**-코드 실행시간 확인**

long beforeTime = System.currentTimeMillis(); //코드 실행 전에 시간 받아오기

long afterTime = System.currentTimeMillis(); // 코드 실행 후에 시간 받아오기

long secDiffTime = (afterTime - beforeTime)/1000; //두 시간에 차 계산 1000나누면 초 출력

System.out.println("시간차이(m) : "+secDiffTime);

**-코드 메모리 확인**

// Garbage Collection으로 메모리 초기화

System.gc();

// 실행전 메모리 사용량 조회

long before = Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory();

// Garbage Collection으로 메모리 정리

System.gc();

// 실행 후 메모리 사용량 조회

long after = Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory();

// 메모리 사용량 측정

long usedMemory = (before - after)/1024;

System.out.println("Used Memory : " + usedMemory); //mb 단위

**-강제 타입 가능여부 확인하기**

boolean result = 좌항(객체) instanceof 우항(타입)

좌항의 객체가 우항의 타입으로 생성되었다면 true, 아니면 false리턴

**-추상클래스 왜 쓰는가**

1. 공통된 필드와 메소드의 이름을 통일시킬 수 있다.

2. 실체 클래스를 작성할 때 시간 절약한다 -> 공통적인 필드와 메소드가 모두 추상클래스에 선언되어있으니깐

-선언법

public abstract class Phone {

public abstract void sound(); //추상메소드

} 부모

public class SmartPhone extends Phone {

@Override

public void sound() {} //추상메소드 재정의 오버라이드

} 자식 추상클래스 상속

**-인터페이스**

인터페이스 요소로는 상수 필드와 추상 메소드

인터페이스를 구현하는 클래스(구현클래스)는 각 클래스 입맛에 맞게 인터페이스의 추상메소드를 재정의한다.

인터페이스는 구현클래스의 설계를 제한하는 제약조건 혹은 설계도이다.

또한 여러 개의 인터페이스를 구현할 수 있다.

public interface RemoteControl {

//상수

public int MAX\_VOLUME = 10;

public int MIN\_VOLUMN = 0;

//추상 메소드

public void turnOn();

public void turnOff();

public void setVolume(int volume);

}

**-정적필드와 정적메소드**

정적멤버는 자바 힙영역에 저장되는 것이 아니어서 가비지머신이 프로그램 종료시 메모리 정리를 안해준다.정적영역에 등록돼서 객체를 만들지 않고도 클래스명.정적멤버명(메소드명)으로 바로 사용가능, 원주율 등

static int data1 = 10;

static void doFunc1() {}

**-final 필드**

초기값이 저장되면 최종적인 값이 되어서 프로그램 실행 중에는 수정할 수 없다.

**-상수 static final**

불변의 값

static final double EARTH\_RADIUS = 6400;

**-객체배열 만들기**

//Book 타입을 가지는 arr1 객체배열 생성

Book[] arr1 = new Book[3];

Book book1 = new Book("java", "hong", 1000);

Book book2 = new Book("zzz", "asdf", 2000);

Book book3 = new Book("jadf", "qwer", 3000);

//객체배열 안에 객체할당

arr1[0] = book1;

arr1[1] = book2;

arr1[2] = book3;

System.out.println("-----------------");

for(Book book: arr1) {

System.out.println(book.title);

System.out.println(book.author);

System.out.println(book.price);

}

**-super**

super연산자로 부모 클래스의 필드, 생성자, 메소드를 사용할 수 있다.

class Parent {

Parent(String data) {

System.out.println("Parent 생성자"+this);

}

}

class Child extends Parent {

Child() {

super("홍길동"); //여기

System.out.println("Child 생성자"+this);

}

}

**-객체 비교 equals()**

Object obj1 = new Object();

Object obj2 = new Object();

boolean result = obj1.equals(obj2);

