15-2장. 컬렉션(Set, Map)

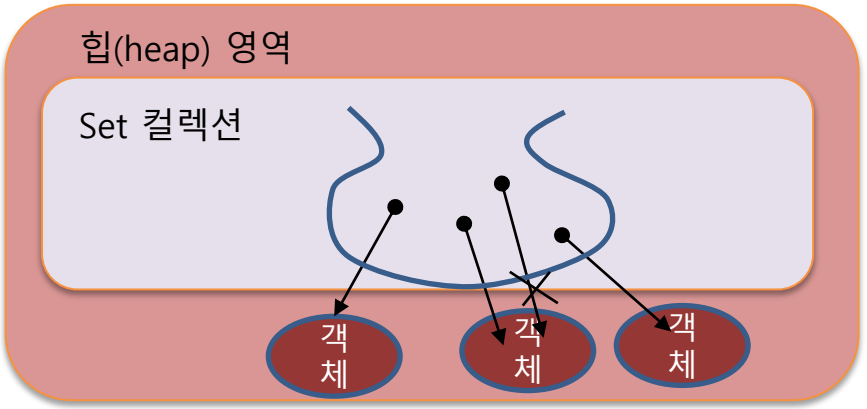
Set, Map

컬렉션 프레임워크

◆ Set 인터페이스

- 특징
  - 수학의 집합에 비유될 수 있다.
  - 저장 순서가 유지되지 않는다.
  - 객체를 중복 저장할수 없다.
- 구현 클래스

  HashSet, LinkedHashSet, TreeSet



컬렉션 프레임워크

◆ Set 인터페이스

● 주요 메소드

기능	메소드	설명
객체 추가	add(element)	주어진 객체를 저장
객체 검색	contains(object)	주어진 객체가 저장되어 있는지 여부
	isEmpty()	컬렉션이 비어 있는지 조사
	iterator()	저장된 객체를 한 번씩 가져오는 반복자 리턴
	size()	저장되어 있는 전체 객체 수를 리턴
객체 삭제	clear()	저장된 모든 객체 삭제
	remove(object)	주어진 객체를 삭제

Set 인터페이스

- 객체 추가, 찾기, 삭제

```
Set<String> set =...;
    set.add("홍길동");
    set.add("임꺽정");
    set.remove("홍길동")
```

- Set컬렉션은 인덱스로 객체를 검색해서 가져오는 메소드가 없다.
  대신, 전체 객체를 대상으로 한번씩 반복해서 가져오는 **반복자(iterator)**를 제공한다.

```
Iterator<String> iterator = set.iterator();
while(iterator.hasNext()) {
    String element = iterator.next();
}
```

Set 인터페이스

◆ Collection 요소를 순회하는 Iterator

- 순서가 없는 Set 인터페이스를 구현한 경우에는 **get(i)** 메서드를 사용할 수 없다.

  이때 **Iterator 클래스**의 **iterator() 메서드**를 호출하여 참조한다.

리턴 타입	메소드명	설명
Boolean	hasNext()	가져올 객체가 있으면 true, 없으면 false 리턴
E	next()	컬렉션에서 하나의 객체를 가져온다.
void	remove()	Set 컬렉션에서 객체를 제거한다.

Set 인터페이스

```java
//자료형이 String 타입인 Set 객체 생성
Set<String> set = new HashSet<>();

//자료 추가
set.add("감");
set.add("귤");
set.add("사과");
set.add("포도");
set.add("사과");

//자료의 개수 - 중복은 제외
System.out.println("총 객체수 : " + set.size());

//자료 출력 - 배열 형태로 출력
System.out.println(set);

//자료 출력 - 값 형태
Iterator<String> ir = set.iterator();
while(ir.hasNext()) { //반복하면서 자료가 있으면
    String fruit = ir.next(); //다음 자료를 1개씩 가져옴
    System.out.println(fruit);
}

//자료 삭제
set.remove("귤");

//자료의 개수
System.out.println("총 객체수 : " + set.size());
```

