컬렉션 프레임워크

컬렉션 프레임워크

- 컬렉션 프레임워크(Collection Framework)
  - ✓ 여러 데이터를 쉽고 효과적으로 처리할 수 있도록 제공하는 클래스
  - ✓ 자료 구조(Data Structure)와 알고리즘(Algorithm)을 구조화한 클래스
  - ✓ 자바 인터페이스(Interface)를 구현한 형태로 작성된 클래스

- 주요 컬렉션 프레임워크 인터페이스
  - ✓ List 인터페이스
  - ✓ Set 인터페이스
  - ✓ Map 인터페이스

컬렉션 프레임워크 인터페이스

List 인터페이스

- 목록 관리
- 순서 있는 목록
- 배열 대신 사용
- 요소 추가/삭제/수정
- 요소의 중복 저장 가능

Set 인터페이스

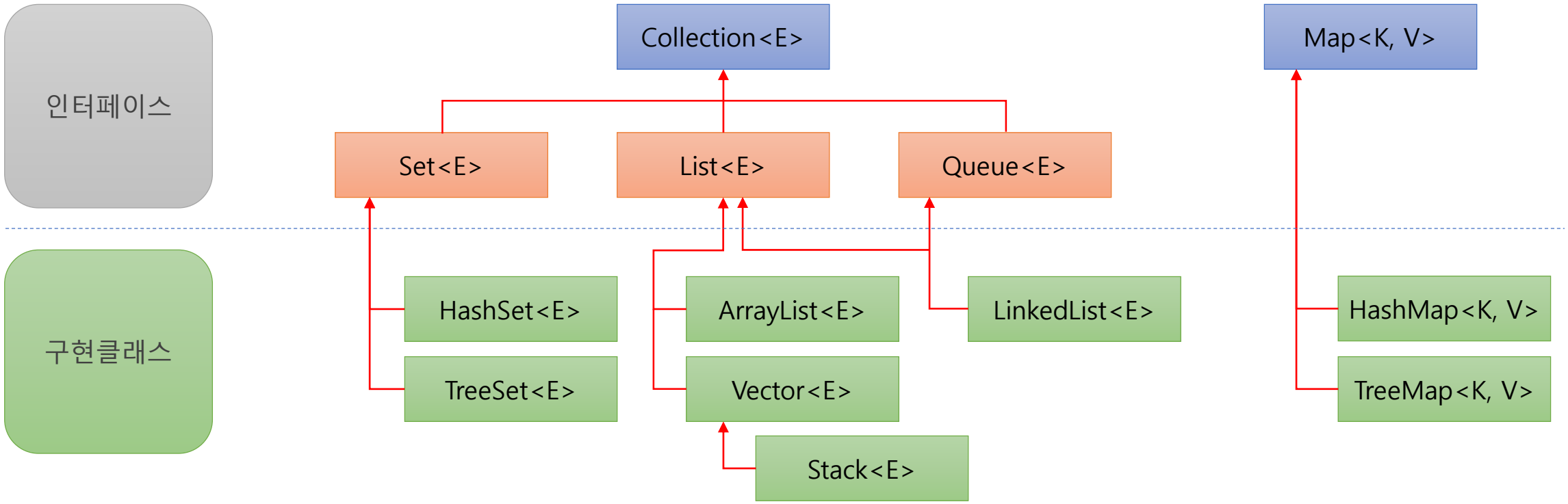
- 집합 관리
- 순서 없는 목록
- 교집합/합집합/차집합 등
- 요소 추가/삭제/수정
- 요소의 중복 저장 불가능

Map 인터페이스

- 데이터를 쌍으로 관리
- Key : 데이터 식별 수단
- Value : 데이터
- 요소 추가/삭제/수정
- Key는 중복 저장 불가능
  Value는 중복 저장 가능

Java

컬렉션 프레임워크의 구조



제네릭

- 모든 컬렉션 프레임워크는 제네릭(generic) 기반으로 구현된 클래스
  - ✓ 어떤 컬렉션 프레임워크를 생성할 때 어떤 타입을 저장할 것인지 구체적으로 명시하는 것
  - ✓ 제네릭으로 지정 가능한 타입은 오직 참조 타입(Reference Type)만 가능
  - ✓ 기본 타입(Primitive Type)을 저장하려면 Wrapper Class 타입으로 처리

