

# Lec\_08 컬렉션

---

자바 프로그래밍2\_12주

# 컬렉션 프레임워크 소개

- 컬렉션 프레임워크(Collection Framework)
  - 컬렉션
    - 사전적 의미로 요소(객체)를 수집해 저장하는 것
- 배열의 문제점
  - 저장할 수 있는 객체 수가 배열을 생성할 때 결정
    - 불특정 다수의 객체를 저장하기에는 문제
  - 객체 삭제했을 때 해당 인덱스가 비게 됨
    - 낱알 빠진 옥수수 같은 배열
    - 객체를 저장하려면 어디가 비어 있는지 확인해야


배열

0	1	2	3	4	5	6	7	8	9
●	●	×	●	×	●	×	●	●	×

# 컬렉션 프레임워크 소개

- 컬렉션 프레임워크(Collection Framework)
  - 객체들을 효율적으로 추가, 삭제, 검색할 수 있도록 제공되는 컬렉션 라이브러리
  - java.util 패키지에 포함
  - 인터페이스를 통해서 정형화된 방법으로 다양한 컬렉션 클래스 이용
- 컬렉션 프레임워크의 주요 인터페이스

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



# List 컬렉션

- 특징
  - 배열과 비슷하게 객체를 인덱스로 관리
  - 저장 용량 자동 증가, 객체 저장 시 자동 인덱스 부여
  - 중복해서 객체 저장 가능

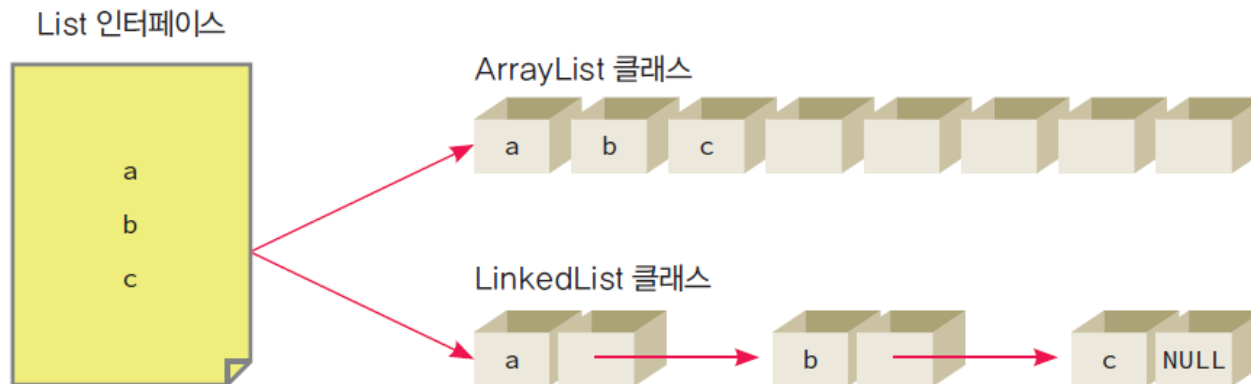
- 구현 클래스
  - ArrayList
  - Vector
  - LinkedList

- 주요 메소드

기능	메소드	설명
객체 추가	<code>boolean add(E e)</code>	주어진 객체를 맨끝에 추가
	<code>void add(int index, E element)</code>	주어진 인덱스에 객체를 추가
	<code>set(int index, E element)</code>	주어진 인덱스에 저장된 객체를 주어진 객체로 바꿈
객체 검색	<code>boolean contains(Object o)</code>	주어진 객체가 저장되어 있는지 여부
	<code>E get(int index)</code>	주어진 인덱스에 저장된 객체를 리턴
	<code>isEmpty()</code>	컬렉션이 비어 있는지 조사
	<code>int size()</code>	저장되어있는 전체 객체수를 리턴
객체 삭제	<code>void clear()</code>	저장된 모든 객체를 삭제
	<code>E remove(int index)</code>	주어진 인덱스에 저장된 객체를 삭제
	<code>boolean remove(Object o)</code>	주어진 객체를 삭제

