Java프로그래밍

11강. 컬렉션 (교재 10장)

컴퓨터과학과 김희천 교수

한국방송통신대학교

오늘의 학습목차

1. JCF
2. HashSet, ArrayList, LinkedList 클래스
3. HashMap 클래스

1. Java Collections Framework

1. Java Collections Framework

1) 컬렉션

◆ 여러 원소를 하나의 그룹으로 묶어 관리해주는 객체

Java Collections Framework(JCF)

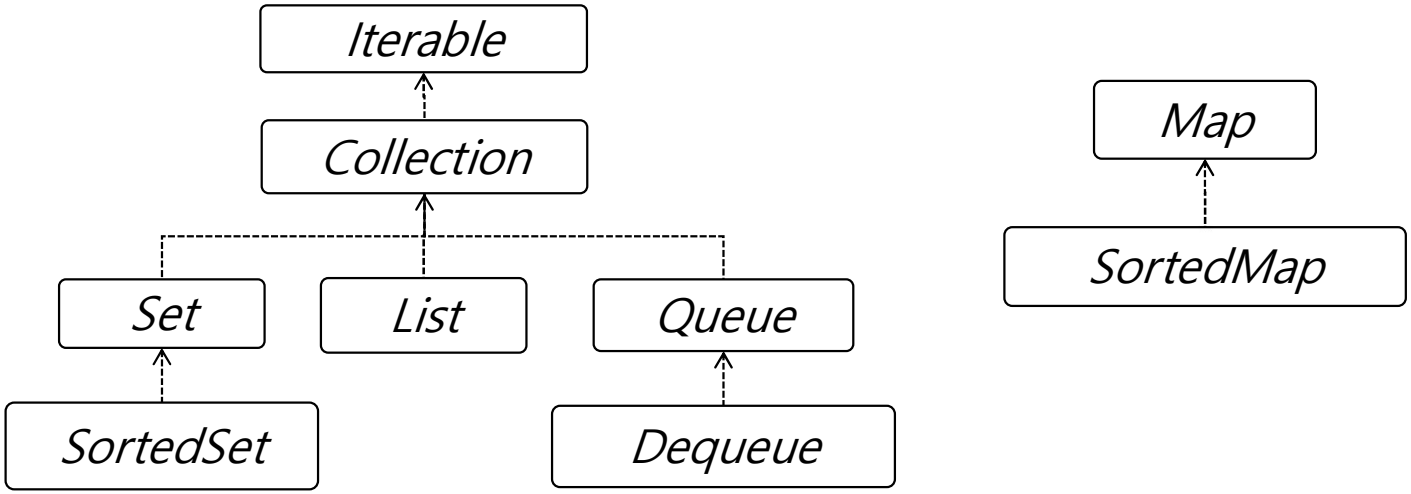
◆ 컬렉션을 표현하고 다루기 위한 통합된 프레임워크
  ✓ 클래스와 인터페이스의 집합
  ✓ 다양한 방식으로 저장, 정렬, 검색, 수정하는 도구를 제공

◆ 컬렉션을 일관된 방법으로 다룰 수 있음
  ✓ 표준화된 인터페이스: 컬렉션의 기능을 표현
    • 어떻게 표현되는 지와 상관없이 일관성 있게 다룸
  ✓ 클래스: 인터페이스를 구현한 클래스를 제공

2) JCF의 구조

◆ JCF의 인터페이스

- ✓ Set: 순서는 의미가 없으며 중복을 허용하지 않는 자료구조
- ✓ List: 중복을 허용하고 순서에 의미가 있는 자료구조
- ✓ Queue: List와 유사하나 원소의 삽입/삭제가 FIFO 방식
- ✓ Map: 원소가 <key, value>의 형태이며 키는 유일해야 함



