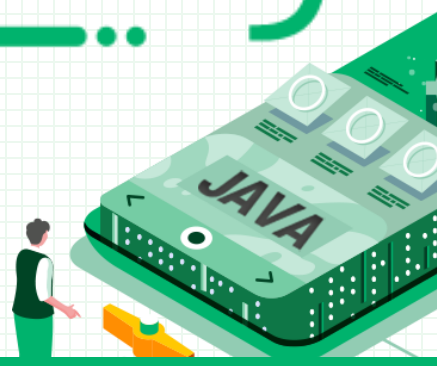
안드로이드 프로그래밍을 위한 자바기초



⑨ 컬렉션 프레임워크 활용하기

한국기술교육대학교
온라인평생교육원

학습목표

- 컬렉션 프레임워크를 이해하고 프로그래밍에 적용할 수 있다.
- List 인터페이스 관련 컬렉션 클래스에 대하여 이해하고 프로그래밍에 활용할 수 있다.
- Set/Map 인터페이스 관련 컬렉션 클래스에 대하여 이해하고 프로그래밍에 활용할 수 있다.

학습내용

- 컬렉션 프레임워크 이해하기
- List 컬렉션 활용하기
- Set/Map 컬렉션 활용하기

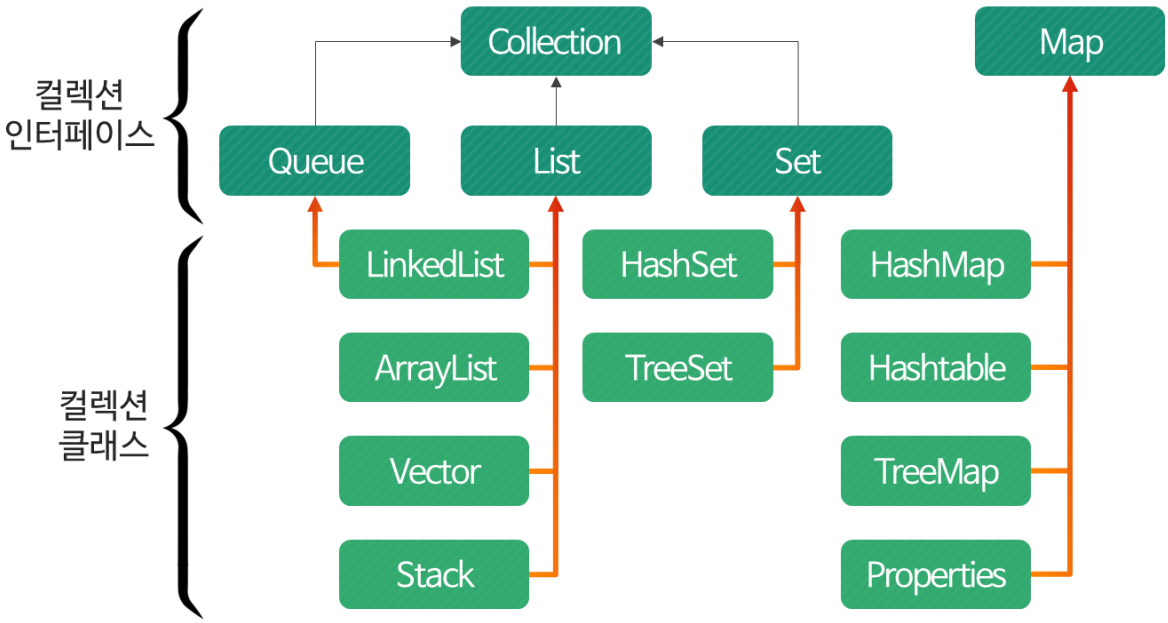
컬렉션 프레임워크 이해하기

컬렉션 프레임워크(framework) 개요

1) 컬렉션 프레임워크란?

- 자료구조를 효율적으로 사용하고 일관성 있게 접근하기 위한 틀
  - ✓ 컬렉션 인터페이스 – 자료구조의 추가, 수정, 삭제, 검색을 위한 인터페이스
  - ✓ 컬렉션 클래스 – 컬렉션 인터페이스를 구현한 클래스
- 패키지 : java.util

2) 컬렉션 프레임워크 구조



컬렉션 프레임워크 이해하기

Collection 인터페이스 개요

1) Collection 인터페이스란?