- 문자열(String) 저장을 위한 ArrayList

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

JDK 1.7 이후
new ArrayList<>(); 가능

- 정수(Integer) 저장을 위한 HashSet

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

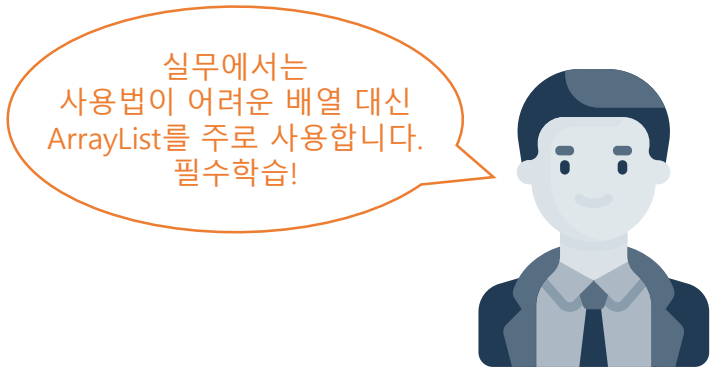
JDK 1.7 이후
new HashSet<>(); 가능

Java

ArrayList<E>

- 패키지 : java.util
- 배열(Array)을 리스트로 구현한 클래스
- 배열보다 쉬운 사용법을 제공

- ArrayList 주요 기능
  - ✓ 리스트 길이 자동 세팅
  - ✓ 요소의 추가/수정/삭제
  - ✓ 인덱스 기반의 접근
  - ✓ 특정 요소 조회
  - ✓ 리스트와 배열의 상호 변환



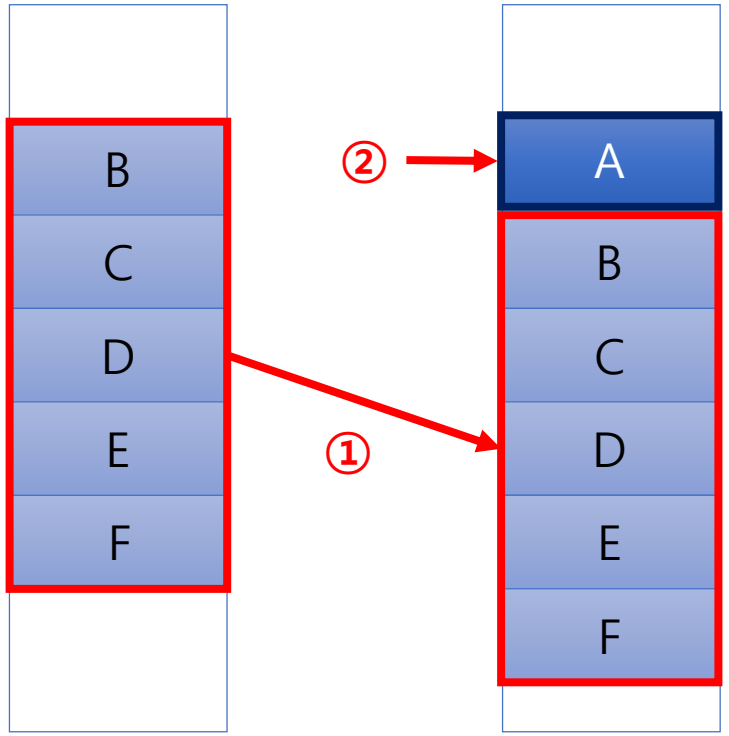
ArrayList<E> 주요 메소드

메소드	역할
boolean add(E element)	마지막에 element 추가. 성공하면 true 반환
void add(int index, E element)	index 위치에 element 추가
boolean addAll(Collection<? extends E> c)	마지막에 컬렉션 c의 모든 요소 추가. 성공하면 true 반환
void clear()	모든 요소 제거
boolean contains(Object o)	객체 o를 포함하고 있으면 true 반환
E get(int index)	index 위치의 요소 반환
E set(int index, E element)	index 위치의 기존 요소를 element로 변경. 변경된 요소 반환
boolean isEmpty()	비어 있으면 true 반환
E remove(int index)	index 위치의 element 제거. 제거된 요소 반환
boolean remove(Object o)	객체 o와 동일한 요소 제거. 성공하면 true 반환
int size()	요소들의 개수 반환
Object[] toArray()	리스트의 모든 요소들을 포함하는 Object 타입의 배열 반환