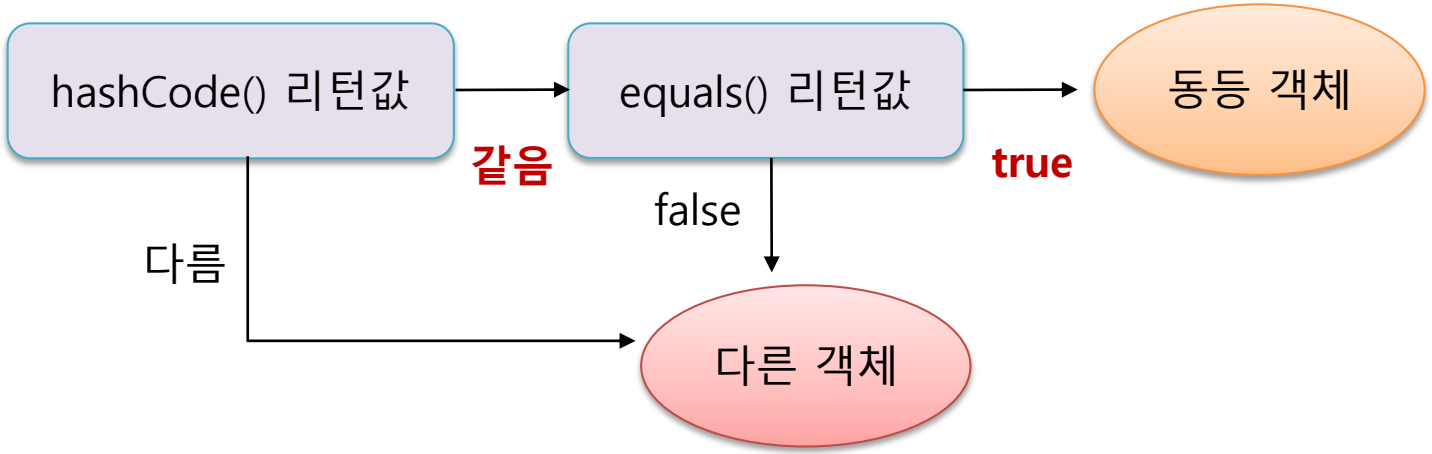
중복이 허용되지 않음.<String>클래스에 객체가 동일한 경우에 대한 처리방법이 이미 구현됨.
순서도 없음

```
총 객체수 : 4
[감, 포도, 귤, 사과]
감
포도
귤
사과
총 객체수 : 3
```

Set 인터페이스

◆ HashSet 클래스

- HashSet은 Set 인터페이스의 구현 클래스이다.
- 특징
  - 동일 객체 및 동등 객체는 중복 저장하지 않는다.
  - 순서없이 저장
  - 동등 객체 판단 방법



Set 인터페이스

◆ HashSet을 활용한 회원관리 프로그램

```
package collection.set;

public class Member {
    private int memberId;        //회원 아이디
    private String memberName;   //회원 이름

    public Member(int memberId, String memberName) {
        this.memberId = memberId;
        this.memberName = memberName;
    }

    public int getMemberId() {
        return memberId;
    }

    public void setMemberId(int memberId) {
        this.memberId = memberId;
    }

    public String getMemberName() {
        return memberName;
    }

    public void setMemberName(String memberName) {
        this.memberName = memberName;
    }
```

Set 인터페이스

◆ Member 클래스

```
@Override
public String toString() {
    return memberName + "회원님의 아이디는 " + memberId + "입니다.";
}

@Override
public boolean equals(Object obj) {
    if(obj instanceof Member) {  //Member의 인스턴스 객체가 obj라면
        Member member = (Member)obj; //다운 캐스팅 - Member로 형 변환
        if(this.memberId == member.memberId) //외부에서 입력한 memberId와 같다면
            return true;
    }
    return false;
}

@Override
public int hashCode() { //equals와 hashCode 모두 재정의되어야 중복이 허용 안됨
    return memberId;
}
```

Set 인터페이스

```
public class MemberHashSet {
    //자료형이 Member인 HashSet 멤버 변수 선언
    private HashSet<Member> hashSet;

    public MemberHashSet() {
        hashSet = new HashSet<>();
    }

    //회원 추가
    public void addMember(Member member) {
        hashSet.add(member);
    }

    //회원 목록 출력
    public void showAllMember() {
        /*for(Member member: hashSet) {
            System.out.println(member);
        }*/
        Iterator<Member> ir = hashSet.iterator();
        while(ir.hasNext()) {
            Member member = ir.next();
            System.out.println(member);
        }
    }
}
```

Set 인터페이스

◆ MemberHashSet 클래스

```
//회원 삭제
public boolean removeMember(int memberId) {//매개변수는 memberId
    Iterator<Member> ir = hashSet.iterator();
    while(ir.hasNext()) {
        Member member = ir.next();
        int dbMemberId = member.getMemberId(); //이미 저장된 memberId를 가져옴
        if(dbMemberId == memberId) { //외부 입력 memberId와 같다면
            hashSet.remove(member); //회원 객체 삭제
            return true;
        }
    }
    System.out.println(memberId + "가 존재하지 않습니다.");
    return false;
}
```