# List 컬렉션

- ArrayList 클래스-타입 매개변수를 갖는 제네릭 클래스로 제공
  - 배열(Array)의 향상된 버전 또는 가변 크기의 배열
  - 객체만 가능 //기초 타입은 JDK1.5 이후 자동 박싱으로 해결
- LinkedList 클래스
  - 빈번하게 삽입과 삭제가 일어나는 경우에 사용



- Vector
  - 동기화된 메소드로 구성
    - 하나의 스레드가 실행을 완료해야 다른 스레드를 실행
    - 복수의 스레드가 동시에 Vector에 접근해 객체를 추가, 삭제하더라도 스레드에 안전

# Vector 사용 예1

```
import java.util.Vector;
public class VectorTest {
    public static void main(String[] args) {
        Vector vc = new Vector(); //크기 생략 가능
        vc.add("Hello World!"); //어떤 타입의 객체도 추가(add()) 가능
        vc.add(new Integer(10));
        vc.add(20);
        System.out.println("vector size :" + vc.size()); //size():저장된 원소 개수
        for (int i = 0; i < vc.size(); i++) {
            System.out.println("vector element " + i + " :" + vc.get(i));
        }
    }
}
```

vector size :3  
vector element 0 :Hello World!  
vector element 1 :10  
vector element 2 :20

# Vector 사용 예2

```
import java.util.*;
class Board{
    private String writer; //글쓴이
    private String title; //제목
    public Board(String writer, String title) {
        this.writer = writer; this.title = title;
    }
    public String toString() {
        return "Board [writer=" + writer + ", title=" + title + "];"
    }
}
```

```
public class VectorExam {
    public static void main(String[] args) {
        List<Board> list=new Vector<>();
        list.add(new Board("java", "List"));
        list.add(new Board("js", "Event"));
        list.add(new Board("C++", "Virtual"));
        System.out.println("----- 삭제 전 -----");
        for(int i=0; i<list.size();i++)
            System.out.println(list.get(i).toString());

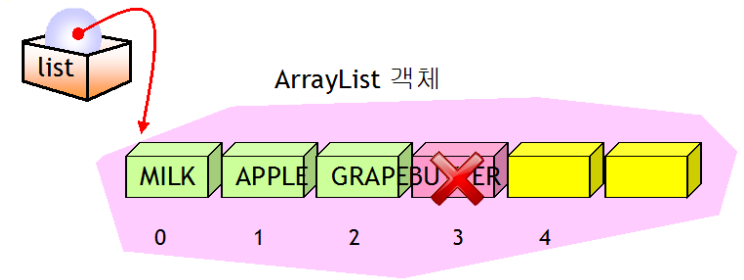
        list.remove(1);
        System.out.println("----- 삭제 후 -----");
        for(Board b : list)
            System.out.println(b.toString());
    }
}
```

# ArrayList 사용 예

```
import java.util.*;
public class ArrayListTest {
    public static void main(String args[]) {
        List<String> list = new ArrayList<String>(); //ArrayList 객체 생성
```

```
list.add("MILK"); //ArrayList에 원소 추가
list.add("BREAD");
list.add("BUTTER");
list.add(1, "APPLE"); // 인덱스 1에 "APPLE"을 삽입
list.set(2, "GRAPE"); // 인덱스 2의 원소를 "GRAPE"로 대체
list.remove(3); // 인덱스 3의 원소 제거
```

```
for (int i = 0; i < list.size(); i++) // size()는 원소 개수 반환
    System.out.println(list.get(i)); // get()는 i번째 위치의 원소 반환
//for(String s:list)
//    System.out.println(s)
}}
```





# 배열을 리스트로 변환하기

---

- 일반적인 배열을 리스트로 변환
  - `List<String> list = Arrays.asList(new String[size]);`
- List 추가 연산
  - 특정 데이터의 위치 반환 - `indexOf()`
    - 동일한 데이터의 경우 맨 처음에 있는 데이터 위치 반환  
`int index = list.indexOf("APPLE");`
  - 반대 방향으로 검색 - `lastIndexOf()`  
`int index = list.lastIndexOf("MILK");`