3) JCF의 인터페이스와 클래스

◆ java.util 패키지에 포함되며 제네릭 타입

✓ 다루는 자료의 유형을 지정해야 함

	Set	*List*	*Queue*	*Map*
해싱	**HashSet**			**HashMap**
배열		**ArrayList** Vector(Stack)		
연결리스트		**LinkedList**	**LinkedList**	
해싱+ 연결리스트	LinkedHashSet			LinkedHashMap
	SortedSet			*SortedMap*
트리	TreeSet			TreeMap

4) 컬렉션 객체의 선언

- 변수 선언은 해당 인터페이스 유형으로,
  객체 생성은 인터페이스를 구현하는 클래스를 사용

```
Set<Integer> set = new HashSet<>();
List<Integer> list = new ArrayList<>();
List<Integer> list = new LinkedList<>();
Queue<Integer> queue = new LinkedList<>();
Map<String, Integer> map = new HashMap<>();
```

5) Collection<E> 인터페이스(1)

- Set, List, Queue에서 공통으로 지원해야 하는 기능을 정의

원소 삽입/삭제 메소드

- boolean add(E e)
  - ✓ 컬렉션에 변화를 주면 true를 리턴함
- boolean addAll(Collection<? extends E> c)
- boolean remove(Object o)
- boolean removeAll(Collection<?> c)
- boolean retainAll(Collection<?> c)
- void clear( )

5) Collection<E> 인터페이스(2)

원소 탐색 메소드

- boolean contains(Object o)
- boolean containsAll(Collection<?> c)
- boolean isEmpty( )

기타 메소드

- int size( )
- int hashCode( )
- Object[ ] toArray( )
- Iterator<E> iterator( )
- boolean equals(Object)

2. HashSet, ArrayList, LinkedList 클래스

2. HashSet, ArrayList, LinkedList 클래스

1) HashSet 클래스 예제

```
import java.util.*;

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

        set.add("one");      set.add("two");
        set.add("three");    set.add("four");
        System.out.println(set.add(new String("one")));

        System.out.println(set.size( ));
        System.out.println(set.contains("four"));
        System.out.println(set.contains("one"));
        System.out.println(set.contains(new String("one")));

        set.remove("four");
        set.remove(new String("one"));
        System.out.println(set.size());

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

```
false
4
true
true
true
2
0
```

2) ArrayList 클래스

- List 인터페이스를 구현한 클래스
  - ✓ 크기 조절이 가능한 배열로 구현
- 같은 자료가 중복될 수 있으며, 입력된 순서대로 관리됨
  - ✓ 특정 위치의 자료를 참조할 수 있음
- List 인터페이스를 살펴봐야 함

메소드

- boolean add(E e), void add(index, E element)
- boolean remove(Object o), E remove(int index)
- E get(int index), E set(int index, E element)
- int indexOf(Object o), int lastIndexOf(Object o)

3) List<E> 인터페이스

- 순서가 있고 중복을 허용하는 구조
- 원소를 순차적으로 처리하는 구조
  - ✓ 첨자에 의한, 특정 위치의 원소 처리가 가능

메소드

- int indexOf(Object o)
- int lastIndexOf(Object o)
- E set(int index, E element)
- List<E> subList(int from, int to)
- E remove(int index),
- boolean remove(Object o)
- ListIterator<E> listIterator( )
- ListIterator<E> listIterator(int index)

2. HashSet, ArrayList, LinkedList 클래스

4) ArrayList 클래스 예제(1)

```
import java.util.*;

public class ArrayListTest {
    public static void main(String args[]) {
        List <String> list = new ArrayList <String> ( );

        list.add("one");      list.add("two");
        list.add("three");    list.add(1, "one"); //삽입
        list.add("five");

        System.out.println(list.size( ));
        System.out.println(list.indexOf("one"));
        System.out.println(list.get(2));
        System.out.println(list.lastIndexOf("one"));
        System.out.println(list.set(3, "four")); //대체
        System.out.println(list.remove(4));
        System.out.println(list.remove("one"));
    }
}
```

```
5
0
two
1
three
five
true
```

4) ArrayList 클래스 예제(2)