Set 인터페이스

```
public class MemberHashSetTest {

    public static void main(String[] args) {
        //객체 생성
        MemberHashSet memberHashSet = new MemberHashSet();

        //회원 추가
        memberHashSet.addMember(new Member(1001, "네이버"));
        memberHashSet.addMember(new Member(1002, "카카오"));
        memberHashSet.addMember(new Member(1003, "엔씨소프트"));
        memberHashSet.addMember(new Member(1001, "네이버"));  //중복 불가

        //회원 목록 조회
        memberHashSet.showAllMember();
        System.out.println("==================================");

        //회원 삭제
        memberHashSet.removeMember(1003);
        memberHashSet.removeMember(1004);

        //회원 목록 조회
        memberHashSet.showAllMember();
    }
}
```

```
네이버 회원님의 아이디는 1001입니다.
카카오 회원님의 아이디는 1002입니다.
엔씨소프트 회원님의 아이디는 1003입니다.
==================================
회원 아이디 1004가 존재하지 않습니다.
네이버 회원님의 아이디는 1001입니다.
카카오 회원님의 아이디는 1002입니다.
```

HashSet 응용 프로그램

- 실습 예제

StudentTest의 출력 결과가 다음처럼 나오도록 Student 클래스를 구현해 보세요.

```
public class StudentTest {
    public static void main(String[] args) {
        HashSet<Student> set = new HashSet<>();

        set.add(new Student("100", "홍길동"));
        set.add(new Student("200", "강감찬"));
        set.add(new Student("300", "이순신"));
        set.add(new Student("400", "정약용"));
        set.add(new Student("100", "송중기"));

        System.out.println(set);
    }
}
```

<출력 결과>

400:정약용 100:홍길동, 200:강감찬, 300:이순신

HashSet 응용 프로그램

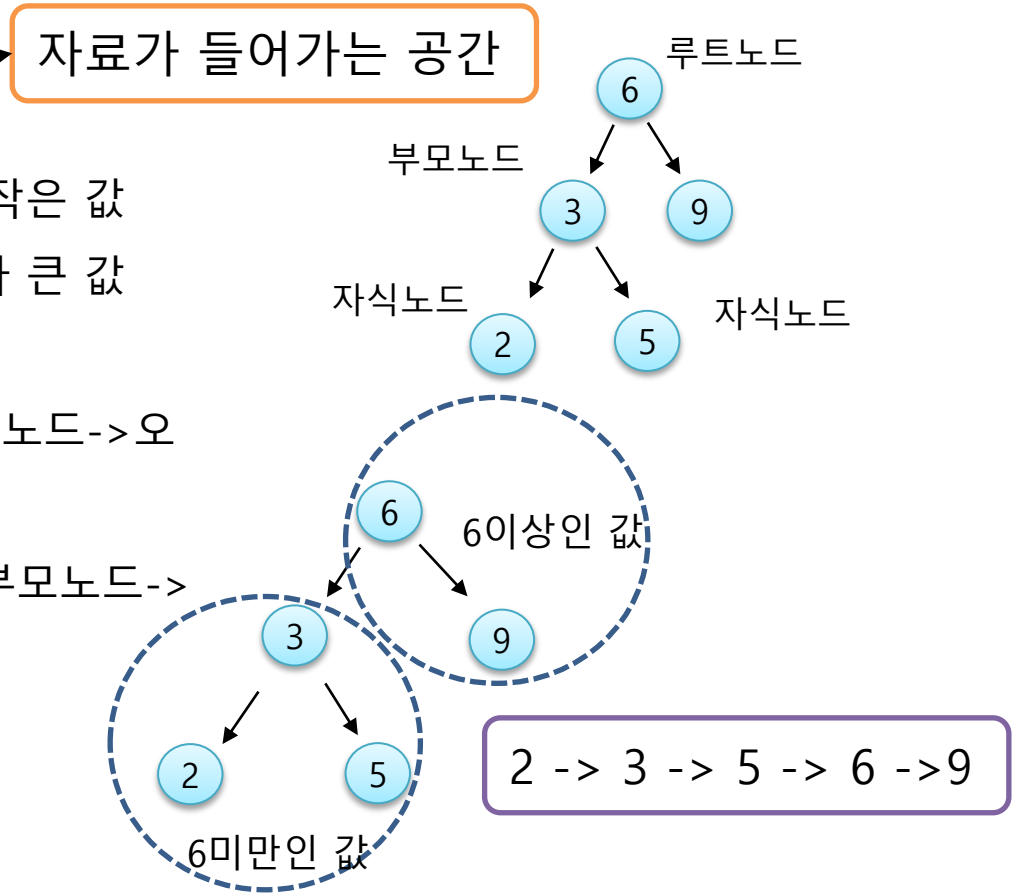
```
public class Student {
    private String number;
    private String name;

    public Student(String number, String name) {
        this.number = number;
        this.name = name;
    }

    @Override
    public String toString() {
        return number + ":" + name;
    }

    @Override
    public int hashCode() {
        return Integer.parseInt(number);
    }

    @Override
    public boolean equals(Object obj) {
        if(obj instanceof Student) {
            Student student = (Student)obj;
            if(this.number==student.number)
                return true;
        }
        return false;
    }
```