# 배열을 리스트로 변환 예

```
import java.util.*;
public class ArraysAsListExample {
    public static void main(String[] args) {
        List<String> list1 = Arrays.asList("홍길동", "신용권", "홍길동");
        for(String name: list1) {
            System.out.println(name);
        }
        System.out.println("홍길동 객체 위치 : " + list1.indexOf("홍길동")); //0
        System.out.println("홍길동 객체 위치 : " + list1.lastIndexOf("홍길동")); //2

        List<Integer> list2 = Arrays.asList(1, 2, 3);
        for(int value : list2) {
            System.out.println(value);
        }
    }
}
```

# LinkedList 사용 예

```
import java.util.*;
public class LinkedListTest {
    public static void main(String args[]) {
        LinkedList<String> list = new LinkedList<String>();

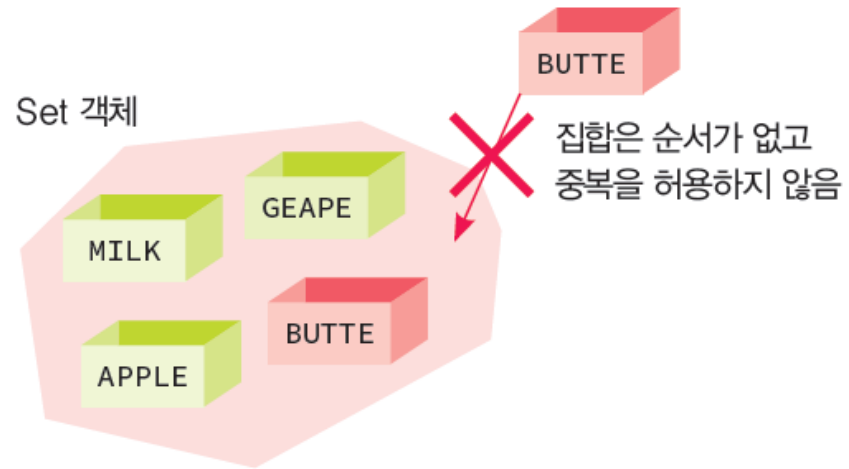
        list.add("MILK");
        list.add("BREAD");
        list.add("BUTTER");
        list.add(1, "APPLE"); // 인덱스 1에 "APPLE"을 삽입
        list.set(2, "GRAPE"); // 인덱스 2의 원소를 "GRAPE"로 대체
        list.remove(3); // 인덱스 3의 원소를 삭제한다.
        for (int i = 0; i < list.size(); i++)
            System.out.println(list.get(i));
    }
}
```

실행결과

MILK  
APPLE  
GRAPE

# Set 컬렉션

- Set 컬렉션의 특징 및 주요 메소드
  - 특징
    - 수학의 집합에 비유
    - 저장 순서가 유지되지 않음
    - 객체 중복 저장 불가
    - 하나의 null만 저장 가능
  - 구현 클래스
    - HashSet, LinkedHashSet, TreeSet