ArrayList<E> vs LinkedList<E>

- ArrayList(배열 리스트)
  - ✓ 연속된 메모리 할당
  - ✓ 빠른 참조
  - ✓ 느린 추가/삭제

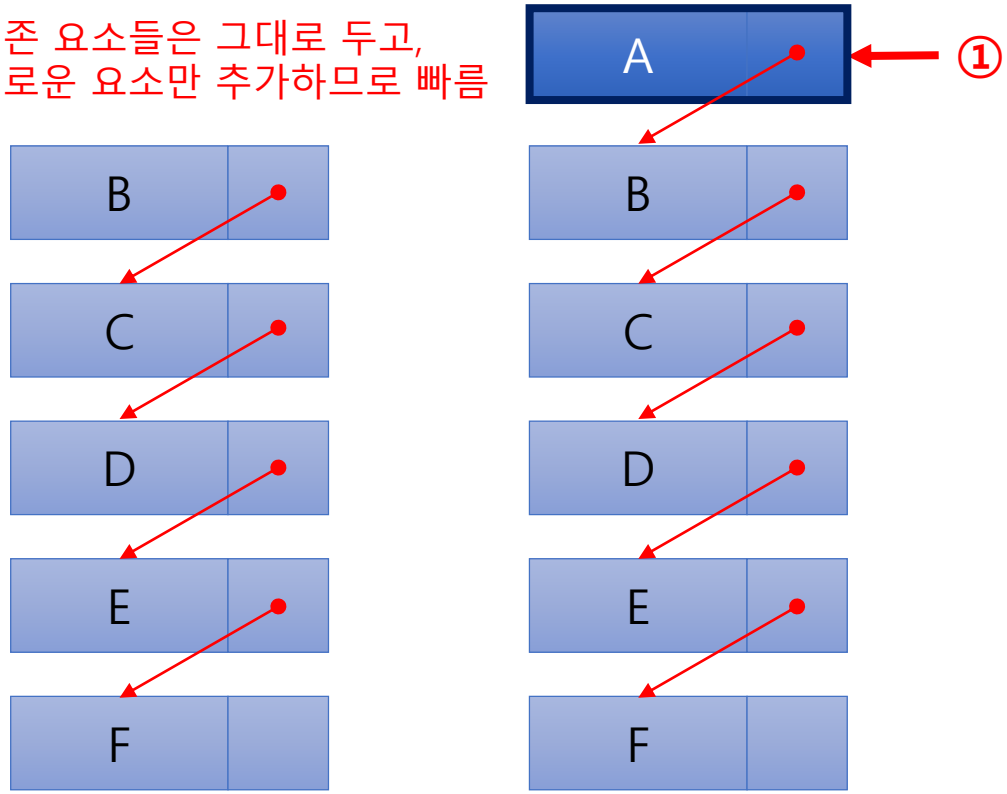
기존 요소들을 모두 옮긴 뒤
새로운 요소를 추가하므로 느림



추가 동작 비교

- LinkedList(연결 리스트)
  - ✓ 비연속적 메모리 할당
  - ✓ 느린 참조
  - ✓ 빠른 추가/삭제

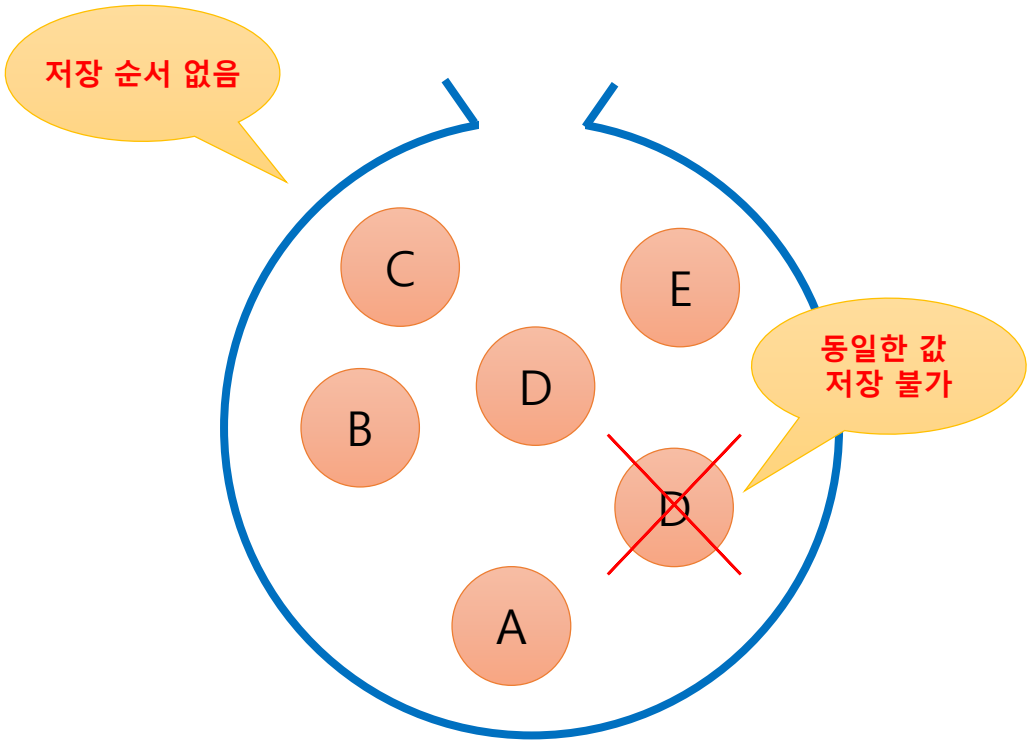
기존 요소들은 그대로 두고,
새로운 요소만 추가하므로 빠름



HashSet<E>

- 패키지 : java.util
- 집합을 관리하기 위한 자료 구조
- 중복된 요소가 저장되지 않는 특성을 가짐
- 인덱스가 없어 순서대로 저장되지 않는 특성을 가짐
- 해시(Hash)를 이용하기 때문에 삽입, 삭제, 검색이 빠름

- HashSet 주요 기능
  - ✓ 세트 길이 자동 세팅
  - ✓ 요소의 추가/수정/삭제
  - ✓ 리스트와 세트의 상호 변환
  - ✓ 교집합, 합집합, 차집합, 부분집합 등 집합 연산