Set 인터페이스

◆ 검색 기능을 강화시킨 Set 컬렉션

이진 검색 트리 구조

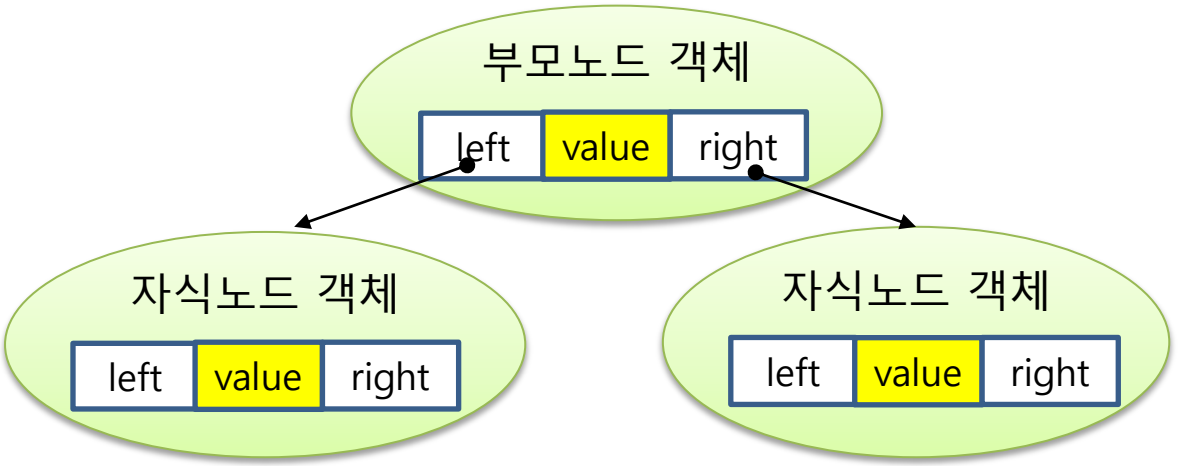
- **부모 노드와 자식 노드로 구성**
  - 왼쪽 자식노드 : 부모 보다 작은 값
  - 오른쪽 자식노드 : 부모 보다 큰 값
- **정렬이 쉬움**
  - 오름차순 : 왼쪽노드->부모노드->오른쪽노드
  - 내림차순 : 오른쪽노드 ->부모노드->왼쪽노드
- **범위 검색이 쉬움**



Set 인터페이스

◆ TreeSet 클래스

- 객체의 정렬에 사용되는 클래스
- 중복을 허용하지 않으면서 **오름차순이나 내림차순으로 객체를 정렬함**
- 내부적으로 이진 검색 트리(binary search tree)로 구현되어 있음
- 객체 비교를 위해 **Comparable**이나 **Comparator** 인터페이스를 구현해야 함



Set 인터페이스

◆ TreeSet 클래스

```
public class TreeSetTest2 {

    public static void main(String[] args) {
        TreeSet<String> names = new TreeSet<>();

        names.add("이현대");
        names.add("최삼성");
        names.add("김엘지");
        names.add("박한화");
        names.add("최삼성"); //중복 불가

        //조회
        System.out.println(names);

        for(String name : names) {  //기본 오름차순
            System.out.println(name);
        }

        //삭제
        names.remove("박한화");
        System.out.println(names);
    }
}
```

정렬이 되는 이유

```
public final class String
    implements java.io.Serializable, Comparable<String>, CharSequence {
```

```
[김엘지, 박한화, 이현대, 최삼성]
김엘지
박한화
이현대
최삼성
[김엘지, 이현대, 최삼성]
```