```
import java.util.*;

public class ArrayListTest2 {
    public static void main(String args[ ]) {
       List<String> list = new ArrayList<String>( );

       list.add("one");
       list.add("two");
       list.add("three");
       list.add("four");
       list.add("five");

// for 구문을 이용한 자료 탐색
        for (int i = 0; i < list.size( ); i++)
           System.out.println(list.get(i));
```

```
// foreach 구문을 이용한 자료 탐색
for (String s : list)
     System.out.println(s);

// Iterator 인터페이스를 이용한 자료 탐색
Iterator<String> it = list.iterator();
while (it.hasNext( ))
     System.out.println(it.next( ));
 }
}
```

4) ArrayList 클래스 예제(3)

```
import java.util.*;
import java.util.function.Consumer;

public class ArrayListTest {

  public static void main(String args[ ]) {

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

     list.add("one"); list.add("three");
     list.add("two"); list.add(1, "one");

     Consumer<String> con1 = new Consumer<>() {
           public void accept(String t) {
              System.out.println(t);
           }
     };
     list.forEach( con1 );
```

```
Consumer<String> con2 = t -> System.out.println(t);
list.forEach( con2 );


list.forEach( t -> System.out.println(t) );


   }


}
```

5) Iterator<E> 인터페이스

- 컬렉션에 저장된 원소를 차례대로 다룰 수 있음
- 다음 메소드를 제공
  - ✓ boolean hasNext( ), E next( ), void remove( )
- HashSet, ArrayList, LinkedList 등에서 Iterator 객체를 리턴하는 메소드가 정의됨

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


Iterator <String> it = list.iterator( );
while(it.hasNext( ))
       System.out.println(it.next( ));
```

6) LinkedList 클래스

- ArrayList와 마찬가지로 List 인터페이스를 구현한 클래스
  - ✓ 앞의 예제에서 ArrayList를 LinkedList로 바꿔도 됨
- Queue 인터페이스를 구현함
  - ✓ 아래 표
- 스택 자료구조에서 필요한 메소드도 제공함
  - ✓ void push(E), E pop( ) //앞에서 넣거나 뺌

Queue 인터페이스의 메소드

- boolean offer(E), boolean add(E) //뒤에 넣고
- E poll( ), E remove( ) //앞에서 빼고
- E peek( ), E element( )

7) LinkedList를 이용하여 큐를 구현한 예제

```
import java.util.*;

public class QueueTest {
    public static void main(String args[]) {
       LinkedList<String> queue = new LinkedList<String>( );

        queue.offer("one");    queue.offer("two");
        queue.offer("three");  queue.offer("four");

        String s = queue.poll( );
        while (s != null) {
               System.out.println(s);
               s = queue.poll( );
        }
    }
}
```

```
one
two
three
four
```

3. HashMap 클래스

3. HashMap 클래스

1) Map<K, V> 인터페이스

- (key, value)을 갖는 원소로 구성되는 컬렉션을 다루기 위한 인터페이스
  - ✓ key는 중복되지 않으며, 하나의 key에 하나의 value만 대응됨