- 컬렉션 프레임워크의 최상위 인터페이스
- 객체를 추가, 삭제, 검색 등을 할 수 있는 일반적인 자료구조의 기능을 정의

2) Collection 인터페이스의 주요 메소드

기능	주요 메소드	설명
객체 추가	boolean add(E o)	컬렉션에 객체 추가
객체 삭제	boolean remove(Object o)	컬렉션에 객체 삭제
	void clear()	컬렉션의 모든 객체를 삭제
객체 검색	boolean contains(Object o)	컬렉션에 객체가 존재하는 여부
	int size()	컬렉션에 저장된 객체의 개수
	boolean isEmpty()	컬렉션이 비워져 있는지 여부 검사
	Iterator<E> iterator()	반복자 객체 반환
기타	Object[] toArray()	컬렉션에 있는 객체를 배열로 변환하여 반환

List 컬렉션 활용하기

List 인터페이스 이해하기

1) List 인터페이스란?

(1) Collection 인터페이스의 하위 인터페이스

- ✓ Collection 인터페이스의 모든 메소드를 상속함

(2) 객체의 추가, 삭제, 검색, 수정이 가능
(3) 순서가 있는 자료 구조로 배열과 같이 인덱스로 관리
(4) 동일한 객체의 중복 저장 가능
(5) List 인터페이스를 구현한 컬렉션 클래스

2) List인터페이스의 주요 메소드

기능	주요 메소드	설명
객체 검색	ListIterator<E> listIterator()	List 반복자 객체 반환
	E get(int index)	index 위치의 객체 반환
	int indexOf(Object o)	객체의 index 값을 반환
객체 수정	E set(int index, E element)	index 위치의 객체를 주어진 객체로 변경

List 컬렉션 활용하기

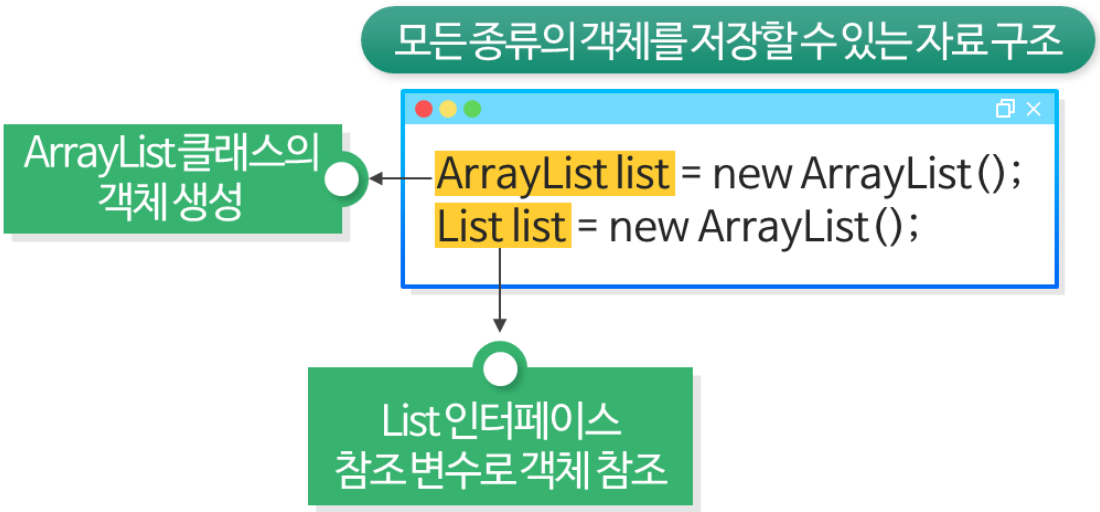
ArrayList 클래스 활용하기

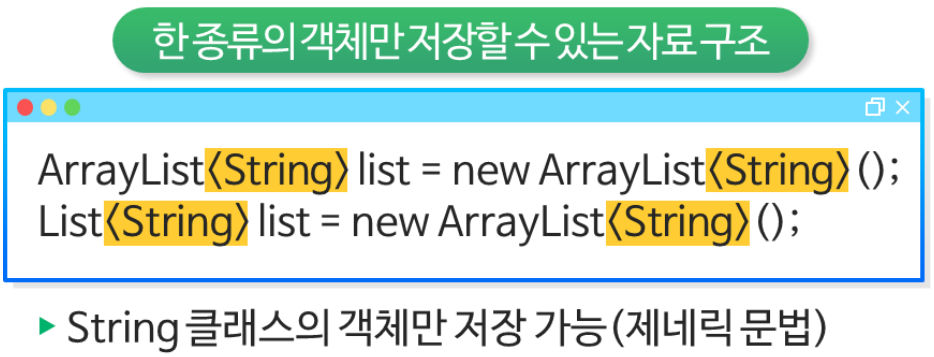
1) ArrayList 클래스란?