# Set 컬렉션

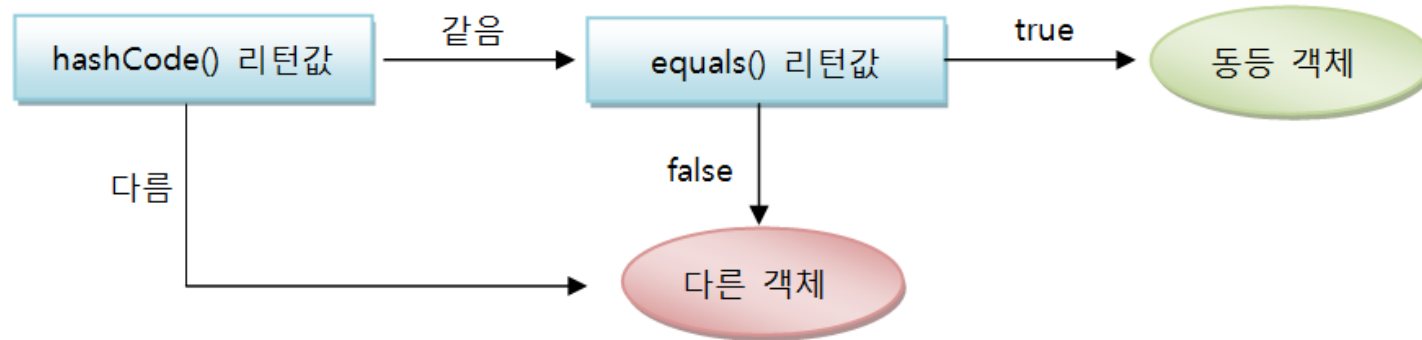
- Set 컬렉션의 특징 및 주요 메소드
  - 주요 메소드

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

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

# Set 구현 클래스

- HashSet
  - HashSet은 해시 테이블에 원소를 저장하기 때문에 성능면에서 가장 우수
  - 원소들의 순서가 일정하지 않은 단점이 있다.
  - 동등 객체 판단 방법



- TreeSet
  - 값에 따라서 순서가 결정되지만 HashSet보다는 느리다.
- LinkedHashMap
  - 해시 테이블과 연결 리스트를 결합한 것으로 원소들의 순서는 삽입되었던 순서와 같다.

# HashSet 활용 - 문자열 객체 저장하기

```
import java.util.*;
public class SetExam {
    public static void main(String[] args) {
        Set<String> set = new HashSet<String>();

        set.add("Milk");
        set.add("Bread");
        set.add("Butter");
        set.add("Cheese");
        set.add("Ham");
        //중복된 원소는 추가되지 않음
        set.add("Ham");

        //반복자를 사용한 출력
        //반복자 얻기
        Iterator<String> iterator = set.iterator();
        while(iterator.hasNext()) {
            String element = iterator.next();
            System.out.print("    "+element); }
        System.out.println();
```

//객체 삭제

```
set.remove("Butter");
set.remove("Ham");
```

//람다식을 사용한 출력

```
set.forEach((element)->System.out.print("\t"+element));
```

```
set.add("Apple");
```

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

//향상된 for문을 사용한 출력

```
for(String element : set)
    System.out.print("    "+element);
```

```
set.clear(); //set에 저장된 모든 객체 제거
```

```
if(set.isEmpty()) //set이 비어있으면 true 반환
```

```
    System.out.println("Hashset이 비었습니다");
```

```
}}
```

# HashSet 활용 - 중복되는 단어 찾기

```
import java.util.*;
public class HashSetTest {
    public static void main(String args[]) {
        Set<String> s = new HashSet<String>();
        String[] sample = { "java", "C++", "C#", "C++" };
        for (String a : sample)
            if (!s.add(a)) //중복되는 단어는 저장되지 않으며 false 반환
                System.out.println("중복된 단어==> " + a);
        System.out.println("HashSet에 저장된 단어 개수 : " + s.size());
        System.out.println("중복되지 않은 단어: " + s);
    }
}
```

중복된 단어==> C++  
HashSet에 저장된 단어 개수 : 3  
중복되지 않은 단어: [C#, C++, java]



# HashSet 활용 - 중복된 객체 없이 저장하기

```
import java.util.*;
class Member{
    private String id;
    private int age;
    public Member(String id, int age) {
        this.id = id;
        this.age = age;
    }
    @Override
    public String toString() {
        return "Member [id=" + id + ", age=" + age + "];"
    }
    @Override
    public int hashCode() {
        return id.hashCode()+age;
    }
}
```

```
@Override
public boolean equals(Object obj) {
    if(obj instanceof Member) {
        Member mem = (Member)obj;
        return (mem.id.equals(id)) && (mem.age == age);
    }
    else { return false; }
}

public class HashSetExam {
    public static void main(String[] args) {
        Set<Member> set=new HashSet<>();
        set.add(new Member("hallym", 30));
        set.add(new Member("software", 25));
        set.add(new Member("hallym", 30));

        System.out.println("저장된 객체 개수 : " + set.size());
    }
}
```

