**Collection**

1. **제네릭(Generic)**

자료형을 정하지 않는 기법.

사용하는 쪽에서 타입을 정해주는 기법

**제네릭 문법**

<대문자>

E: Element

K: Key

N: Number

T: Type

V: Value

**Object로 설정하지 않고 제네릭으로 설계하는 이유**

1. up casting 된 후 자식에서 선언된 필드를 사용할 수 없기 때문에

2. 들어올 수 있는 타입의 제한을 줄 수 있기 때문에

**1. Generic Class**

클래스 내부에서 사용될 제네릭을 선언한다.

선언 시 클래스 명 뒤에 선언한다.

**2. Generic Method**

클래스가 아닌 메소드에서 선언된 제네릭을 의미한다.

리턴 타입 앞에 제네릭을 선언한다.

**3. Generic Interface**

지정한 클래스에서 타입을 결정해준다.

1. **List**

***ArrayList***

**컬렉션 클래스 중 가장 많이 사용되는 클래스.**

**배열을 이용해서 값을 저장한다.**

**인덱스를 이용해서 배열 요소에 빠르게 접근이 가능하다.**

**하지만, 인덱스를 고려해서 저장해야 하기 때문에**

**인덱스 관리 부분에서 시간이 많이 소요된다.**

**배열과 ArrayList**

**배열: 크기에 제한을 두기 위해 사용한다.**

**ArrayList보다 상대적으로 속도가 빠르다.**

**ArrayList: 크기에 제한을 둘 수 없다.**

**크기가 정해져 있지 않기 때문에 가변적인 데이터를 저장할 수 있다.**

**다양한 기능들을 제공받는다.**

**=======================asList===================**

1. ***LinkedList(큐 구조)***

ArrayList보다 LinkedList가 더 빠르다

.addFirst : 맨 처음에 넣는 것

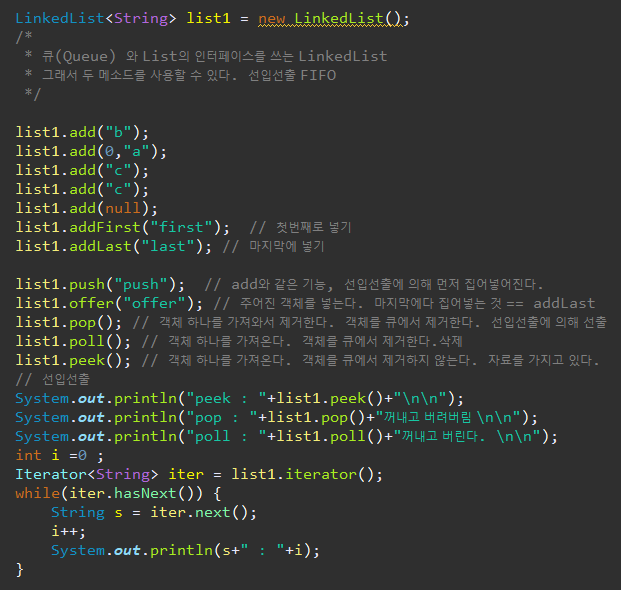
.push : 맨 처음에 넣는 것(addFirst랑 동일)

.pop: first element 출력하고 삭제

.poll: first element 출력하고 삭제

.peek: 처음 것 출력하고 유지

.peekFirst : 처음거 출력하고 없으면 null return

.remove(): 객체를 선택적으로 지울 수 있다.아무것도 적지 않으면 선입선출에 의해 선출, remove(인덱스번호)

1. **Set**

Set은 집합이다.

중복되는 원소를 포함시킬 수 없다.

값(데이터, 객체)을 중복해서 저장할 수 없다.

저장된 값들은 저장 순서가 고정되어 있지 않다. (iterator가 아니다, 값을 가져올수 없다.)

Set의 목적

값의 유무 검사.

Set에 순서를 부여하는 방법

iterator(): 순서가 없는 자료구조에 순서를 부여해주는 메소드

**HashSet**: 중복 안되고 순서없는 형태

**ThreeSet(2진트리):** HashSet에서 값에따라 (a-z,1-9)순서 생성, 검색 속도가 빠르다. 가장 뛰어나다. 그러나 삽입과 삭제할 때 다른 트리구조도 변경해야해서 오래걸림

**LinkedHashSet:** add한 순서에 따라 순서가 결정

{10, 0, 8, 9, 12, 8, -10, 5, 0}

*자료를 TreeSet으로 넣었다면,*

비교자료보다 작으면 왼쪽 자식, 크면 오른쪽 자식, 같으면 트리에 달지 않는다.

정렬은 TreeSet으로 달아주고 중위오더로 읽으면 오름차순 정렬된 자료가 추출.

-10 0 5 8 9 10 12

1. **Map**

****

*※비 객체지향 처리 코드*

Object[] key;

Object[] value;

*※객체지향 처리 코드*

Entry[] table;

Class Entry{

Object key;

Object value;

}

HashMap

Key와 Value 한 쌍(Entry)으로 저장된다.