- List 인터페이스를 구현한 대표적인 컬렉션 클래스

2) ArrayList 객체 생성

- 모든 종류의 객체를 저장할 수 있는 자료 구조
- 한 종류의 객체만 저장할 수 있는 자료 구조



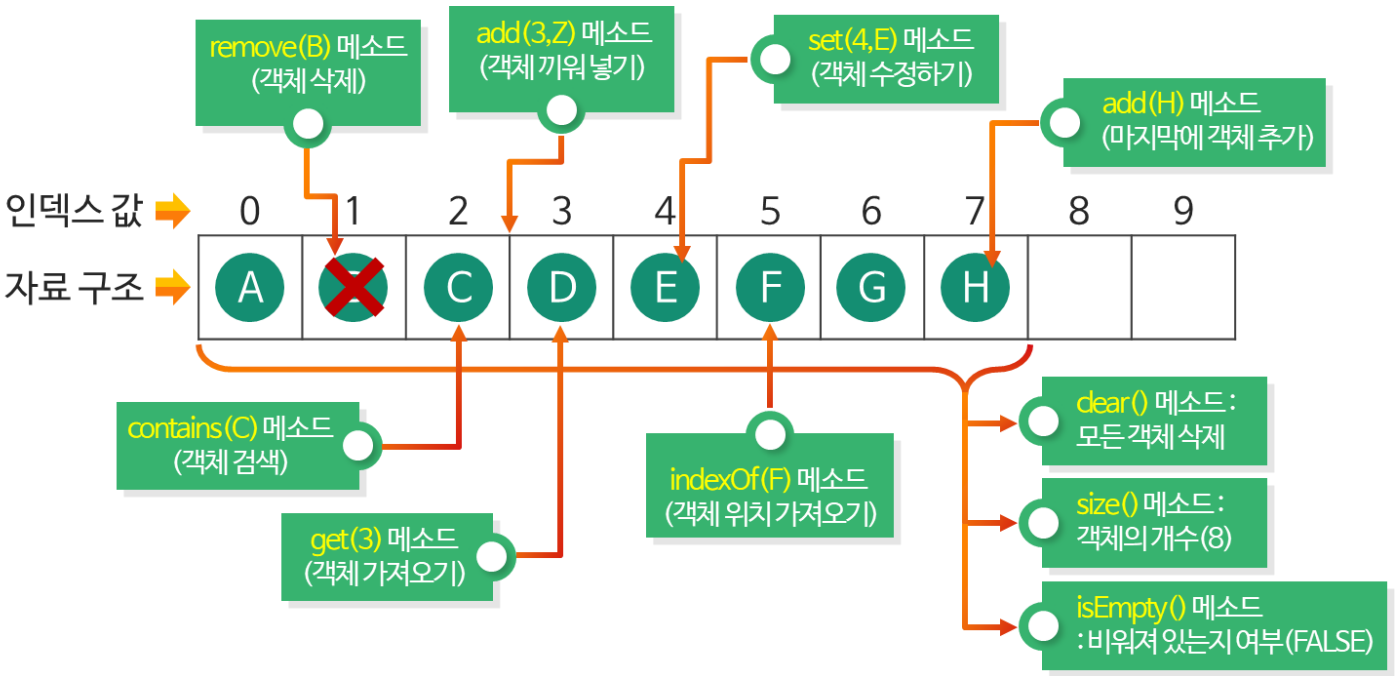


▸ String 클래스의 객체만 저장 가능(제네릭 문법)