# LinkedHashSet & TreeSet

- LinkedHashSet 사용

```
LinkedHashSet<String> set = new LinkedHashSet<>();
```

- 입력된 순서대로 출력

## 실행결과

```
[Milk, Bread, Butter, Cheese, Ham]
```

```
set.add("Milk");  
set.add("Bread");  
set.add("Butter");  
set.add("Cheese");  
set.add("Ham");  
set.add("Ham");
```

- TreeSet 사용

```
TreeSet<String> set = new TreeSet<>();
```

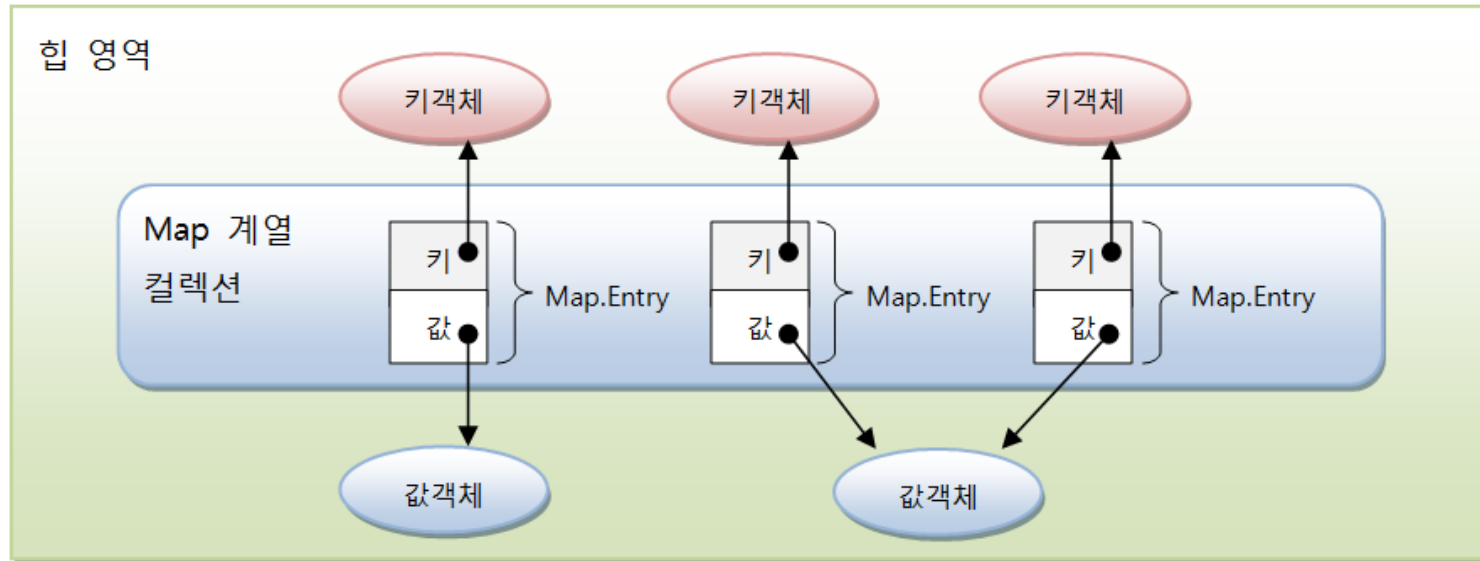
- 알파벳 순으로 정렬

## 실행결과

```
[Bread, Butter, Cheese, Ham, Milk]
```

# Map 컬렉션

- 특징
  - 키(key)와 값(value)으로 구성된 Map.Entry 객체를 저장하는 구조
  - 키와 값은 모두 객체
  - 키는 중복될 수 없지만 값은 중복 저장 가능
- 구현 클래스
  - HashMap, Hashtable, LinkedHashMap, Properties, TreeMap



