**﻿**자바 2 - 변수+형변환+연산자

final 상수 - 클래스, 함수, 변수를 고정적인 값으로 사용하고 싶다면 final로 선언하는 것이다.

협업시에는 상호관의 논의가 필요함. -> 상수는 한 번 저장된 값을 변경 불가능하다.

오버플로우 - 값의 범위가 자료형에서 허용된 크기를 넘을시에 발생하는 문제.

﻿-> 값의 범위를 수정 - 즉 자료형을 더 큰 자료형으로 변경해주면 해결.

증감 연산자 ++, --

비교 연산자 ==, <=, >=, <, >

대입 연산자 +=, -=, \*=, /=, %=

->대입 연산자는 메모리에 직접 연산 수행함 즉 처리 속도가 더 빠름.

​

% 는 조건문에서 짝수/홀수 판별에 유용하게 사용된다.

논리 연산자 &&(AND), ||(OR)

​

<

형변환>

정수는 큰 값

정수는 실수

문자는 정수로

-> 형변환 될 수 있다.

﻿

final 상수 - 클래스, 함수, 변수를 고정적인 값으로 사용하고 싶다면 final로 선언하는 것이다.

협업시에는 상호관의 논의가 필요함. -> 상수는 한 번 저장된 값을 변경 불가능하다.

오버플로우 - 값의 범위가 자료형에서 허용된 크기를 넘을시에 발생하는 문제.

-> 값의 범위를 수정 - 즉 자료형을 더 큰 자료형으로 변경해주면 해결.

증감 연산자 ++, --

비교 연산자 ==, <=, >=, <, >

대입 연산자 +=, -=, \*=, /=, %=

->대입 연산자는 메모리에 직접 연산 수행함 즉 처리 속도가 더 빠름.

텍스트 추가

publicvoidmyhome(){ int num1 =1; int num2 =2; System.out.println("num1 + num2 : "+(num1 + num2)); System.out.println("num1 - num2 : "+(num1 - num2)); System.out.println("num1 \* num2 : "+(num1 \* num2)); System.out.println("num1 / num2 : "+(num1 % num2)); System.out.println("num1 % num2 : "+(num1 / num2));}

텍스트 추가

% 는 조건문에서 짝수/홀수 판별에 유용하게 사용된다.

논리 연산자 &&(AND), ||(OR)

텍스트 추가

publicvoidmyhome(){//short cut 연산 int num =4; System.out.println("&& 연산 전 num 값: "+ num); boolean result1 =true&&++num >0; // num 값이 4임 and 연산은 두 두 값을 비교함. 그래서 ++num 때문에 num값 1 상승// 만일 앞에서 false를 통해 비교하면 Dead code 경고와 함께 ++num이 진행되지 않아 1값 상승X System.out.println("result1 : "+ result1); System.out.println("&& 연산 후의 num 값: "+ num); boolean result2 =false||++num <0;// || 연산자도 위의 설명과 마찬가지 System.out.println("result1 : "+ result1); System.out.println("&& 연산 후 num 값: "+ num);}

텍스트 추가

<형변환>

정수는 큰 값

정수는 실수

문자는 정수로

-> 형변환 될 수 있다.