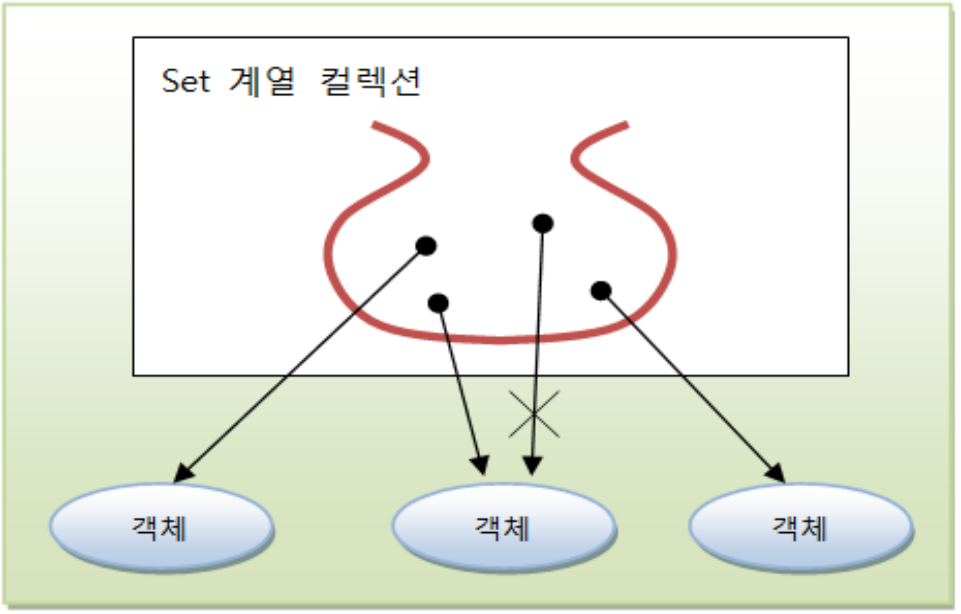
Set 컬렉션

❖ Set 컬렉션의 특징 및 주요 메소드

- 특징
  - 수학의 집합에 비유
  - 저장 순서가 유지되지 않음
  - 객체를 중복 저장 불가
  - 하나의 null만 저장 가능

- 구현 클래스
  - HashSet, LinkedHashSet, TreeSet



Set 컬렉션

❖ Set 컬렉션의 특징 및 주요 메소드

- ㅇ 주요 메소드

기능	메소드	설명
객체 추가	boolean add(E e)	주어진 객체를 저장, 객체가 성공적으로 저장되면 true 를 리턴하고 중복 객체면 false 를 리턴
객체 검색	boolean contains(Object o)	주어진 객체가 저장되어 있는지 여부
	isEmpty()	컬렉션이 비어 있는지 조사
	Iterator<E> iterator()	저장된 객체를 한번씩 가져오는 반복자 리턴
	int size()	저장되어있는 전체 객체수 리턴
객체 삭제	void clear()	저장된 모든 객체를 삭제
	boolean remove(Object o)	주어진 객체를 삭제

❖ Set 컬렉션의 특징 및 주요 메소드

- 전체 객체 대상으로 한 번씩 반복해 가져오는 반복자(Iterator) 제공
  - 인덱스로 객체를 검색해서 가져오는 메소드 없음

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

리턴 타입	메소드명	설명
boolean	hasNext( )	가져올 객체가 있으면 true를 리턴하고 없으면 false를 리턴한다.
E	next( )	컬렉션에서 하나의 객체를 가져온다.
void	remove( )	Set 컬렉션에서 객체를 제거한다.

❖ Set 컬렉션의 특징 및 주요 메소드

- 전체 객체 대상으로 한 번씩 반복해 가져오는 반복자(Iterator) 제공

```
Set<String> set = …;
Iterator<String> iterator = set.iterator();
while(iterator.hasNext()) {
    //String 객체 하나를 가져옴
    String str = iterator.next();
}
```

저장된 객체 수만큼 루핑한다.