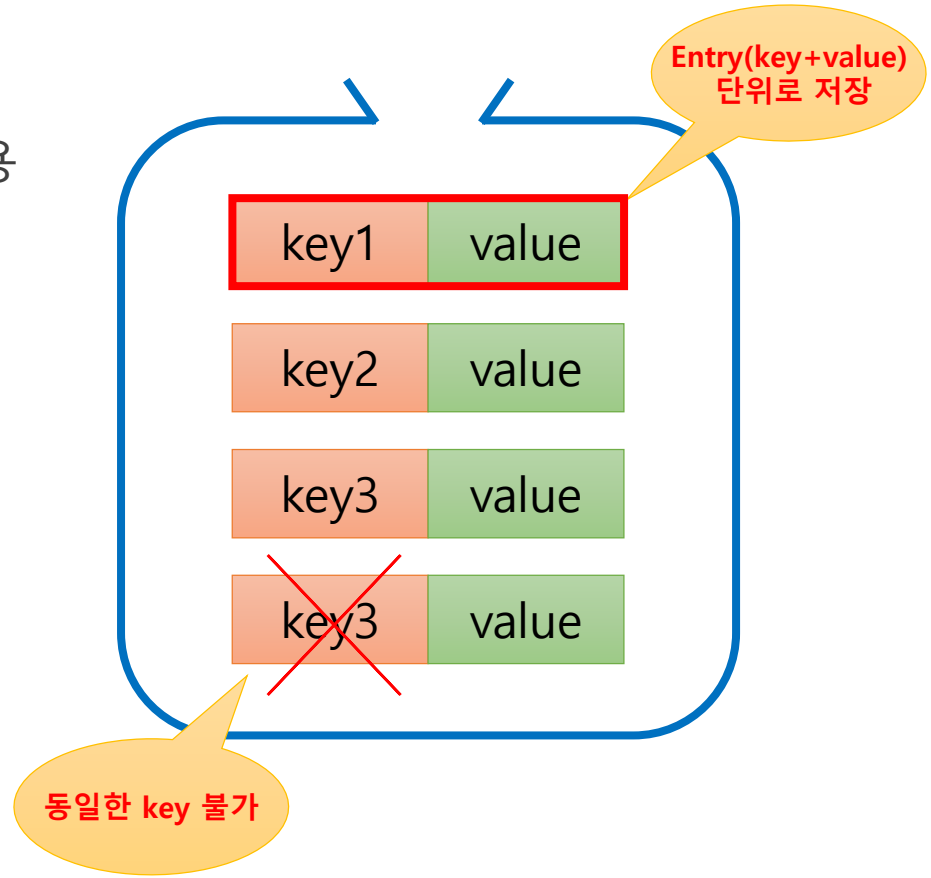
HashSet<E> 주요 메소드

메소드	역할
boolean add(E element)	element 추가. 성공하면 true 반환
boolean addAll(Collection<? extends E> c)	컬렉션 c의 모든 요소 추가. 성공하면 true 반환
void clear()	모든 요소 제거
boolean contains(Object o)	객체 o를 포함하고 있으면 true 반환
boolean isEmpty()	비어 있으면 true 반환
boolean remove(Object o)	객체 o와 동일한 요소 제거. 성공하면 true 반환
boolean removeAll(Collection<? extends E> c)	컬렉션 c와 동일한 모든 요소 제거. 성공하면 true 반환
int size()	요소들의 개수 반환
Object[] toArray()	세트의 모든 요소들을 포함하는 Object 타입의 배열 반환

HashMap<K, V>

- 패키지 : java.util
- 키(Key)와 값(Value)으로 하나의 Entry가 구성되는 자료 구조
- 저장하려는 데이터는 값이고, 값을 알아내기 위해서 키를 사용
- 값(Value)은 중복이 가능하지만 키(Key)는 중복이 불가능
- 해시(Hash)를 이용하기 때문에 삽입, 삭제, 검색이 빠름