ArrayList 클래스 활용하기

List 컬렉션 활용하기

3) ArrayList 클래스 메소드



4) 반복자 인터페이스

- 컬레션에 저장된 객체를 순차적으로 접근하기 위한 인터페이스
- 컬렉션 클래스에 구현되어 있음
- 종류

▸ List 인터페이스를 구현한 클래스만 사용 가능

List 컬렉션 활용하기

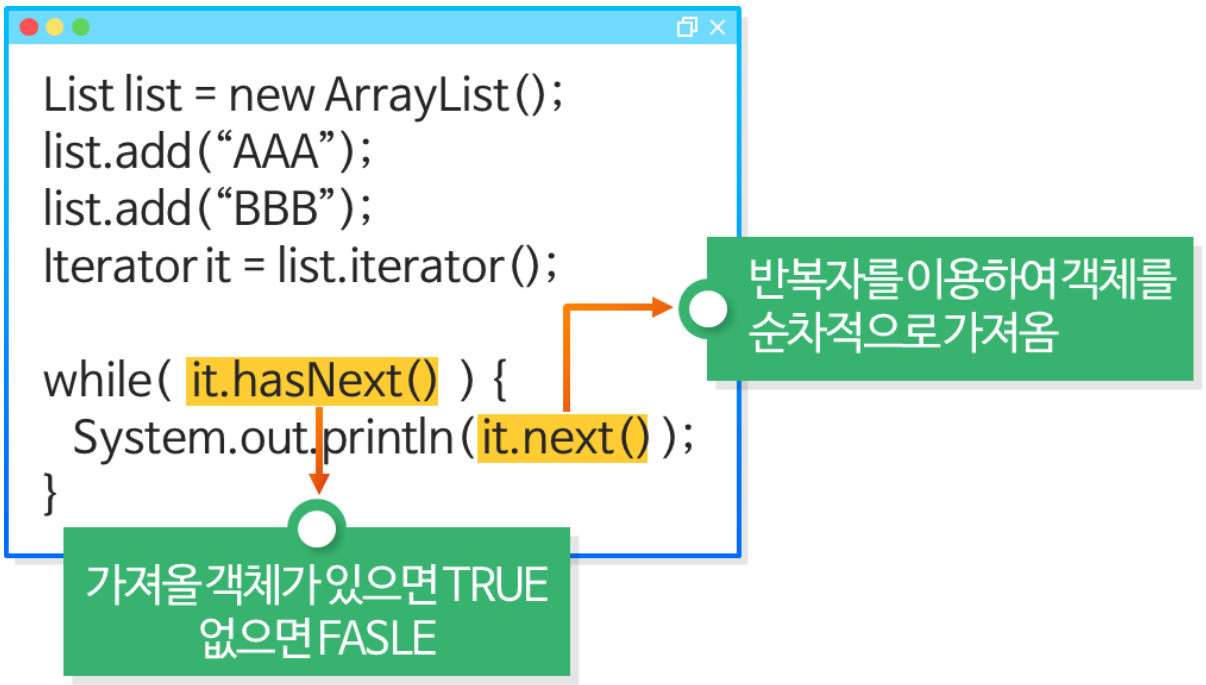
- 주요 메소드

인터페이스	주요 메소드	설명
Enumeration	boolean hasMoreElements()	컬렉션에서 읽어올 다음 객체가 있는지 여부
	E nextElement()	다음 객체를 반환
Iterator	boolean hasNext()	컬렉션에서 읽어올 다음 객체가 있는지 여부
	E next()	다음 객체를 반환
	void remove()	객체 삭제

인터페이스	주요 메소드	설명
ListIterator	boolean hasNext()	컬렉션에서 읽어올 다음 객체가 있는지 여부
	E next()	다음 객체를 반환
	void remove()	객체 삭제
	void add(E e)	객체 추가
	void set(E e)	객체 변경
	boolean hasPrevious()	컬렉션에 읽어올 이전 객체가 있는지 여부
	E previous()	이전 객체를 반환

List 컬렉션 활용하기

- 반복자 인터페이스 활용하기



- ✓ ArrayList 클래스의 객체를 생성하여 List 인터페이스의 참조 변수로 참조함
- ✓ 컬렉션에 2개의 문자열 객체 추가
- ✓ list 참조 변수의 iterator() 메소드를 이용하여 Iterator 반복자 객체 생성

실습

ArrayList와 반복자 실습

실행 화면

```
[모든 객체를 저장할 수 있는 ArrayList]
< 반복자 없이 출력 >
AAAA
123
100.45
[ArrayListClass] x =10
< Iterator 반복자 사용 >
AAAA
123
100.45
[ArrayListClass] x=10
[String 객체만 저장할 수 있는 ArrayList]
[ AA BB DD EE ] …
```

- 소스 파일명 : [ArrayListClass.java]
- 자세한 내용은 실습 영상을 확인해보세요.

