

0316

인터페이스 : 서로 관련 없는 프로그램 끼리 묶어 사용할 수 있게 되는 것.

LinkedList : **List** 후손으로 앞과 뒤의 주소값만 기억한 순서임으로 수정 추가제거가 굉장히 편하다
원래라면 List에서 Linked로 바꿀데 오류가 나지만 Liked는 Llist에서 상속을 받은거기 때문에 문제가 발생하지 않는다.

결합도 :

한줄을 바꾸면 코드를 바꿔야되네 : 코드 결합도가 높구나

한줄을 바꿔도 바꿀점이 별로 없네? : 결합도가 낮네

객체지향은 결합도가 최대한 낮은 걸 목표로 한다.



빨간색이 주요 사용.

Vector - **ArrayList**의 구 버전.

- **Vector**는 동기화(**Synchronized**)를 제공한다 (**List**중 성능이 가장 안 좋음)

Array - 검색 / 조회

Comparable : **Comparator** (비교와 정렬) : 모든 DB를 sql로 정렬

List : 배열 모양

Set : 주머니 (마구잡이로 집어 넣음)

- 중복값이 들어올 경우 원래 있던 값이 달라짐
- 값이 Null 도 중복할 수 없다.



Set<String> set = new HashSet<String>();

Set의 자식인 HashSet이라는 객체를 부모 참조형 Set을 String 자료만 저장 가능한 set컬렉션을 생성한다.

참고 Hash라는 단어가 붙은 컬렉션은 반드시 equals() hashCode()를 오버라이딩 해야한다/



중복의 수가 없다

Set은 순서가 없어서 저장된 객체를 하나를 얻어올 수 있는 방법이 없다.
대신 Set전체의 데이터를 하나씩 반복적으로 얻어올 수는 있다

set . hashCode
equals 오버라이딩
hashCode 오버라이딩
Wrapper으로 기본 자료형을 객체로 포장했다.

Map

키(Key): 인덱스 번호를 대신함
값(Value):데이터
키와 값은 모두 객체
키는 중복저장을 허용하지 않는다.
값은 중복 저장이 된다
키가 중복 될 경우 기존에 있던 키에 해당하는 값을 덮어 씌운다

구현클래스

- HashMap
- HashTable
- LinkedHashMap
- Properties
- TreeMap

Map에서는 키를 기억해 값을 찾아 간다

List , Set , Map 특징

기본 사용법

equals() , hashCode() 오버라이딩

기본자료형 Wrapper Class으로 박싱 언박싱

향상된 for문

 12_입출력(IO).pdf

723 kB

입출력 IO

input : 외부에서 내부로 값이 들어오는 것

output : 내부에서 외부로 값이 나가는 것

외부와 내부사이에서 값이 왔다 갔다하는 공간 : 스트림 (Stream)
서버가 클라이언트에게 값을 제공할 때 필요한게 스트림



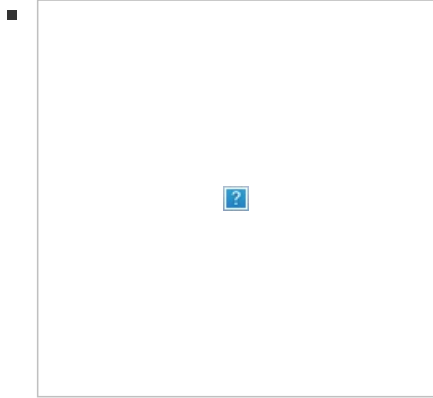
스트림은 일방 통행이다.
스트림(Stream)클래스



오늘 배운 것 정리

List , Set , Map 특징

- List
 - 배열과 비슷한 형태
 - 중복 저장이 가능하다
 - 순서를 유지한다
 - 배열의 종류는 ArrayList , LinkedList , Vector이 있다
 - ArrayList : List중 가장 기본적인 형태
 - 최초 저장 용량은 10칸 이다 저장 용량을 초과하면 자동적으로 늘어나며 고정도 가능하다.

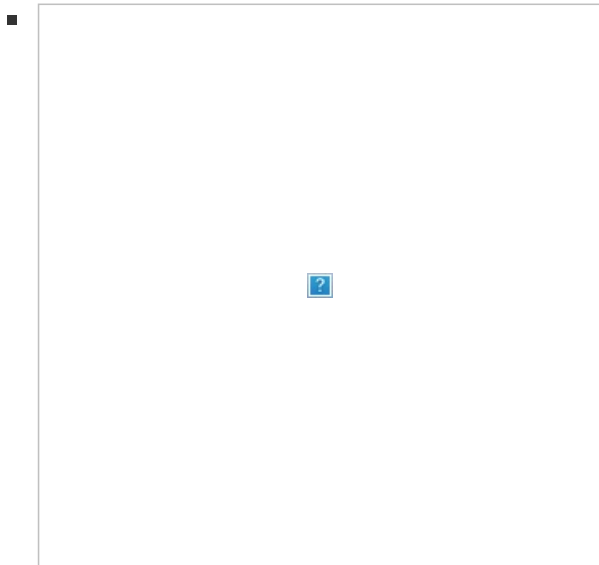


- <E>에서 E는 원하는 자료형을 넣으면 된다

- **Vector: List의 후손**

- ArrayList와 동등하지만 동기화 기능을 제공한다
 - 현재로는 잘 사용하지 않는 기술이다.