**-비트연산 관련**

**A<<B (십진수 A를 이진수로 인식하고 왼쪽으로 B비트 만큼 민다.)**  
1 << 0 : 1을 왼쪽으로 0비트만큼 민다. = 1  
1 << 1 : 1을 왼쪽으로 1비트만큼 민다. = 10  
3 << 4 : 3을 왼쪽으로 4비트만큼 민다. = 11 -> 110000 = 2^4 + 2^5 = 48

**A >> B ( A를 오른쪽으로 B 비트만큼 민다. )**  
1 >> 0 : 1   
1 >> 1 : 0   
10 >> 1 : 1010(2) --> 0101(2) : 5 (2^2 + 2^0)  
10 >> 2 : 1010(2) --> 0010(2) : 2 (2^1)  
10 >> 3 : 1010(2) --> 0001(2) : 1 (2^0)

**N에서 x의 수가 포함되어 있는 지 검사하기**  
int n = 570;  
int x = 3;

**-문자를 숫자로 바꾸기 아스키코드(char to int)**

char c1 = ‘9’;

int i = c1 – ‘0’; //아스키코드 57에서 48을 빼면 9이고, 9의 아스키코드는 57이다.

char c1 = ‘9’;

Charcter.getNumericValue( c1 ); //9

-숫자말고 문자인 경우

char c = br.readLine().charAt(0);

System.out.println( (int)c );

**-문자열을 아스키코드로 바꾸기(String to int)**

String str = “QWER”;

for( int i=0; i<str.length(); i++ ) {  
 System.out.print( (int)str.toCharArray()[i] + “ “ );  
}  
//결과 81 87 69 82  
  
-아스키코드를 문자로 변환  
int a = 81;  
System.out.println( (char)81 );  
//Q

**-숫자를 그대로 문자열로 변환하기**int a = 10;  
String.valueOf( a );

**-문자열 변환 toString()**

Object 객체를 바로 만들어서 toString() 메소드를 출력하면 단순히 객체참조값이 출력되지만

String 객체를 만들어서 toString() 메소드를 사용하면 해당 문자열이 출력된다.

**-String클래스의 메소드**

-문자열 길이 확인 : str1.length();

-문자열 추출 : substring(시작인덱스, 끝인덱스)

String pstr1 = str1.substring(2, 7);

-문자열 검색 : indexOf("문자열"); 해당 문자열이 있는 곳의 시작하는 인덱스값 출력

int pos = str1.indexOf("Hello");

-뒤에서 문자열 검색 : lastIndexOf( "문자열" ); : 뒤에서부터 해당 문자열이 나오는 인덱스값 출력

-문자열 존재 검색

1. str1.startWith("문자열")

2. str1.endWith("문자열")

3. str1.contains("문자열")

-문자열 치환 : str1.replaceAll("world", "세상");

-문자열 합치기 : str1.concat("bye") str1에 bye붙이기

-대소문자 변환 : str1.toUpperCase() / str1.toLowerCase();

-문자열 좌우 공백제거 : str1.trim();

-문자열 분리 : str1.split( "," );

-분리된 문자열 결합 : String.join( ",", str1 );

**-문자열을 문자배열로 만들기**String str = “abcdefg”;  
for( char s : s.toCharArray() ) {  
 System.out.print( v+” “ ); //a b c d e f g  
}

**@@@ 수학 관련**

**-거듭 제곱 구하기**  
Math.pow( 5, 2 );  
>> 25

**-제곱근 구하기**  
Math.pow( 15, 0.5 );  
>> 3.52xxxxx  
정수만 구하고 싶으면 앞에 강제형변환시킨다.  
(int)Math.pow( 15, 0.5 );

**-나눗셈에서 소수점 x자리 까지 구하기**

int a = 10;  
int b = 3;  
String result = String.format( “%.2f”, a/(double)b ); //소수점 2번째 자리에서 반올림  
  