List 컬렉션 활용하기

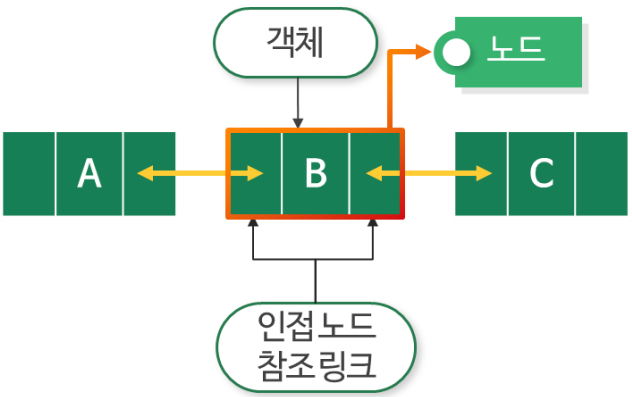
LinkedList 클래스 활용하기

1) LinkedList 클래스란?

- List 인터페이스와 Queue 인터페이스를 구현
- LinkedList 자료 구조와 Queue 자료 구조 사용

2) LinkedList 자료 구조

(1) 인접한 노드와 참조링크를 이용한 체인과 같은 구조
(2) ArrayList 클래스와 동일하게 사용



3) Queue 자료 구조

(1) FIFO(First In First Out) 구조

✓ 먼저 추가된 객체가 먼저 나오는 파이프와 같은 구조

(2) 주요 메소드

✓ offer() 메소드 : 객체를 추가
✓ peek() 메소드 : 처음 추가된 객체를 가져옴(가져온 객체는 삭제되지 않음)
✓ poll() 메소드 : 처음 추가된 객체를 가져옴(가져온 객체는 삭제됨)

실습

LinkedList 클래스 실습

실행 화면

```
[LinkedList 자료 구현]
[ AAA BBB CCC DDD ]
add(2,EEE) : [ AAA BBB EEE CCC DDD ]
set(1,ZZZ) : [ AAA ZZZ EEE CCC DDD ]
remove(ZZZ) : [ AAA EEE CCC DDD ]
contains(EEE) : true
get(3) : DDD
indexOf(CCC) : 2
size() : 4
isEmpty() : false
[Queue 자료 구조 구현]
[ AAA BBB CCC DDD ]
peek() : AAA…
```

- 소스 파일명 : [LinkedListClass.java]
- 자세한 내용은 실습 영상을 확인해보세요.

List 컬렉션 활용하기

Vector 클래스 활용하기

1) Vector 클래스란?

- ArrayList와 동일한 구조와 사용 방법
- 다양한 메소드 제공

2) Vector 클래스의 저장 용량

(1) 기본용량 : 10개
(2) 주요 메소드

구분	주요 메소드	설명
생성자 메소드	Vector()	기본 용량 10개 생성
	Vector(int size)	size 만큼 용량 생성
	Vector(int size, int inc)	size 만큼 용량 생성, inc 만큼 용량 증가
멤버 메소드	int capacity()	용량 반환
	int size()	저장된 객체의 개수 반환
	void setSize(int size)	size 크기로 용량 설정
	void trimToSize()	저장된 객체의 개수만큼 용량을 설정

List 컬렉션 활용하기

Stack 클래스 활용하기

1) Stack 클래스란?