- HashMap 주요 기능
  - ✓ 요소의 삽입/삭제/수정
  - ✓ 일치하는 키(Key) 검색
  - ✓ 일치하는 값(Value) 검색

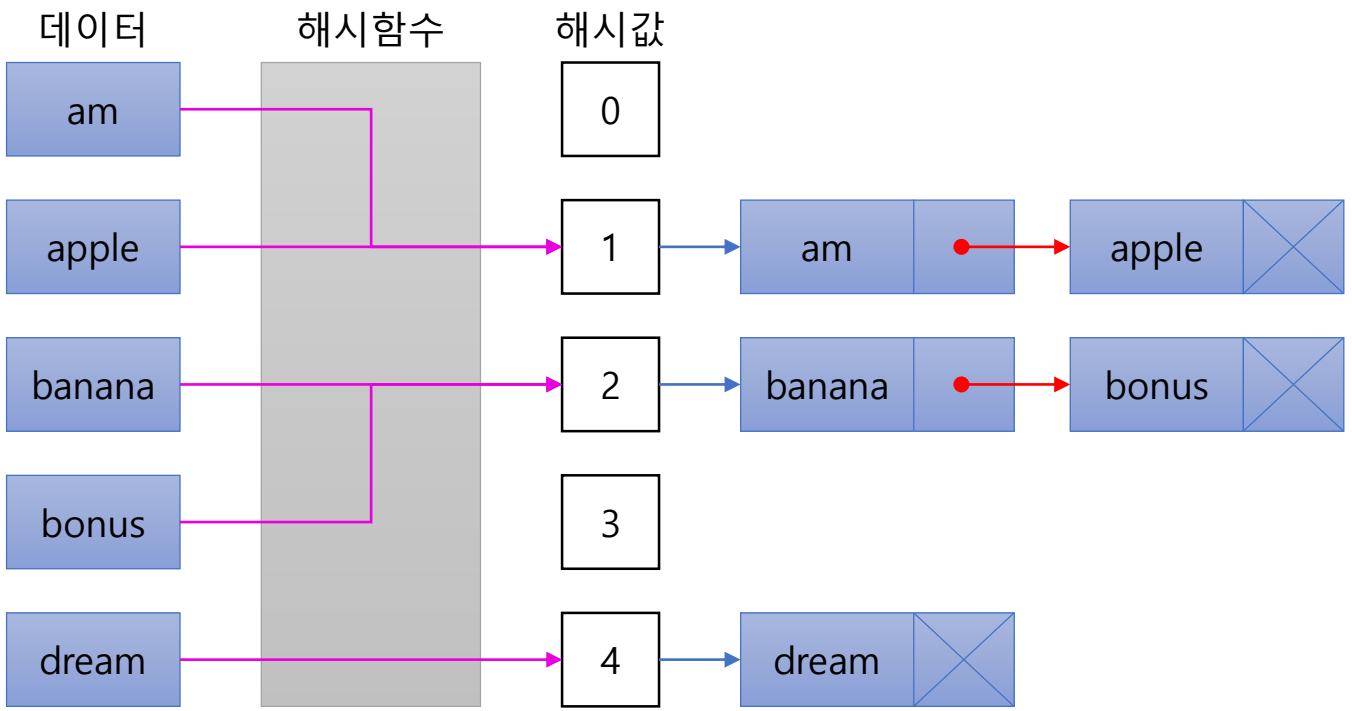


HashMap<K, V> 주요 메소드

메소드	역할
void clear()	모든 Entry 삭제
boolean containsKey(Object key)	전달된 key를 포함하고 있으면 true 반환
boolean containsValue(Object value)	전달된 value를 포함하고 있으면 true 반환
Set<Map.Entry<K, V>> entrySet()	해시맵의 모든 Entry를 저장한 Set 반환
V get(Object key)	전달된 key의 값(value) 반환, 없으면 null 반환
boolean isEmpty()	해시맵이 비어 있으면 true 반환
Set<V> keySet()	해시맵의 모든 key를 저장한 Set 반환
V put(K key, V value)	key와 value를 하나의 Entry로 해시맵에 저장하고 저장한 value를 반환
V remove(Object key)	전달된 key를 가진 Entry를 해시맵에서 삭제하고 삭제한 value를 반환
int size()	전체 Entry 개수 반환

Hash Algorithm

- 해시(Hash)
  - ✓ 다양한 길이를 가진 데이터를 고정된 길이를 가지는 데이터로 매핑(Mapping)한 값
  - ✓ 어떤 데이터를 저장할 때 해당 데이터의 해시값을 계산해서 인덱스(Index)로 사용함
  - ✓ 어떤 데이터를 검색할 때 해당 데이터의 해시값을 인덱스(Index)로 사용하면 되기 때문에 빠른 조회가 가능

