용어정리

1. Collection Framework이란?

데이터 군(群)을 저장하는 클래스들을 표준화한 설계

- 다수의 데이터를 쉽게 처리할 수 있는 방법을 제공하는 클래스들로 구성
- JDK 1.2 부터 제공

▶ 컬렉션(Collection)

- 다수의 데이터, 즉, 데이터 그룹을 의미한다.

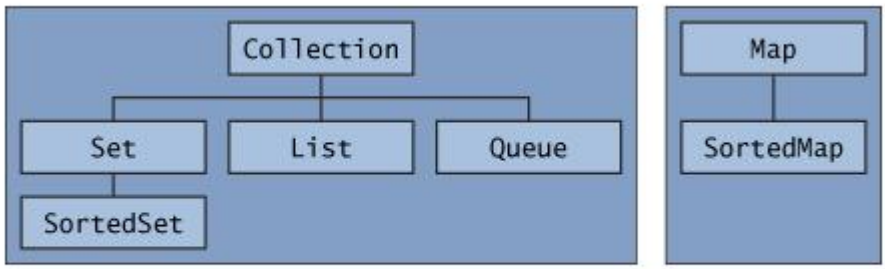
▶ 프레임웍(Framewrok)

- 표준화, 정형화된 체계적인 프로그래밍 방식

▶ 컬렉션 클래스(Collection Class)

- 다수의 데이터를 저장할 수 있는 클래스 (예, Vector, ArrayList, HashSet..._)

출처 : 자바의 정석, 자바프로그래밍 Bible(영진)
http://mousevm.tistory.com/221



The core collection interfaces.

[1] 컬렉션 핵심 인터페이스

<1> Collection

- 컬렉션 계층 구조의 최상위를 이루는 인터페이스
- **중복될 수 있는 객체들의 집합(2번 이상 같은 객체가 포함될 수 있음을 의미)을 나타냄**
- Iterable을 상속하며, 원소의 추가, 삭제, 검색, 크기 지원 등의 메소드를 추가로 지원한다.

<참고> - Iterable=>기본적인 컬렉션 인터페이스로 순차 열람(iteration)을 지원한다

<2> List

원소의 순서가 정의되어 있으며, 컬렉션상의 위치를 통해 원소에 접근할 수 있다.
따라서 **List를 사용하면 컬렉션 상에서의 인덱스를 통해 어떤 원소를 접근 할 수 있다.**
원소간의 순서가 중요한 경우, 예를 들어 도착 순서대로 메세지를 처리하는 큐의 경우에는 리스트를 사용해야 한다.

<3> Set

중복된 원소가 없는 컬렉션(원소 중복이 불가능, 순서를 보장하지 않음)

중복원소(상호간 equals()의 결과가 참인 원소)를 허용하지 않는 컬렉션.
원소 사이의 순서가 없으므로, 이전 순차 열람할 때의 원소 순서가 다음 순차 열람할 때 보장되지 않는다.

<4> SortedSet

중복된 원소가 없으나 원소간의 순서가 정해진 컬렉션 (정렬된 집합)

컬렉션에 추가된 순서나 명시적인 인덱스 번호에 따라 순서가 정해지는 List와 달리 SortedSet은 **Comparator에 의해 순서를 정한다**. 명시적인 순서를 제공하지 않는 경우에는 "자연 순서(natural order)"가 사용된다. 예를 들어 문자열은 알파벳 순으로 정렬

아래는 Comparator의 사용예

```
public Collection<String> getAlphabeticalAuthors()
{
    Comparator<Author> sorter = new Comparator<Author>()
    {
        public int compare(Author o1, Author o2)
        {
            if (o1.getLastName().equals(o2.getLastName()))
            {
                return o1.getFirstName().compareTo(o2.getFirstName());
                //return o1.getLastName().compareTo(o2.getLastName());
            }
        };

        SortedSet<Author> results = new TreeSet<Author>(sorter);
        for (Book each: getBooks())
        {
            results.add(each.getAuthor());
        }

        return results;
    }
}
```

<5>Map

키에 의해 원소를 저장하고 접근하는 컬렉션

키와 밸류를 매핑하여 저장함

키의 중복은 허용하지 않으나 밸류의 중복은 허용함
Map은 List처럼 키를 사용해서 원소를 저장하지만, List가 정수만을 키로 사용할 수 있는 반면 Map은 임의의 객체를 키로 사용할 수 있다.
또 Map는 다른 컬렉션 인터페이스와는 형태가 상이하여 다른 컬렉션 인터페이스를 상속하지 않고, 내부적으로 키에 대한 컬렉션과 데이터에 대한 컬렉션의 2개 컬렉션을 유지한다.
<6> SortedMap
- 키값에 따라서 정렬된 상태의 Map
- SortedSet 과 비슷하지만 차이점은 SortedSet의 경우 값에 대해 정렬하지만, SortedMap은 키에 의한 정렬을 한다는 것이 다를 뿐이다.