혹은 Math.round() 메서드를 이용한다.  
int result = Math.round(a\*100)/100; //소수점 2번째

**@@@StringBuilder**

**-선언**  
StringBuilder sb = new StringBuilder();

**-문자열 추가하기**  
sb.append( “문자열” );

**-문자열 뒤집기**  
System.out.println( sb.reverse() );

**@@@StringBuffer**

**-선언**  
StringBuffer sb = new Stringbuffer();

**-데이터 추가**  
sb.append( “문자열” );

**-내용지우기**  
sb.delete( 0, sb.length() );  
sb = null;  
sb.setLength( 0 );

**-==와 equals(), hashCode()**

==는 동일성 비교로 객체 인스턴스의 주소 값만 비교

equals()는 객체 내부의 값을 비교

hashCode()는 객체 내부의 내용을 hash값으로 바꿔서 hash값을 리턴해주는 메소드

같은 내용을 가진 문자열이라면 무조건 동일한 hash값을 가진다.

-객체비교를 할 때는 아래의 알고리즘을 수행한다.

1. hashCode() 리턴값 같으면 2번, 다르면 다른 객체

2. equlas() 리턴값 같으면 동등객체, 다르면 다른 객체

**-자바 타입확인**

변수명.getClass().getName();

Integer a = 5;

System.out.println( a instanceof Integer );

//true

**-Switch-case 문**

-문자 사용

String str = “가”;  
switch( str ) {  
 case “가” :  
 System.out.println( “str은 가 입니다.” );  
 break;  
 case “나”:  
 System.out.println( “str은 나 입니다.” );  
 break;  
 default: //모두 해당이 안되는 경우  
 System.out.println( “기타” );  
}

-숫자인경우

int i = 3;   
switch (i) {   
 case 1: // 1 인 경우   
 System.out.println("1입니다.");   
 break;   
 case 2: // 2 인 경우   
 System.out.println("2입니다.");  
 break;  
}

**-자바 삼항연산자**

int a = ( 5 > 4 ) ? 50 : 40;

**-forEach**

컬렉션객체의 요소를 순회하는 반복문

컬렉션객체.forEach( System.out::println );

**@@@ 배열관련 @@@**

**-배열의 선언**

-초기선언 int[] i\_array = new int[8];

int[] i\_array = { 1, 2, 3, 4, 5 };

-나중에 선언

int[] i\_array;

i\_array = new array[8];

**-배열의 길이**

배열명.length

**-배열 정렬하기** import java.util.Arrays;

Arrays.sort( 배열명 )  
내림차순  
 Arrays.sort( 배열명, Collections.reverseOrder() ); //이때 배열은 int가 아니라 Integer로 선언해야 한다.

**-배열 복사하기** import java.util.Arrays;

Arrays.copyOfRange( array, 시작인덱스, 끝인덱스 );

**-배열 최대값, 최소값 구하기**

import Arrays;

//arr이 배열

Arrays.stream( arr ).max().getAsInt();

Arrays.stream( arr ).min().getAsInt();

**-배열의 특정값으로 채우기**Arrays.fill( arr, 9999 );

**-배열 사용자 조건에 맞게 정렬하기**

arr 배열의 값들을 글자수, 같은 글자일 경우 사전순으로 정렬해보자. 이때 중복되는 문자는 출력하지 않는다.  
  
String[] arr = { “bbb”, “cc”, “a”, “ccc”, “aaa”, “bb”, “bbb” };  
Arrays.sort( arr, new Comparator<String> {  
 @Override  
 public int compare( String s1, String s2) {  
 if( s1.length() == s2.length() ) return s1.compareTo( s2 ); //사전순 비교  
 else return s1.length() – s2.length(); //글자수 비교  
 }  
});  
StringBuilder sb = new StringBuilder();  
sb.append( arr[0] ).append( ‘\n’ );  
for( int i=1; i<n; i++ ) {  
 if( !arr[i].equals( arr[i-1] ) ) { //중복되는 값 제거  
 sb.append( arr[i] ).append( ‘\n’ );  
 }  
}  
System.out.println( sb );

**-HashMap 선언법**

해시맵이란?