- Vector 클래스의 하위 클래스

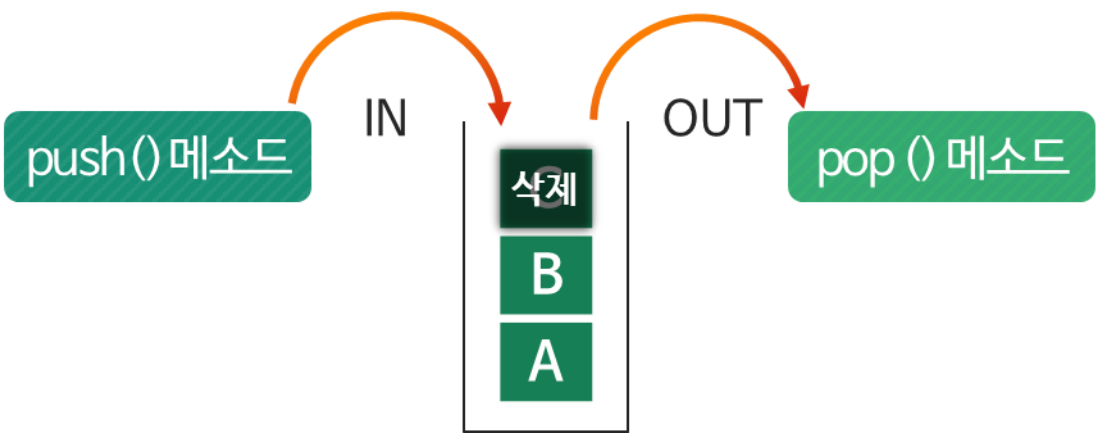
2) Stack 자료 구조

(1) LIFO(Last In First Out) 구조

✓ 마지막에 추가된 객체가 먼저 나오는 항아리와 같은 구조

(2) 주요 메소드

✓ push() 메소드 : 객체를 추가
✓ peek() 메소드 : 마지막에 추가된 객체를 가져옴(가져온 객체는 삭제되지 않음)
✓ pop() 메소드 : 마지막에 추가된 객체를 가져옴(가져온 객체는 삭제됨)



실습

Vector와 Stack 클래스 실습

실행 화면

```
[Vector 자료 구현]
[ AAA BBB CCC DDD ]
add(2,EEE) : [ AAA BBB EEE CCC DDD ]
set(1,ZZZ) : [ AAA ZZZ EEE CCC DDD ]
remove(ZZZ) : [ AAA EEE CCC DDD ]
contains(EEE) : true
get(3) : DDD
indexOf(CCC) : 2
isEmpty() : false
[Vector 용량]
v1 : capacity() : 10
v1 : size() : 4
v1 : trimToSize() : capacity() : 4…
```

- 소스 파일명 : [VectorStack.java]
- 자세한 내용은 실습 영상을 확인해보세요.

Set/Map 컬렉션 활용하기

Set/HashSet 이해하기

1) Set 인터페이스란?

(1) Collection 인터페이스의 하위 인터페이스

✓ Collection 인터페이스의 모든 메소드를 상속함

(2) 객체의 추가, 삭제, 검색, 수정이 가능
(3) 순서가 없는 자료 구조
(4) 동일한 객체의 중복 저장 불가
(5) 주요 메소드 : List 컬렉션과 동일