Set 인터페이스

◆ TreeSet 클래스

```java
public class TreeSetTest {

    public static void main(String[] args) {
        TreeSet<Integer> scores = new TreeSet<>();
        //TreeSet은 기본 오름차순 정렬
        //객체 추가
        scores.add(87);
        scores.add(98);
        scores.add(75);
        scores.add(95);
        scores.add(80);
        scores.add(98);  //중복 불가

        //객체 조회
        System.out.println(scores);

        for(Integer score : scores) {
            //set은 get(index)가 없으므로 향상 for문 사용
            System.out.println(score);
        }
```

```
[75, 80, 87, 95, 98]
75
80
87
95
98
가장 낮은 점수 : 75
가장 높은 점수 : 98
87 아래 점수 : 80
87 위의 점수 : 95
98(남은 객체 수 : 4)
95(남은 객체 수 : 3)
87(남은 객체 수 : 2)
80(남은 객체 수 : 1)
75(남은 객체 수 : 0)
```

Set 인터페이스

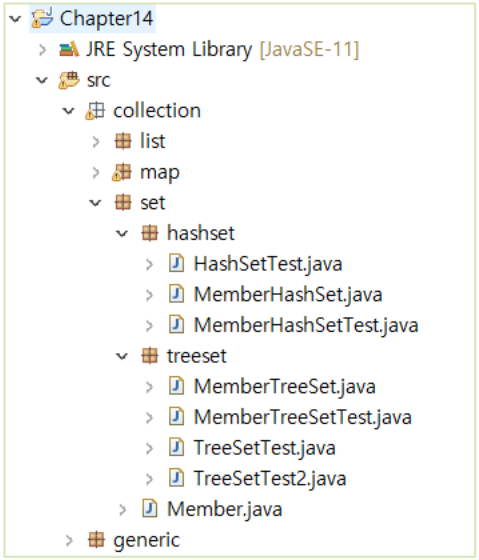
◆ TreeSet 클래스

```
        Integer score = scores.first();
        System.out.println("가장 낮은 점수 : "  + score);

        score = scores.last();
        System.out.println("가장 높은 점수 : " + score);

        score = scores.lower(87);
        System.out.println("87 아래 점수 : " + score);

        score = scores.higher(87);
        System.out.println("87 위의 점수 : " + score);

        //꺼내기
        while(!scores.isEmpty()) {
            score = scores.pollLast();  //큰 숫자부터 꺼내기(내림차순)
            System.out.println(score + "(남은 객체 수 : " + scores.size() + ")");
        }
}
```

Set 인터페이스

◆ TreeSet을 활용한 회원관리 프로그램

```java
package collection.set;

public class Member implements Comparable<Member>{
    private int memberId;        //회원 아이디
    private String memberName;   //회원 이름

    public Member(int memberId, String memberName) {
        this.memberId = memberId;
        this.memberName = memberName;
    }
```




```java
@Override
public boolean equals(Object obj) {
    if(obj instanceof Member) {
        Member member = (Member)obj;
        if(this.memberId==member.memberId)
            return true;
    }
    return false;
}

@Override
public int hashCode() {
    return memberId;
}

@Override
public int compareTo(Member member) {
    return (this.memberId -member.memberId);
}
```

Set 인터페이스

◆ TreeSet을 활용한 회원관리 프로그램

```
public class MemberTreeSet {
    private TreeSet<Member> treeSet;

    public MemberTreeSet() {
        treeSet = new TreeSet<>();
    }

    public void addMember(Member member) {
        treeSet.add(member);
    }

    //회원 목록
    public void showAllMember() {
        for(Member member : treeSet) {
            System.out.println(member);
        }
    }
```

```
//회원 삭제
public boolean removeMember(int memberId) {
    Iterator<Member> iterator = treeSet.iterator();
    while(iterator.hasNext()) { //회원 데이터가 있다면(true)
        Member member = iterator.next(); //하나씩 가져오기
        int dbMemberId = member.getMemberId(); //회원아이디를
        if(dbMemberId==memberId) {
            treeSet.remove(member);  //회원 객체를 삭제
            return true;
        }
    }
    return false;
}
```

Set 인터페이스

◆ TreeSet을 활용한 회원관리 프로그램

```
public class MemberTreeSetTest {

    public static void main(String[] args) {
        MemberTreeSet memberTreeSet = new MemberTreeSet();

        memberTreeSet.addMember(new Member(1002, "네이버"));
        memberTreeSet.addMember(new Member(1003, "카카오"));
        memberTreeSet.addMember(new Member(1001, "엔씨소프트"));
        memberTreeSet.addMember(new Member(1003, "카카오"));

        memberTreeSet.showAllMember();
    }
}
```