- **LinkedList:**

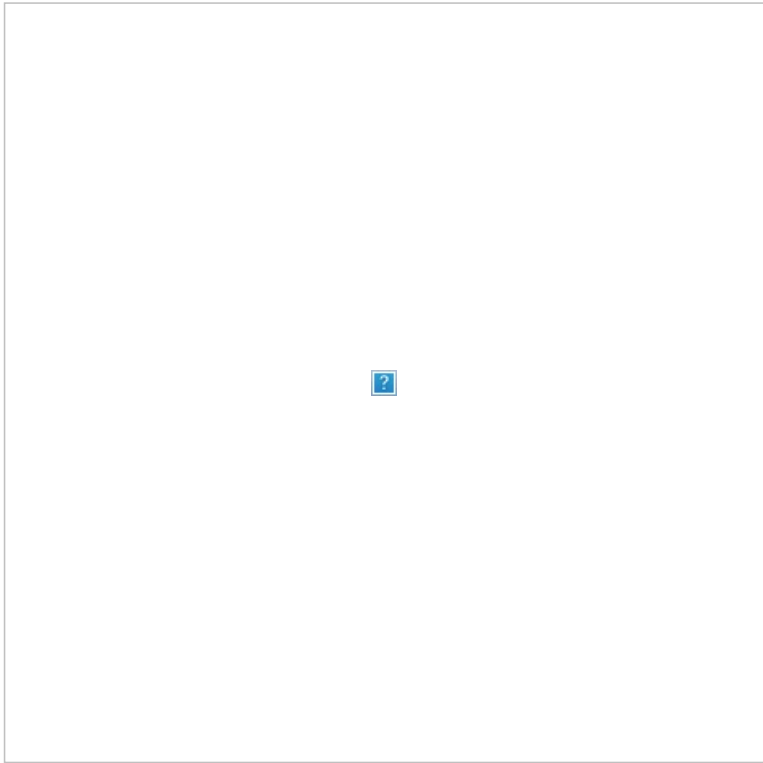



- LinkedList는 List의 후손으로 체인 처럼 연결되어져 있다. ArrayList처럼 배열 형태가 아니고 각각의 값들이 양옆의 위치 값만 기억하고 있다
 - 객체의 수정 삭제 조회가 쉽다

- **Set**

- 주머니와 같은 형태이다
 - 중복이 허용되지 않고
 - 순서대로 기억되지 않는다
 - 저장되는 객체들은 반드시 .equals() / .hashCode() 가 오버라이딩 되어 있어야 한다
 - set은 저장된 객체를 하나를 얻어오는 방법이 없어 모든 값을 반복하면서 원하는 값을 추출해내야 한다.
 - 이때 추출하는 방법은 Iterator을 사용한다