- 향상된 for 문으로 대체 가능

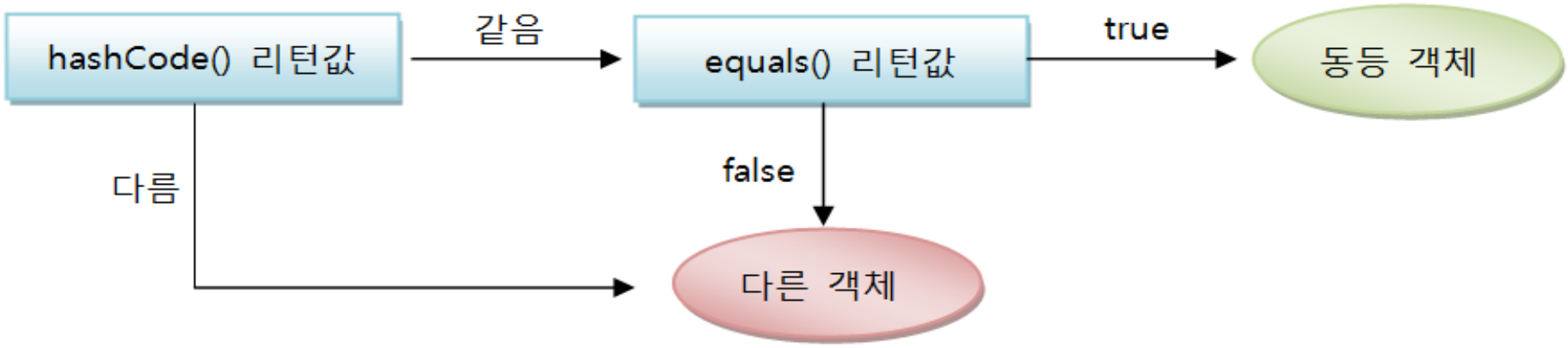
```
Set<String> set = …;
for(String str : set) {
}
```

저장된 객체 수만큼 루핑한다.

❖ HashSet

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

- 특징
  - 동일 객체 및 동등 객체는 중복 저장하지 않음
  - 동등 객체 판단 방법



❖ String 객체를 중복 없이 저장하는 HashSet: HashSetExample1.java

```
import java.util.*;

public class HashSetExample1 {
    public static void main(String[] args) {
        Set<String> set = new HashSet<String>();

        set.add("Java");
        set.add("JDBC");
        set.add("Servlet/JSP");
        set.add("Java");
        set.add("iBATIS");

        int size = set.size();
        System.out.println("총 객체수: " + size);

        Iterator<String> iterator = set.iterator();
        while(iterator.hasNext()) {
            String element = iterator.next();
            System.out.println("\t" + element);
        }
```

❖ String 객체를 중복 없이 저장하는 HashSet: HashSetExample1.java

```
        set.remove("JDBC");
        set.remove("iBATIS");

        System.out.println("총 객체수: " + set.size());

        for(String element : set) {
            System.out.println("\t" + element);
        }

        set.clear();
        if(set.isEmpty()) { System.out.println("비어 있음"); }
    }
}
```

❖ hashCode()와 equals() 메서드 재정의: Member.java

```
public class Member {
    public String name;
    public int age;

    public Member(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public boolean equals(Object obj) {
        if(obj instanceof Member) {
            Member member = (Member) obj;
            return member.name.equals(name) && (member.age==age) ;
        } else {
            return false;
        }
    }

    public int hashCode() {
        return name.hashCode() + age;
    }
}
```

❖ Member 객체를 중복없이 저장하는 HashSet: HashSetExample2.java

```
import java.util.*;

public class HashSetExample2 {
    public static void main(String[] args) {
        Set<Member> set = new HashSet<Member>();

        set.add(new Member("홍길동", 30));
        set.add(new Member("홍길동", 30));

        System.out.println("총 객체수 : " + set.size());
    }
}
```