Map인터페이스의 한 종류로써 key와 value값으로 데이터를 저장하는 형태

map이란 키와 값을 하나의 쌍으로 묶어서 저장하는 컬렉션 클래스들을 구현할 때 사용

HashMap<String, Integer> hm = new HashMap<>();

-데이터추가

hm.put( "abc", 1 );

-데이터삭제

hm.remove( 1 ); //key값 1인 데이터 삭제

-모든 값 제거

hm.clear();

-값 출력( entrySet() 활용 )

for( Entry<String, Integer> entry : hm.entrySet() ) {

System.out.println( entry.getKey() + entry.getValue() );

}

-값 출력( KeySet() 활용 )

for( Integer i : hm.keySet() ) {

System.out.println( hm.get(i) );

}

-해시맵이름.getOrDefault( a, 0 ) : a값이 있으면 a값의 키값, 없으면 0

-해시맵 반복하기

int max\_value = 0;

int max\_ground = 0;

for( Integer i : hm.keySet() ) { //hm.values는 값들을 반복

if( max\_value < hm.get(i) ) {

max\_value = hm.get(i);

max\_ground = i;

}

}

아래코드는 키와 값을 모두 반복

Entry<Integer, Integer> maxEntry = null;

for( Entry<Integer, Integer> entry : hm.entrySet() ) {

if( maxEntry == null || entry.getValue() > maxEntry.getValue() ) {

maxEntry = entry;

}

}

**-찾는 키가 존재하면 키의 값반환, 없으면 기본값 반환하기 getOfDefault**

hm.put( key, hm.getOrDefault(key, 0)+1 )  
찾는 키가 있으면 그 키값에다가 +1한 상태로 put하고  
찾는 키가 없으면 해당 키에 값은 0+1의 값이 들어간다.

**-특정 키 또는 값 존재여부 확인하기**hm.containsKey( “A” );  
hm.containsValue( “AAA” );

**-ArrayList** import java.util.ArrayList;

-선언 ArrayList<Integer> list = new ArrayList<>();

-배열을 arraylist로 변환  
 이때 배열의 자료형과 ArrayList자료형이 맞아야한다.

ArrayList<Integer> rr = new ArrayList<Integer>( Arrays.asList(nrr) );

-arraylist를 배열로 변환

String[] strArr = (String[])list.toArray( new String[ list.size() ] );

-ArrayList 정렬하기 import java.util.Collections;

-내림차순: Collections.sort( list, new Descending() );

-오름차순: Collections.sort( list, new Ascending() );

-ArrayList 모든 값 제거 및 초기화

list.cleart();

-arraylist 삭제

list.remove(인덱스값)

**­-ArrayList 복사하려면 =을 쓰면 안되고 아래처럼 해야한다.**

ArrayList<String> names1 = new ArrayList<>();  
names1.add("Alan");  
names1.add("Alex");  
names1.add("Bob");  
  
ArrayList<String> names2 = new ArrayList<String>(names1);

2차원 arraylist

ArrayList<ArrayList<Integer>> datas = new ArrayList<ArrayList<Integer>>();

ArrayList<Integer> data1 = new ArrayList<>();

data1.add( 1 );

data1.add( 2 );

ArrayList<Integer> data2 = new ArrayList<>();

data2.add( 1 );

data2.add( 2 );

datas.add( data1 );

datas.add( data2 );

for(ArrayList<String> data: datas) {

System.out.println(data);

}

[1, 2, 3]

[11, 22, 33]

[111, 222, 333]

**2차원 ArrayList 정렬하기**

Comparator<ArrayList<Integer>> myCom = new Comparator<ArrayList<Integer>>() {

@Override

public int compare(ArrayList<Integer> o1, ArrayList<Integer> o2) {

if( o1.get(0) == o2.get(0) )

return o1.get(1) - o2.get(1);

else

return o1.get(0) - o2.get(0);

}

};

Collections.sort( Ss, myCom );

**-LinkedList**

Linkedlist는 각 노드가 데이터와 포인터를 가지고 한 줄로 연결되어 있는 방식의 자료구조이다.

데이터를 담고 있는 노드들이 연결되어 있고, 노드의 포인터가 이전 노드와 다음 노드와의 연결을 담당한다.