Member클래스가 Comparable 인터페이스를 구현하지 않았다

```
eption in thread "main" java.lang.ClassCastException: collection.Member cannot be cast to java.base/java.lang.Comparable
    at java.base/java.util.TreeMap.compare(TreeMap.java:1291)
    at java.base/java.util.TreeMap.put(TreeMap.java:536)
    at java.base/java.util.TreeSet.add(TreeSet.java:255)
    at collection.treeset.MemberTreeSet.addMember(MemberTreeSet.java:16)
    at collection.treeset.MemberTreeSetTest.main(MemberTreeSetTest.java:14)
```

Set 인터페이스

◆ TreeSet을 활용한 회원관리 프로그램

```
2
3  public class Member implements Comparable<Member>{
4
5      private i
6      private S
7
8      public Member(int memberId, String memberName) {
9          this.memberId = memberId;
10         this.memberName = memberName;
```

The type Member must implement the inherited abstract method Comparable<Member>.compareTo (Member)

2 quick fixes available:

- Add unimplemented methods
- Make type 'Member' abstract

Press 'F2' for focus

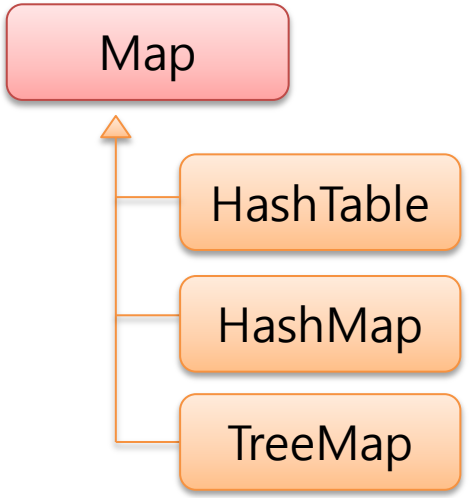
```
@Override
public int compareTo(Member member) {
    return (this.memberId - member.memberId);
}
```

오름차순 정렬

```
엔씨소프트 회원님의 아이디는 1001입니다.
네이버 회원님의 아이디는 1002입니다.
카카오 회원님의 아이디는 1003입니다.
```

Map 인터페이스

◆ Map 인터페이스



Module java.base
Package java.util

Interface Map<K,V>

Type Parameters:

K - the type of keys maintained by this map

V - the type of mapped values

All Known Subinterfaces:

Bindings, ConcurrentMap<K,V>, ConcurrentNavigableMap<K,V>,

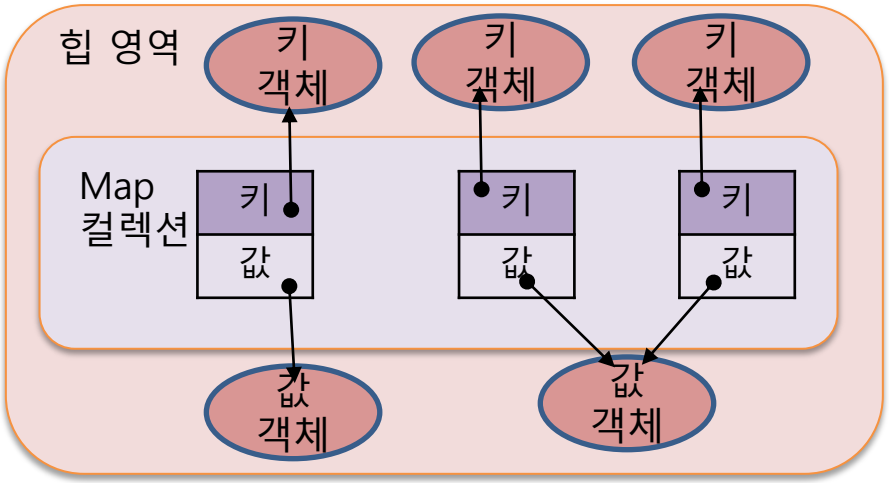
All Known Implementing Classes:

AbstractMap, Attributes, AuthProvider, ConcurrentHashMap, C
Provider, RenderingHints, SimpleBindings, TabularDataSuppo

```
public interface Map<K,V>
```

Map 인터페이스

◆ Map 인터페이스



분류	특 징
Map 인터페이스	- 키(key)와 값(Value)의 쌍으로 저장 - 키는 중복 저장 안되고, 값은 가능 - 구현 클래스 :**HashMap, HashTable, TreeMap**