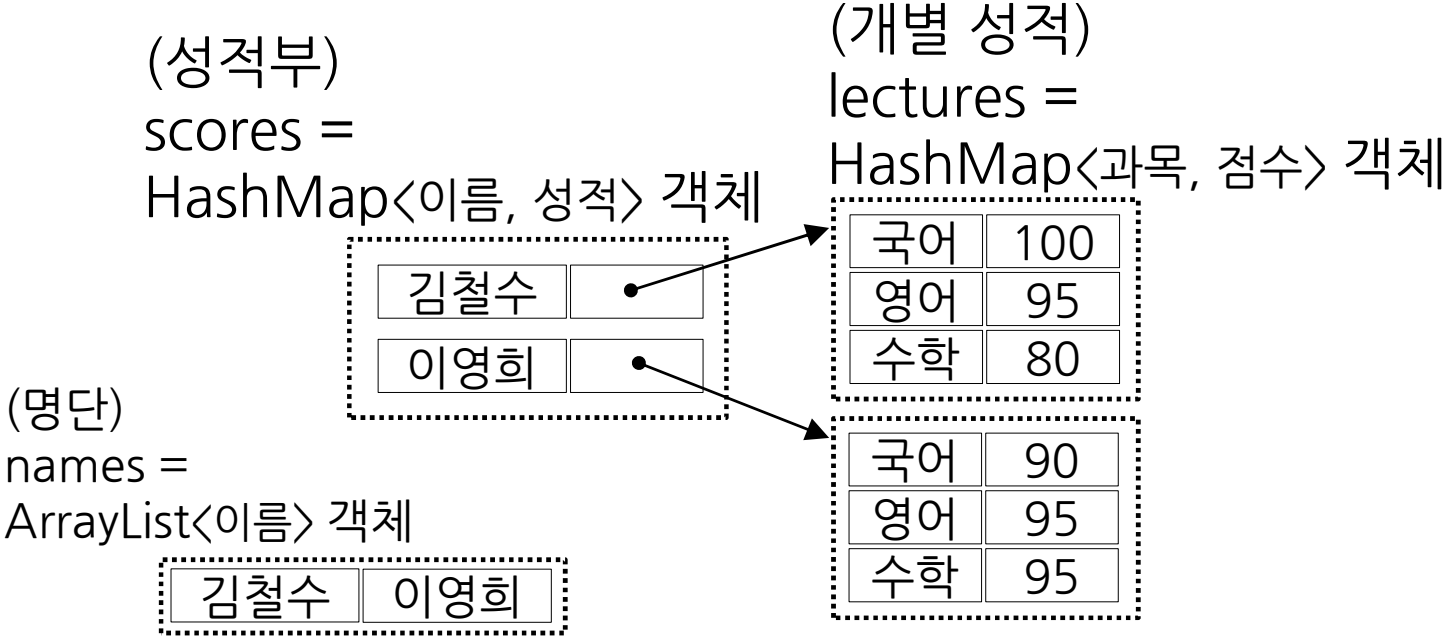
메소드

- V put(K key, V value)
- V get(Object key)
- V remove(Object key)
- boolean containsKey(Object key)
- Collection<V> values( )
- Set<K> keySet( )

2) HashMap 클래스

- 해싱을 이용하여 Map 인터페이스를 구현한 클래스
  - ✓ 자료 탐색 방법이 ArrayList나 LinkedList 클래스와 다름

- 복잡한 자료 관리(교재의 예)



3) 복잡한 자료 관리 - HashMap 클래스 예제

```
import java.util.*;

public class HashMapTest {
   public static void main(String args[ ]) {
     List <String> names = new ArrayList<String> ( );
     Map <String, Integer> lectures;
     Map <String, Map> scores = new HashMap<>( );

     names.add("김철수");  names.add("이영희");

     Iterator <String> it = names.iterator();
     while(it.hasNext( )) {
       String name = it.next();
       if (name.equals("김철수")) {
           lectures = new HashMap<String, Integer>( );
           lectures.put("국어", 100);
           lectures.put("영어", 95);
           lectures.put("수학", 80);
           scores.put(name, lectures);
       } else if (name.equals("이영희")) {
           lectures = new HashMap<String, Integer>( );
          … … …
           scores.put(name, lectures);
       }
     }
```

```
Iterator <String> it2 = names.iterator( );
while(it2.hasNext()) {
     String name = it2.next();
     System.out.println(name);

     System.out.print("국어 : ");
     System.out.println(
     scores.get(name).get("국어"));

     System.out.print("영어 : ");
     System.out.println(
     scores.get(name).get("영어"));

     System.out.print("수학 : ");
     System.out.println(
     scores.get(name).get("수학"));
     System.out.println( );
  }
 }
}
```

```
김철수
국어 : 100
영어 : 95
수학 : 80

이영희
… …
```

Java프로그래밍

다음시간안내

12강. 멀티 스레드 프로그래밍 (교재 11장)