LinkedList<Integer> list = new LinkedList<Integer>(); //타입설정 int타입만 사용가능

//값추가

list.addFirst(1); //가장 앞에 데이터 추가

list.addLast(2); //가장 뒤에 데이터 추가

list.add(3); //데이터 추가

list.add(1, 10); //index 1 뒤에 데이터 10 추가

//값 삭제

list.removeFirst(); //가장 앞 데이터 제거

list.removeLast(); //가장 뒤 데이터 제거

list.remove(); //인덱스값 생략시 0번째 index제거

list.clear(); //모든 값 제거

//크기 구하기

list.size();

//값 출력하기

list.get( 0 ); //0번째 index의 값 출력

//전체값 출력

Iterator<Integer> iter = list.iterator(); //Iterator 선언

while(iter.hasNext()){ //다음값이 있는지 체크

System.out.println(iter.next()); //값 출력

}

//사용자 객체 담기

LinkedList<Student> list = new LinkedList<Student>();

Student student = new Student(name,age);

members.add(student);

members.add(new Member("홍길동",15));

**-HashSet: 중복이 없이 저장하기**

import java.util.HashSet;

import java.util.Iterator;

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

// HashSet<String> set = new HashSet<String>(10); //초기용량 지정

set.add("자바");

set.add("카푸치노");

set.add("에스프레소");

System.out.println("저장된 데이터의 수 : " + set.size());

Iterator<String> iterator = set.iterator();

while (iterator.hasNext()) {

String str = iterator.next();

System.out.println(str);

}

HashSet는 데이터에 순서가 없기 때문에 데이터를 순서대로 읽어오거나,

특정 위치의 데이터를 읽어올 수 있는 방법이 없기 때문에

"Iterator" 메소드를 활용해서 집합에 있는 "전체" 데이터를 불러올 수 있다.

-값 제거  
set.remove( “에스프레소” );

-값 검색  
System.out.println( set.contains( “카푸치노” ) ); //true

**-LinkedHashSet으로 입력된 순서대로 중복제거**

import java.util.LinkedHashSet;

String[] goodsList = {"notebook", "TV", "monitor", "keyboard", "mouse", "TV", "monitor", "keyboard"};

LinkedHashSet<String> linkedHashSet = new LinkedHashSet<>();

for(String item : goodsList){

linkedHashSet.add(item);

}

System.out.println(linkedHashSet);

//결과 = [notebook, TV, monitor, keyboard, mouse]

**-자바 입력받은 값 분리하기**

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.StringTokenizer;

BufferedReader br = new BufferedReader( new InputStreamReader(System.in) );

int T = Integer.parseInt( br.readLine() );

StringTokenizer st = new StringTokenizer( br.readLine(), " " );

ranks[j][0] = Integer.parseInt( st.nextToken() );

ranks[j][1] = Integer.parseInt( st.nextToken() );

br.close();

**-버퍼로 출력하기**

BufferedWriter bw = new BufferedWriter( new OutputStreamWriter(System.out));

bw.write( + "\n" );

bw.close();

**-1차원 배열 정렬하기**

import java.util.Arrays;

int[] scores = {899, 982, 1090, 982, 1018};

Arrays.sort(scores);

System.out.println(Arrays.toString(scores)); // [899, 982, 982, 1018, 1090]

**-2차원배열에서 1열의 값을 오름차순으로 정렬하기**

import java.util.Arrays;

Arrays.sort( ranks, new Comparator<int[]>() {

public int compare( int a[], int b[] ) {

if( a[0] == b[0] )

return a[1] - b[1];

else

return a[0] - b[0];

}

} );

**-comparable과 comparator**

Comparable : 기본 정렬기준을 구현하는데 사용

Comparator : 기본 정렬기준 외에 다른 기준으로 정렬하고자 할 때 사용

Comparable은 객체 간의 정렬에 있어서 오름차순, 내림차순등의

일반적인 순서를 잡는 기준이 필요할 때 객체 클래스에 확장해서 사용한다.