Map 인터페이스

◆ Map 인터페이스

- Key-value pair의 객체를 관리하는데 필요한 메서드가 정의 됨
- Key를 이용하여 값을 저장하거나 검색, 삭제 할때 사용하면 편리함
- 내부적으로 hash 방식으로 구현됨 . **Index = hash(key)** ->해시함수가 위치 계산
- Key가 되는 객체는 유일성 여부를 알기 위해 equals()와 hashCode() 메서드를 재정의 함

키타입 값타입

```
Map<K, V> map = new HashMap<K, V>();
Map<String, Integer> map = new HashMap<>();
```

기능	메소드	설명
객체 추가	put(key, value)	주어진 키로 값을 저장
객체 검색	contains(object key)	주어진 키가 저장되어 있는지 여부
	isEmpty()	컬렉션이 비어 있는지 조사
	size()	저장되어 있는 키의 총 수를 리턴
객체 삭제	clear()	저장된 모든 키와 값을 삭제
	remove(Object key)	주어진 키와 일치하는 객체를 삭제하고 값을 리턴

Map 인터페이스

◆ HashMap

```
public class HashMapTest {

    public static void main(String[] args) {
        // 자료형이 String, Integer인 map 객체 생성
        Map<String, Integer> map = new HashMap<>();

        //자료 추가
        map.put("네이버", 400000);
        map.put("카카오", 150000);
        map.put("엔씨소프트", 600000);
        map.put("네이버", 450000);  //key는 중복불가, value는 변경 가능

        //총 객체수
        System.out.printf("총 객체수 : %d\n", map.size());

        //자료 출력 - map 형태
        System.out.println(map);
```

Map 인터페이스

◆ HashMap

```
//자료 1개 검색
System.out.println("네이버 : " + map.get("네이버")); //key로 가져옴
System.out.println("================================");

//전체 검색
/*Set<String> keySet = map.keySet(); //map에서 찾아와서 Set 객체 생성
Iterator<String> ir = keySet.iterator();*/

Iterator<String> ir = map.keySet().iterator();
while(ir.hasNext()) {
    String key = ir.next();  //다음 자료 key를 가져오고
    Integer value = map.get(key); //key로 검색해서 value를 가져옴
    System.out.println(key + " : " + value);
}

//자료 삭제
if(map.containsKey("카카오")) {
    map.remove("카카오");
}

//총 객체수
System.out.println(map);
```

```
총 객체수 : 3
{카카오=150000, 네이버=450000, 엔씨소프트=600000}
네이버 : 450000
================================
카카오 : 150000
네이버 : 450000
엔씨소프트 : 600000
{네이버=450000, 엔씨소프트=600000}
```

Map 인터페이스

◆ HashMap 예제

```
var map = new HashMap<String, Integer>();
int idx = 0;
map.put("Java",       ++idx);
map.put("C",          ++idx);
map.put("C++",        ++idx);
map.put("JavaScript", ++idx);
map.put("Python",     ++idx);
map.put("PHP",        ++idx);

Set<String> keys = map.keySet();

System.out.printf("총 %d개의 Map Entry가 있습니다.\n", keys.size());

//요소 삭제 후 추가하기
if(map.containsKey("PHP")) map.remove("PHP");
map.put("Delphi",     idx);
```

Map 인터페이스

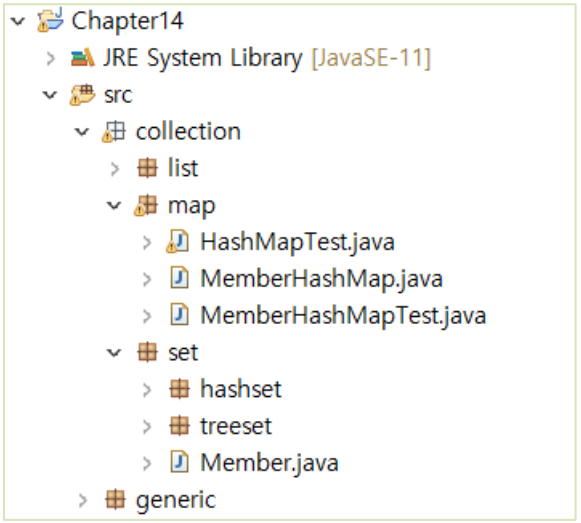
◆ HashMap 예제

```
//전체 요소 조회
for(String key : keys) {
    System.out.println(key + " : " + map.get(key));
}

System.out.println("==Lambda 식====");
keys.forEach(key -> System.out.println(key));

System.out.println("====Method reference====");
map.keySet().forEach(System.out::println);

//자료 존재 유무
System.out.println(map.containsKey("PHP"));
```

```
총 6개의 Map Entry가 있습니다.
Java : 1
C++ : 3
C : 2
JavaScript : 4
Delphi : 6
Python : 5
==Lambda 식====
Java
C++
C
JavaScript
Delphi
Python
====Method reference====
Java
C++
C
JavaScript
Delphi
Python

false
```