-



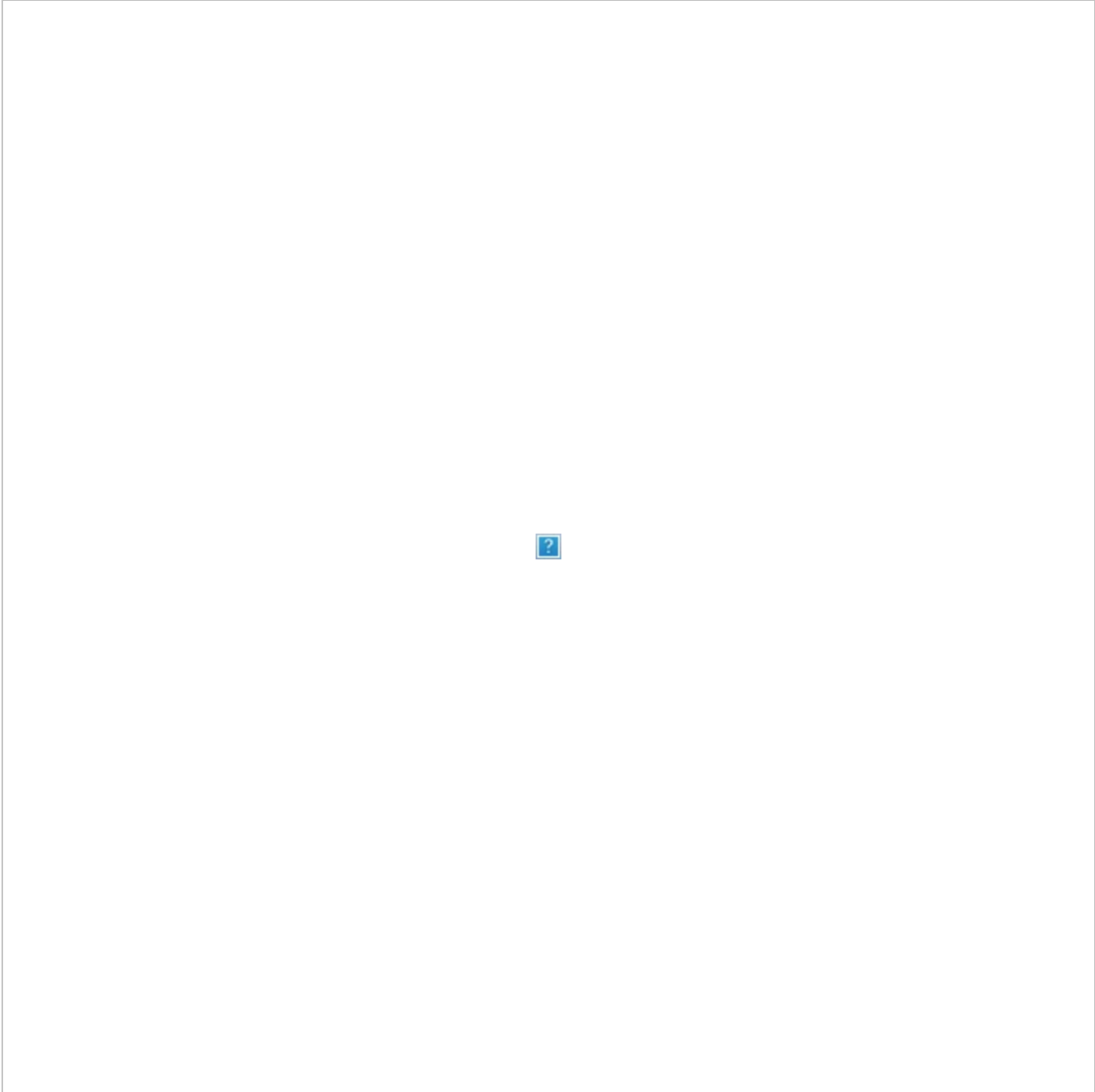
- **Iterator** 은 컬렉션에서 제공하는 반복자다
- **Set** 자식 객체들
 - **HashSet**
 - Set에 객체를 저장할때 Hash함수를 사용하여 처리 속도 가 빠르다.
 - 동일 객체와 동등 객체도 저장하지 않는다. (null도 중복 X)
 - **LinkedHashSet**
 - HashSet과 기능이 같지만 들어온 순서대로 값을 정렬 가능하다.
- **Map**
 -
 - List와 Set이 섞인 형태
 - Map<K, V>으로 구성 되어있다
 - 키 와 값으로 이루어져 있다 이때 key는 Set의 특징을 가지고 value는 List의 형태를 띈다 / 각각의 속성에 따라 key는 중복 X , value는 중복이 허용된다.
 - List와 set에서 add로 객체를 추가했지만 Map은 put으로 추가한다.
 - 
 - **Map의 후손들**
 - **HashMap**
 - 키 객체는 hashCode(), equals를 재정의해 동등 객체가 될 조건을 정해야 한다.
(이로 인해 보통 키 타입은 hashCode와 equals가 재정이 되어있는 String타입을 주로 사용)

기본 사용법

• equals(), hashCode() 오버라이딩

- equals(), hashCode()가 필요한 클래스 하위에 선언한다

-



- 이때 ALT + SHIFT+S 후 추가 할 수도 있다.

기본자료형 Wrapper Class으로 박싱 언박싱

- List , Set, Map 3가지 컬렉션 모두 '객체'만 저장 가능하다
- 이때 기본 자료형 8가지를 저장하기 위해 있는 클래스



- 컴파일러는 Integer가 정수인걸 인지하고 있기 때문에 . 정수로 자동적으로 **Auto UnBoxing**해준다
- 반대로 int형을 Integer 객체에 담을 때에도 컴파일러가 자동으로 **AutoBoxing**을 해준다
- **boolean : Boolean**
- **char : character**
- **byte : Byte**
- **short : Short**
- **int = Integer**
- **long = Long**
- **float : Float**
- **double = Double**

이처럼 기본 자료형의 첫 글자를 대문자로 해주면서 축약 단어일 경우 모든 단어를 표기한다.

향상된 for문

#배열도 사용 가능하다.

for(컬렉션의 형 변수명 : 컬렉션){

1. 오른쪽에는 원하는 배열 혹은 컬렉션 명을 적는다
2. 왼쪽에는 컬렉션 혹은 배열을 담을 변수를 생성한다(이때 배열 또는 컬렉션과 자료형이 같아야한다)
3. 이 for문은 컬렉션 혹은 배열의 0번부터 마지막 값까지 순차적으로 반복한다.

}

TreeSet /

이진 트리를 기반으로 한 **set**컬렉션으로 왼쪽과 오른쪽 자식 노트를 참조하기 위한 두개의 변수로 구성된다



TreeMap

이진 트리를 기반으로 한 **Map** 컬렉션으로 키와 값이 저장된 **Map.Entry**를 저장한다
왼쪽과 오른쪽 자식 노드를 참조하기 위한 두개의 변수로 구성된다.

