1. 타입 변환.

자바에는 자동 타입 변환(묵시적), 강제 타입 변환(명시적) 두가지 타입 변한이 있다.

이번 시간에는 자동 타입 변환에 대해 배웠다.

먼저, 자동 타입 변환은 프로그램 실행 도중에 자동으로 타입변환이 일어나는 것으로 작은 크기를 가지는 타입이 큰 크기를 가진 타입에 저장될 때 발생한다.

<보기>

타입별 크기 순서

Byte<short<int<long<float<double

Ex1) byte byteVal = 10;

Int intVal = byteVal;

➡ 자동 타입 변환으로 byteVal 은 int 형으로 변환된다. (가지고 있는 값 10은 변하지 않는다.)

Ex2) char charVal = 'A';

Int intVal = charVal; //65 저장됨

➡ Char 타입이 int타입으로 변환되면 유니코드 값이 저장된다.

단, 음수가 저장될 수 있는 byte, int 등의 타입은 char 타입으로 자동 타입 변환할 수 없다.

Ex3) int intVal = 20;

double doubleVal = intVal; //20.0

➡ 실수 타입으로 변환되면 .0이 붙은 실수값이 된다.

다음은 강제 타입 변환이다. 강제 타입 변환은 큰 크기 타입은 작은 타입으로 자동 타입 변환을 할 수 없다. 하지만 강제로 int 타입의 1byte 를 잘라서 byte 타입 변수에 저장할 수 있다. (나머지 3byte는 버려지게 된다.)

Ex1) int intVal = 103029770; //~~00000110001001000000011100~~ **00001010 (10이 저장됨)**

byte byteVal; //강제 타입 변환

➡ 원래 값은 보존되지 못한다. (밑줄 친 비트들만 저장되기 때문이다.)

만약 intVal이 1byte 크기 내의 값을 가진다면, 값은 보존된다.

Ex2) int intVal = 'A'; //65저장

Char charVal = (char)intVal; //65에 해당되는 유니코드 문자가 저장된다.

System.out.println(charVal); //저장된 문자 출력

➡ 실수 타입(float,double)은 정수 타입으로 변환되지 않기 때문에 강제 타입 변환을 사용해야 한다.

소수점 이하 부분은 버려지고 (값 손실), 정수 부분만 저장된다.

2. 데이터 입력 받는 방법

사용자로부터 값을 입력 받는 방법은

Scanner 호출 > Scanner의 객체 생성 > Scanner의 메소드 의 순으로 진행된다.

먼저 `import java.util.Scanner;`

➡ Scanner을 사용하기 위해서는 import를 통해 호출한다. Java.util 패키지에 포함되어 있으므로

다음과 같이 import 한다.

두번째로 `Scanner sc = new Scanner(System.in);`

➡ sc라는 객체를 생성하여 System.in으로 입력한 값을 바이트 단위로 읽는 것을 의미한다.

세번째는 Scanner가 상황마다 다양한 메소드를 제공하는데 예시로 확인할 수 있다.

Ex)
```
import java.util.Scanner;

public class test {

        public static void main(String[] args) {
                String name;
                int age;
                double height;
                String intro;
                String buffer;

                Scanner sc = new Scanner(System.in);
                System.out.println("이름을 입력하세요");
                name = sc.next();
                System.out.println("나이를 입력하세요");
                age = sc.nextInt();
                System.out.println("키를 입력하세요");
                height = sc.nextDouble();
```

```
                System.out.println("자기소개를 입력하세요");
                buffer = sc.nextLine();
                intro = sc.nextLine();

                System.out.println("이름은 "+name+"나이는 "+age+",키는
"+height+"입니다.");
                System.out.println(intro);
        }
}
```

3. 연산자

연산자는 말그대로 연산(계산)을 하기 위해 사용되는 기호다.

종류	연산자	종류	연산자
최우선	( ),[ ]	논리 연산자	&,^,|,~,&&,||
단항 연산자	++,--,+,-,(type),!,~	삼항 연산자	?:
산술 연산자	%,/,*,+,-	대입 연산자	=,+=,-=,*=,%=,/=,>>=,<<=
시프트 연산자	>>,<<,>>>	콤마 연산자	,
관계 연산자	<,<=,>,>=,==,!=		

피 연산자의 개수에 따른 연산자의 종류

- 단항 연산자 : 피 연산자가 1개인 연산자
- 이항 연산자 : 피 연산자가 2개인 연산자
- 삼항 연산자 : 피 연산자가 3개인 연산자

!!참고 : 피 연산자란 연산에 참여하는 변수나 상수

4. 연산자 우선순위

기본적으로 연산자에는 우선순위가 있으며, 괄호의 우선순위가 제일 높고,

산술>비교>논리>대입의 순서이며**, 단항>이항>삼항**의 순서이다. 연산자의 연산 진행방향은 왼쪽에서 오른쪽으로 수행되며, 단항 연산자와 대입 연산자의 경우 오른쪽에서 왼쪽으로 수행된다.

우선순위	연산자	내용
1	() , []	괄호 / 대괄호
2	!, ~, ++, --	부정 / 증감 연산자
3	*, /, %	곱셈 / 나눗셈 연산자
4	+, -	덧셈 / 뺄셈 연산자
5	<<, >>, >>>	비트단위의 쉬프트 연산자
6	<, <=, >, >=	관계 연산자
7	==, !=	
8	&	비트단위의 논리연산자
9	^	
10	|	
11	&&	논리곱 연산자
12	||	논리합 연산자
13	? :	조건 연산자
14	=, +=, -=, *=, /=, %=, <<=, >>=, &=, ^=, ~=	대입 / 할당 연산자