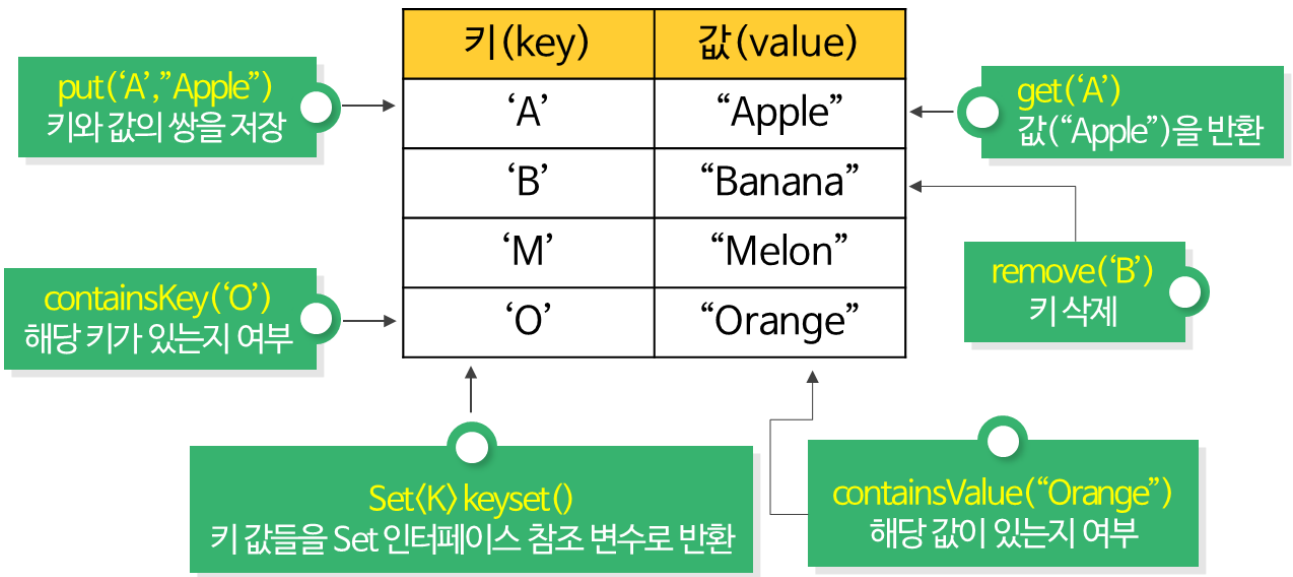
- HashSet 클래스 – Set 인터페이스를 구현한 컬렉션 클래스

Map/HashMap 이해하기

1) Map 인터페이스란?

(1) 키(key)와 값(value)의 쌍으로 이루어진 구조를 정의
(2) 순서가 없는 자료 구조
(3) 키(Key)는 중복 불가
(4) 값(Value)은 중복 가능

- HashMap 클래스 – Map 인터페이스를 구현한 컬렉션 클래스



실습

HashSet과 HashMap 클래스 실습

실행 화면

```
[HashSet 클래스 실습]
size() : 3
Java
Program
Eclipse
remove(Eclipse), size() : 2
Java
Program
contains(Java) : true
isEmpty() : false
[HashMap 클래스 실습]
size() : 4
K=A V=Apple…
```

- 소스 파일명 : [SetMap.java]
- 자세한 내용은 실습 영상을 확인해보세요.

정리하기

▪ 컬렉션 프레임 워크 이해하기

- 컬렉션 프레임워크(framework)
  - 자료구조를 효율적으로 사용하고, 일관성 있게 접근하기 위한 틀
  - 컬렉션 인터페이스 : 자료구조의 추가, 수정, 삭제, 검색을 위한 인터페이스
  - 컬렉션 클래스 : 컬렉션 인터페이스를 구현한 클래스
  - 패키지 : java.util

- Collection 인터페이스
  - 컬렉션 프레임워크의 최상위 인터페이스
  - 객체를 추가, 삭제, 검색 등을 할 수 있는 일반적인 자료구조의 기능을 정의

정리하기

▪ 기초 클래스 활용하기

- List 인터페이스
  - Collection 인터페이스의 하위 인터페이스
  - 객체의 추가, 삭제, 검색, 수정이 가능
  - 순서가 있는 자료 구조로 배열과 같이 인덱스로 관리
  - 동일한 객체의 중복 저장 가능
  - List 인터페이스를 구현한 컬렉션 클래스 : ArrayList, LinkedList, Vector, Stack

- ArrayList 클래스
  - List 인터페이스를 구현한 대표적인 컬렉션 클래스
  - ArrayList 객체 생성
    - ➢ 모든 종류의 객체를 저장할 수 있는 자료 구조
    - ➢ 한 종류의 객체만 저장할 수 있는 자료 구조

정리하기

▪ 기초 클래스 활용하기

- LinkedList 클래스

- List 인터페이스와 Queue 인터페이스를 구현
- LinkedList 자료 구조
  - ➢ 인접한 노드와의 참조 링크를 이용한 체인과 같은 구조
  - ➢ ArrayList 클래스와 동일하게 사용
- Queue 자료 구조
  - ➢ FIFO(First In First Out) 구조 : 먼저 추가된 객체가 먼저 나오는 파이프와 같은 구조

- Stack 클래스

- Vector 클래스의 하위 클래스
- Stack 자료 구조
  - ➢ LIFO(Last In First Out) 구조 : 마지막에 추가된 객체가 먼저 나오는 항아리와 같은 구조

정리하기

▪ Set/Map 컬렉션 활용하기

- Set 인터페이스

- Collection 인터페이스의 하위 인터페이스
- 객체의 추가, 삭제, 검색, 수정이 가능
- 순서가 없는 자료 구조
- 동일한 객체의 중복 저장 불가
- 주요 메소드 : List 컬렉션과 동일
- HashSet 클래스 : Set 인터페이스를 구현한 컬렉션 클래스

- Map 인터페이스

- 키(key)와 값(value)의 쌍으로 이루어진 구조를 정의
- 순서가 없는 자료 구조
- 키(key)는 중복 불가, 값(Value)은 중복 가능
- HashMap 클래스 : Map 인터페이스를 구현한 컬렉션 클래스