Map 인터페이스

◆ HashMap을 활용한 회원관리 프로그램




```java
package collection.map;

import java.util.HashMap;
import java.util.Iterator;

import collection.set.Member;

public class MemberHashMap {
    HashMap<Integer, Member> hashMap;

    public MemberHashMap() {
        hashMap = new HashMap<>();
    }

    //회원 추가
    public void addMember(Member member) {
        hashMap.put(member.getMemberId(), member);
    }
```

Map 인터페이스

◆ HashMap을 활용한 회원관리 프로그램

```
//회원 목록
public void showAllMember() {
    Iterator<Integer> ir = hashMap.keySet().iterator();
    while(ir.hasNext()) {
        int key = ir.next(); //key값을 가져와서
        Member member = hashMap.get(key);  //키로부터 value 가져오기
        System.out.println(member);
    }
    System.out.println();
}

//회원 삭제
public boolean removeMember(int memberId) {
    if(hashMap.containsKey(memberId)) { //입력받은 회원아이디가 존재한다면
        hashMap.remove(memberId);  //해당 회원 삭제
        return true;
    }
    System.out.println(memberId + "가 존재하지 않습니다.");
    return false;
}
```

Map 인터페이스

```
public class MemberHashMapTest {

    public static void main(String[] args) {
        //mHashMap 객체 생성
        MemberHashMap mHashMap = new MemberHashMap();

        //회원 추가
        mHashMap.addMember(new Member(1001, "삼성전자"));
        mHashMap.addMember(new Member(1002, "LG전자"));
        mHashMap.addMember(new Member(1003, "네이버"));
        mHashMap.addMember(new Member(1004, "카카오"));
        mHashMap.addMember(new Member(1002, "현대자동차"));

        //회원 목록 조회
        mHashMap.showAllMember();
        System.out.println("==================================");

        //회원 삭제
        mHashMap.removeMember(1001);
        mHashMap.removeMember(1005);

        //조회
        mHashMap.showAllMember();
    }
}
```

```
삼성전자 회원님의 아이디는 1001입니다.
현대자동차 회원님의 아이디는 1002입니다.
네이버 회원님의 아이디는 1003입니다.
카카오 회원님의 아이디는 1004입니다.
==================================
회원 아이디 1005가 존재하지 않습니다.
현대자동차 회원님의 아이디는 1002입니다.
네이버 회원님의 아이디는 1003입니다.
카카오 회원님의 아이디는 1004입니다.
```

HashMap 응용 프로그램

- 실습 예제

다음 코드에서 CarTest의 테스트 결과가 true, true, false가 되도록 HashMap을 사용하여 CarFactory 클래스를 구현해 보세요

```java
public class Car {
    String name;

    public Car() {}

    public Car(String name) {
        this.name = name;
    }
}
```

```java
public class CarTest {
    public static void main(String[] args) {
        CarFactory factory = CarFactory.getInstance();
        Car sonata1 = factory.createCar("수소차");
        Car sonata2 = factory.createCar("수소차");
        System.out.println(sonata1==sonata2);    //true

        Car avante1 = factory.createCar("전기차");
        Car avante2 = factory.createCar("전기차");
        System.out.println(avante1==avante2);    //true

        System.out.println(sonata1==avante1);    //false
    }
}
```

HashMap 응용 프로그램

```
public class CarFactory {
    private static CarFactory instance;
    private HashMap<String, Car> carMap = new HashMap<>();

    private CarFactory() {}

    public static CarFactory getInstance() {
        if(instance==null) {
            instance = new CarFactory();
        }
        return instance;
    }

    public Car createCar(String name) { //차이름을 매개변수로 전달함
        if(carMap.containsKey(name)) { //카맵에 차이름이 있다면
            return carMap.get(name);   //차이름을 가져와서
        }
        Car car = new Car();
        carMap.put(name, car);   //카맵에 이름과 차 객체를 추가
        return car;
    }

    @Override
    public String toString() {
        return "carMap=" + carMap;
    }
}
```