[2] 구현 클래스

General-purpose Implementations

Interfaces	Implementations				
	Hash table	Resizable array	Tree	Linked list	Hash table + Linked list
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		LinkedList	
Queue					
Map	HashMap		TreeMap		LinkedHashMap

컬렉션에 대해 구현 클래스를 선택하는 것은 주로 성능과 관련이 있다.
위의 표에 소개한 구현 이외에도 무지하게 많은 구현들이 있다. 각 구현체의 특성을 살펴보고 필요한 것을 가져다 사용하면 된다.
일단 가장 단순한 구현을 사용하여 시작하고 추후 경험에 따라 튜닝하는 것이 좋다.
컬렉션중 가장 많이 사용되는 클래스는 ArrayList이며, 그 다음은 HashSet이다.
(이클립스와 JDK에서 ArrayList는 3400번, HashSet은 800번 사용되었다고 한다.)

<1> List 구현
ArrayList와 LinkedList
ArrayList는 원소 접근이 빠르고 원소 추가 및 제거가 느린 반면 LinkedList는 원소 접근이 느리지만 원소 추가와 제거는 빠르다.

<2> Set, SortedSet 구현
HashSet은 가장 빠르지만 원소간의 순서를 보장해주지 않는다.
LinkedHashSet은 원소 간 순서를 보장해 주지만 원소 추가 삭제 시 30% 정도 시간이 더 걸린다.
TreeSet은 Comparator에 따라서 원소를 정렬하지만 원소 추가 삭제 시간이 logn(n은 컬렉션의 크기)에 비례해서 커진다.

<3> Map 구현

Map 구현은 Set 구현과 비슷한 패턴을 보인다.

HashMap은 가장 빠르고 단순하다.

LinkedHashMap은 컬렉션에 추가된 원소 간의 순서를 보장한다.

TreeMap(SortedMap 의 구현)은 키의 순서에 따라 순차 열람이 가능하지만 원소의 추가 제거 시간이 logn(n은 컬렉션의 크기)에 비례한다.

<4>. Collections

Collections는 다른 컬렉션 인터페이스에 넣기 적절치 않은 기능들을 모아 놓은 **유틸리티 클래스**이다.

- 검색

indexOf() 연산에 걸리는 시간은 리스트의 크기에 비례한다. 원소들이 정렬되어 있을 경우 Collections.binarySearch(list, element)를 사용하여 log2n에 비례하는 시간에 검색할 수 있다.

원소가 리스트에 존재하지 않는다면 음수를 반환하고, 리스트가 정렬되어 있지 않다면 결과는 예측불가

- 정렬

reverse(list)는 리스트에 속해 있는 모든 원소 간의 순서를 거꾸로 바꾼다.

shuffle(list)는 순서를 임의로 바꾼다.

sort(list), sort(list, comparator)는 오름차순으로 원소를 정렬한다.

이진 검색과 달리 ArrayList와 LinkedList에서 정렬 수행 성능은 거의 같다.

정렬을 수행할 경우 컬렉션의 원소들이 일단 배열로 복사되어 정렬된 후 다시 본래의 컬렉션으로 복사되기 때문

- 수정 불가능한 컬렉션

신뢰할 수 없는 코드에 컬렉션을 전달하는 경우 Collections.unmodifiableCollection() 메소드를 이용하면 클라이언트가 수정하려 들 경우 예외를 발생시키도록 할 수 있다.

- 단일 원소 컬렉션

하나의 원소를 전달해야 하지만 컬렉션 인터페이스를 사용해야 하는 경우 사용

Set의 경우 Collections.singleton(T o), List와 Map의 경우 singletonList(T o), singletonMap(K key, V value)를 사용

- 무원소 컬렉션
컬렉션 인터페이스를 사용해야 하지만 전달할 원소가 없는 경우에는 Collections에서 수정할 수 없는 무원소 컬렉션을 생성해서 사용
Collections.emptyList(), emptySet(), emptyMap()

- 동기화 컬렉션
이전 시대의 유물인 Vector와 Hashtable이 ArrayList와 HashMap간의 차이점은 전자가 쓰레드 안전인 반면 후자는 아니라는 것이다.
동기화가 필요없는 경우라면 ArrayList, HashMap을 사용하고 동기화가 필요한 경우 Collections.synchronizedCollection(), Collections.synchronizedList(), Collections.synchronizedSet(), Collections.synchronizedMap()를 사용하여 ArrayList, HashMap을 래핑하면 멀티 쓰레드 환경에서도 걱정이 사라진다.

2. ORM (Object Relational Mapping) 이란?

오브젝트(객체) 관계 연결 이란 의미를 가진 약자이다.
한마디로는 DataBase Table 과 JavaBean을 맵핑시켜서 DataBase의 작업을 할 수 있게 해주는 개념
ORM 기법에 따른 프레임워크로는 Hibernate, IBatis,(MyBatis) Oracle TopLink 등이 있다.