# Map 컬렉션

- 주요 메소드

기능	메소드	설명
객체 추가	V put(K key, V value)	주어진 키와 값을 추가, 저장이 되면 값을 리턴
객체 검색	boolean containsKey(Object key)	주어진 키가 있는지 여부
	boolean containsValue(Object value)	주어진 값이 있는지 여부
	Set<Map.Entry<K,V>> entrySet()	키와 값의 쌍으로 구성된 모든 Map.Entry 객체를 Set에 담아서 리턴
	V get(Object key)	주어진 키의 값을 리턴
	boolean isEmpty()	컬렉션이 비어있는지 여부
	Set<K> keySet()	모든 키를 Set 객체에 담아서 리턴
	int size()	저장된 키의 총 수를 리턴
	Collection<V> values()	저장된 모든 값 Collection에 담아서 리턴
객체 삭제	void clear()	모든 Map.Entry(키와 값)를 삭제
	V remove(Object key)	주어진 키와 일치하는 Map.Entry 삭제, 삭제가 되면 값을 리턴

# Map 사용 예(1/2)

```
import java.util.*;
class Student {
    private int number;
    private String name;

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

    public String toString() {
        return name;
    }
}
```

```
public class MapExam {
    public static void main(String[] args) {
        Map<Integer, Student> map = new HashMap<>();
        map.put(201, new Student(201, "펑수"));
        map.put(202, new Student(202, "펑하"));
        map.put(203, new Student(203, "펑바"));
        //키값이 같으면 마지막에 저장한 값으로 대체
        map.put(202, new Student(202, "펑성"));
        map.put(204, new Student(204, "펑수1"));

        // 키 값에 해당하는 value 반환
        System.out.println("map.get(203) value : " + map.get(203));
        System.out.println("--- keySet으로 모든 키 얻기 ---");
        Set<Integer> keyset = map.keySet(); //keyset 얻기
        Iterator<Integer> keytor = keyset.iterator();
        while(keytor.hasNext()) {
            Integer key=keytor.next();
            Student value = map.get(key);
            System.out.println("key=" + key + ", value=" + value);
        }
    }
}
```

# Map 사용 예(2/2)

---

```
map.remove(204); // 하나의 항목을 삭제한다.
```

```
System.out.println("--- entrySet()으로 Map.Entry 객체 얻기 ---");
```

```
for (Map.Entry<Integer, Student> s : map.entrySet()) { // 모든 항목을 방문한다
```

```
    Integer key = s.getKey(); //key값 반환
```

```
    Student value = s.getValue(); //value 반환
```

```
    System.out.println("key=" + key + ", value=" + value);
```

```
}
```

```
System.out.println("--- 람다식으로 출력 ---");
```

```
    map.forEach((key, value)->{
```

```
        System.out.println("key = " + key + ", value = " + value);
```

```
    });
```

```
}
```

```
}
```

# Map 사용 예 - 빈도수 계산

```
import java.util.*;

public class WordFreq{
    public static void main(String[] args){
        Map<String, Integer> m=new HashMap<String, Integer>();
        String[] sample = {"to", "be", "or", "not", "to", "be", "is", "a", "pro"};
        //문자열에 포함된 빈도수 계산
        for(String a : sample){
            Integer freq=m.get(a); //key에 해당하는 value 반환
            m.put(a, (freq==null)?1:freq+1);
        }
        System.out.println(m.size() + " 단어가 있습니다");
        System.out.println(m.containsKey("to")); //해당 키 존재하면 true
        System.out.println(m.isEmpty()); //해쉬 맵이 비어 있으면 true
        System.out.println(m); //해쉬맵에 있는 모든 데이터 출력
    }
}
```

```
7 단어가 있습니다
true
false
{a=1, not=1, be=2, or=1, is=1, to=2, pro=1}
```

# Map 사용 예 – 사전

```
public class EnglishDic {  
    public static void main(String[] args) {  
        Map<String, String> st = new HashMap<String, String>();  
        Scanner sc = new Scanner(System.in);  
        st.put("map", "지도");  
        st.put("java", "자바");  
        st.put("school", "학교");  
        do {  
            System.out.print("영어 단어를 입력하시오:");  
            String key = sc.next();  
            if( key.equals("quit") ) break;  
            System.out.println("단어의 의미는 " + st.get(key));  
        } while(true);  
    }  
}
```