검색에 용이하다.

Key는 중복이 불가하며, Value는 중복이 가능하다.

KEY -> |해쉬 테이블| -> VALUE

순서없 순서있(규칙성은 있어)

(key에 같은 값 넣으면 덮어 씀)

Map 자료구조는 순서가 없기 때문에

값을 가져올 때에는 Iterator가 반드시 필요하다.

각각의 자료구조 분리

1. Key만 필요할 때: keySet() //Set type(중복없애려고->set)

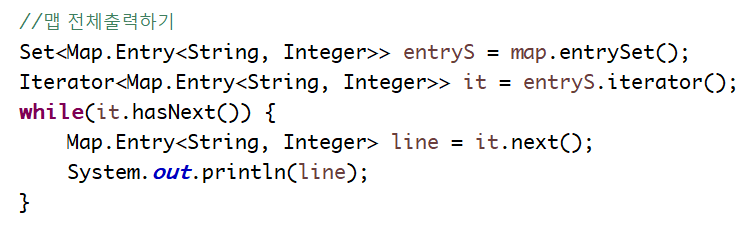
2. Value만 필요할 때: valuse() //Collection type(중복가능해->순서있어)

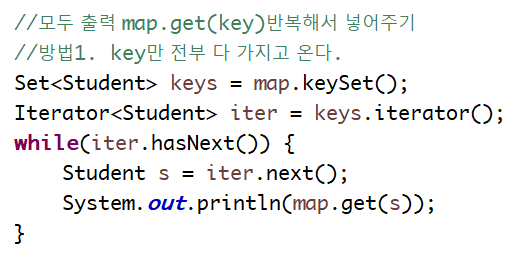
3. 한 쌍씩 순서대로 가져와야 할때: entrySet()

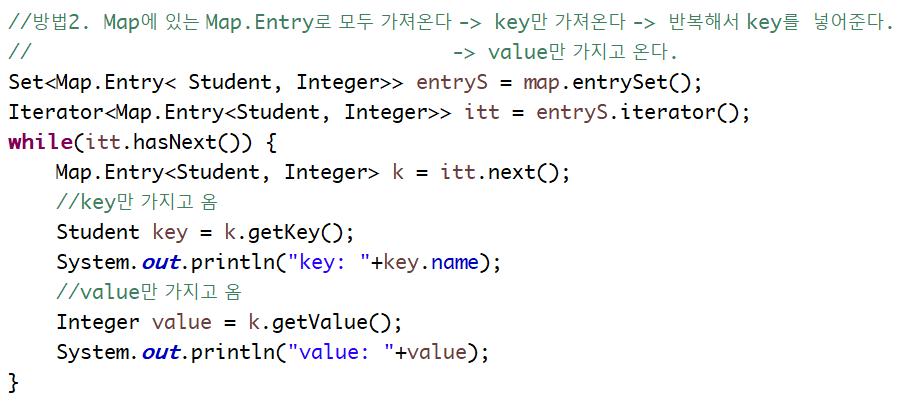
순서가 있다 -> iterator

순서가 없다? -> iterator() -> 순서 생김

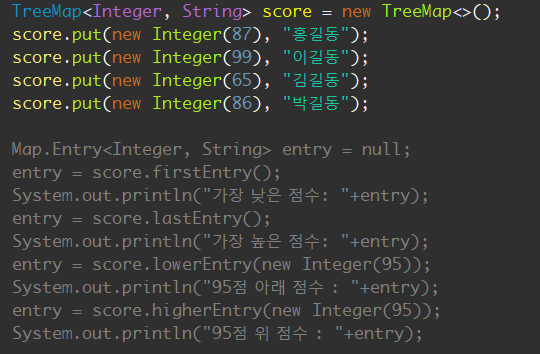
***HashMap 전체출력***

1)

2)

3)

***TreeMap***





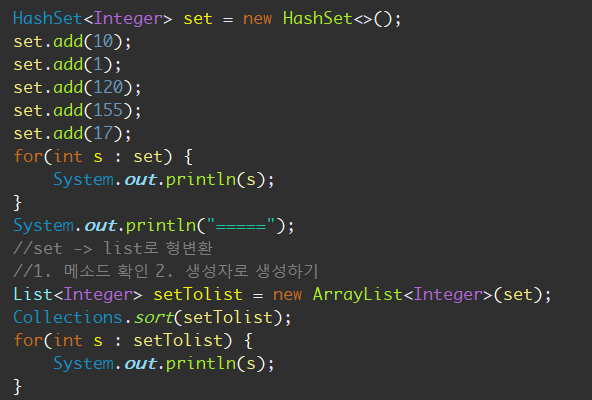
1. **Collections**

* ***Sort***

1. LinkedList



1. HashSet



1. Map

