 저장된 정보를 출력 저장
	stroreToXML(OutputStream os, String comment)	void	저장된 정보를 바이트 스트림으로 xml로 출력 저장
	load(InputStream inStream)	void	바이트 스트림으로 저장된 파일의 내용을 읽어와서 Properties 인스턴스에 저장
	load(Reader reader)	void	문자 스트림으로 저장된 파일의 내용을 읽어와서 Properties 인스턴스에 저장
	loadFromXML(InputStream in)	void	바이트 스트림으로 저장된 xml 파일의 내용을 읽어와서 Properties 인스턴스에 저장

학습점검

- ✓ 컬렉션 프레임워크에 대해 이해할 수 있다.
- ✓ List계열(ArrayList, LinkedList)의 사용 목적을 이해할 수 있다.
- ✓ List계열을 이해하고 적용 할 수 있다.
- ✓ Comparable과 Comparator를 통해 정렬(오름차순, 내림차순)을 할 수 있다.
- ✓ Stack과 Queue의 자료구조에 대해 이해할 수 있다.
- ✓ Set계열(HashSet, LinkedHashSet, TreeSet)의 사용 목적을 이해할 수 있다.
- ✓ Set계열을 이해하고 적용 할 수 있다.
- ✓ Map계열(HashMap)의 사용 목적을 이해할 수 있다.
- ✓ Map계열을 이해하고 적용할 수 있다.
- ✓ EntrySet을 이해할 수 있다.
- ✓ Properties에 대해 이해하고 적용할 수 있다.