기본타입(Integer, String 등)객체 정렬말고 사용자정의 객체를 정렬할 때 사용

해당 인터페이스를 구현(implements)한 클래스는 반드시

compareTo(T o) 메소드를 정의해야 한다.

Arrays.sort()는 compareTo()메소드에서 반환되는 값에 따라, 자동으로 정렬을 해준다.

compareTo()

양수 : 현재객체 > 매개변수객체

0 : 현재객체 = 매개변수객체

음수 : 현재객체 < 매개변수 객체

comparator 익명클래스로 생성하기

Arrays.sort(strings, new Comparator<String>() {

@Override

public int compare(String s1, String s2) {

return s1.length() - s2.length(); //오름차순

}

});

**-while( 변수 --> 0 )**

1회 반복시 변수를 -1 하고 -1이되면 멈춘다.

int T = 3;

int cnt = 10;

while(T-- > 0) {

System.out.println(cnt--+" 루프 : "+T); //8 루프 0에서 끝난다.

}

**-람다식 : 익명함수**

함수의 이름이 없는 함수

비교 예시

int max(int a, int b) {

return a > b ? a : b;

}

(int a, int b) -> {

return a > b ? a : b;

}

(int a, int b) -> a > b ? a : b

**@@@Math클래스**

Math.abs( 변수명 ); //절대값 반환

Math.min( a, b ); //두 변수 중에서 최소값 반환

Math.max( a, b ); //두 변수 중에서 최대값 반환

**@@@stack**

Stack stack = new Stack();  
특정타입을 지정하고 싶다면  
Stack<Integer> stack = new Stack<Integer>();

stack.push( 0 ) //0을 스택에 넣음

stack.pop() //가장 최근에 넣은 값 빼기

stack.peek() //가장 위에있는 값 조회, 삭제는 안됨

stack.empty() //스택 비었는지 확인 비었으면 true, 아니면 false

stack.seach( 0 ) //매개변수 값의 위치 반환, 1

**@@@Queue**

Queue<Integer> que = new LinkedList<Integer>();

q.add( 3 );

// q.offer( 3 ) 도 똑같은 기능을 하지만 add의 경우 큐가 꽉찼을 때 에러를 발생시키고

offer의 경우 false를 리턴해준다.

q.add( 4 );

q.peek(); //가장 첫 번째값 3출력, 꺼내는거 아님

q.poll(); //가장 첫 번째값 꺼내기, 현재 peek값은 4

while( !q.isEmpty() ) {  
}

**-큐 원소 탐색하기**for( int a : q ) { }

**-큐에 두 개의 값 집어넣기**

Queue<Point> q = new LinkedList<Point>();

q.add( new Point( a, b ) );

int x = q.peek().x;

int y = q.peek().y;

**@@@dfs, bfs 기본순서**

-dfs

노드간의 연결정보있는 배열 선언 -> 노드 수에 맞는 방문처리 배열 선언 -> dfs함수선언(매개변수확인)

-> 현재 노드 방문처리 -> for반복문으로 해당 노드와 연결되어 있는 노드 반복 ->

방문처리 되지 않은 노드만 dfs 함수 재귀적 호출

-bfs

노드간의 연결정보있는 배열 선언 -> 노드 수에 맞는 방문처리 배열 선언 -> bfs함수선언(매개변수 확인)

-> 현재 노드 방문처리 -> 큐 선언 -> 큐에 현재 노드추가 -> while문 이용해서 큐가 비어질 때까지 반복

-> 현재 큐에 있는 노드값 다른 변수에 저장하고 빼내기 -> for문 이용해서 현재 노드와 연결되어 있는 노드반복

-> 연결되어 있는 노드 큐에 집어넣고 해당 노드 방문처리

**@@@ 조합과 순열**

**-순열**서로 다른 N개에서 중복을 허락하지 않고 R개를 일렬로 나열한다.

**-조합**서로 다른 N개에서 순서에 상관없이 R개를 뽑는 것