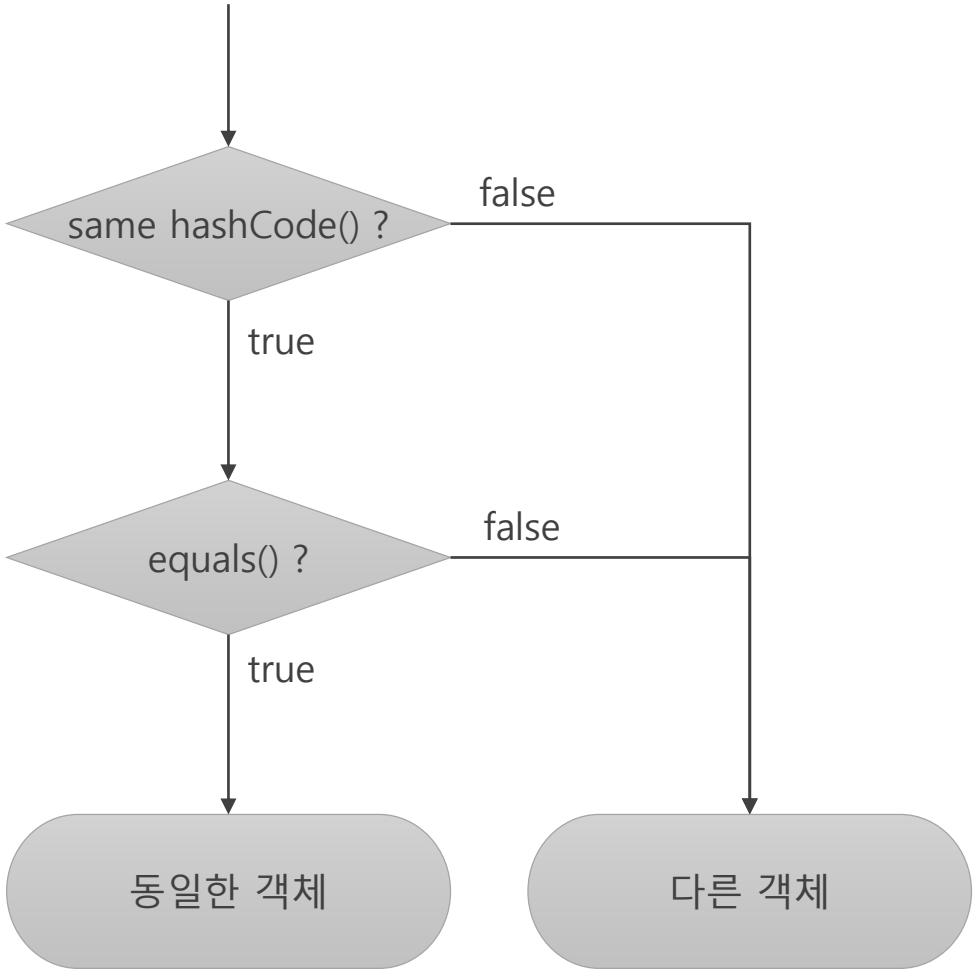


동일한 해시값을 가지는 경우
각 데이터를 연결 리스트로
연결하는 방식을
개별 체이닝(Sepatate Chaining) 방식
이라고 합니다.
자바의 처리 방식입니다.

Hash 기반 컬렉션

- 자바의 해시(Hash)
  - ✓ 모든 객체는 Object 클래스의 hashCode() 메소드가 반환하는 값을 해시값으로 사용
  - ✓ Object 클래스의 hashCode() 메소드는 객체의 참조값을 반환하기 때문에 모든 객체는 서로 다른 해시값을 가짐
  - ✓ 필드값이 동일한 객체도 서로 다른 해시값을 가지므로 서로 다른 객체로 인식함
  - ✓ 필드값이 동일한 경우 동일한 객체로 인식시키기 위해서는 hashCode() 메소드를 목적에 맞게 오버라이드 해야 함

- 해시(Hash) 기반 컬렉션
  - HashSet, HashMap, HashTable 등
  - 요소들의 중복 저장 방지하기 위해서 hashCode() 메소드와 equals() 메소드를 사용함
  - 우선 hashCode() 메소드를 비교해서 해시값이 같으면 동일한 객체일 가능성이 있는 것으로 판단
  - 동일한 해시값을 가지는 경우 equals() 메소드를 호출해서 동일한 객체인지 최종 판단
  - 다른 해시값을 가지는 경우 동일한 객체일 가능성이 없으므로 equals() 메소드를 호출하지 않음



Collections

- 패키지 : java.util
- 컬렉션을 대상으로 특정 연산을 수행할 수 있는 클래스
- 모든 메소드가 클래스 메소드(static)로 되어 있음

- 주요 메소드

메소드	역할
boolean addAll(Collection<? super T> c, T... elements)	모든 elements를 컬렉션 c에 저장
int binarySearch(List<? extends Comparable<? super T>> list, T key)	이진검색으로 key값을 검색
T max(Collection<? extends T> coll)	컬렉션 coll에서 최대값 반환
T min(Collection<? extends T> coll)	컬렉션 coll에서 최소값 반환
void reverse(List<?> list)	리스트 list를 내림차순 정렬
void sort(List<T> list)	리스트 list를 오름차순 정렬