

## 2장. 자료형 & 연산자



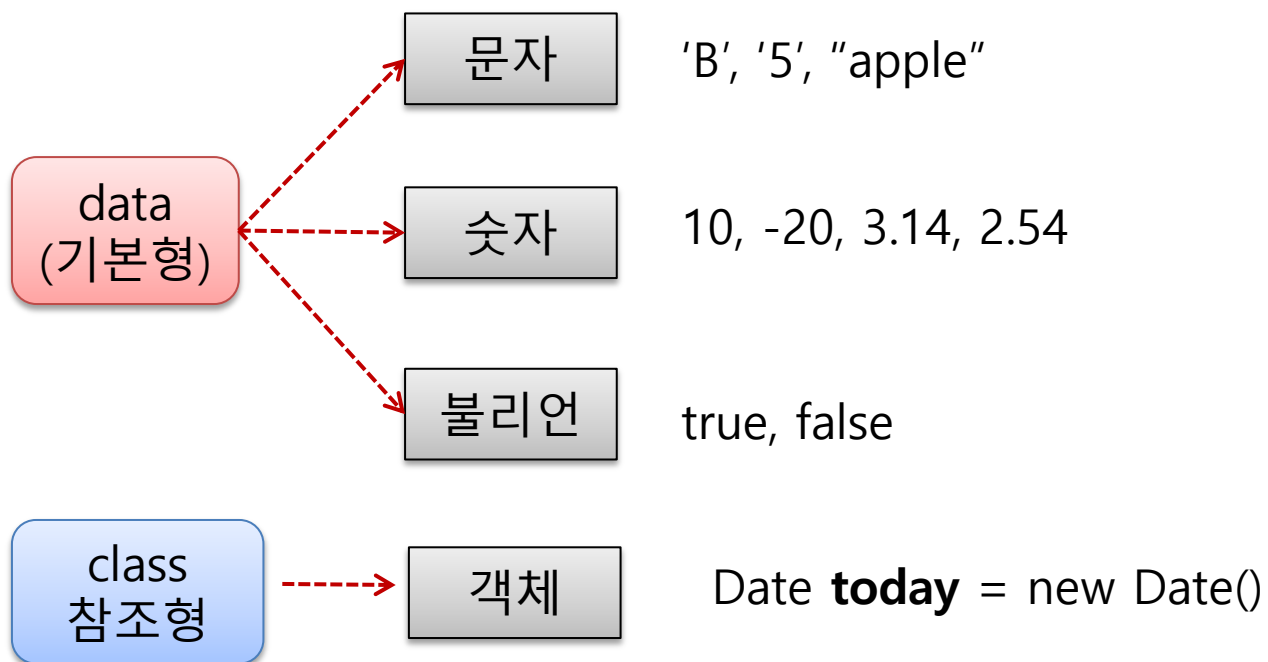
*Type & Operator*



# 자료형

## ● 자료형이란?

- 데이터를 저장하는 공간의 유형
- 사용할 데이터의 종류에 따라 메모리 공간을 적절하게 설정해 주는 것



- 기본 자료형의 크기

	정수형	문자형	실수형	논리형
1byte	byte	-	-	boolean
2byte	short	char	-	-
4byte	int	-	float	-
8byte	long	-	double	-



# 정수 자료형

- 정수 자료형의 종류 및 크기

자료형	바이트크기	수의 범위	설명
byte	1	$-2^7 \sim (2^7 - 1)$	1byte=8bit 1bit : 0, 1 - 2개 2bit : 00, 01, 10, 11 - 4개 3bit : 8개 ... 8bit : 256개
short	2	$-2^{15} \sim (2^{15} - 1)$	
int	4	$-2^{31} \sim (2^{31} - 1)$	
long	8	$-2^{63} \sim (2^{63} - 1)$	

- int로 5와 -5를 표현할 때(32비트)

5 -> 00000000000000000000000000000101



# 정수 자료형

## ▪ byte

- 1 바이트 단위의 자료형
- 동영상, 음악, 이미지 파일등 이진(바이너리) 실행 파일의 자료

## ▪ short

- 2바이트 단위의 자료형
- 주로 c/c++ 언어와의 호환 시 사용됨

## ▪ int

- 4바이트 단위의 자료형
- 프로그램에서 사용하는 모든 숫자(리터럴)은 기본적으로 int로 저장됨

## ▪ long

- 8바이트 자료형
- 가장 큰 정수 자료형으로 숫자 뒤에 'L' 또는 'l'을 써서 표시



# 숫자 자료형 실습

## ■ 정수 자료형

```
public class ByteShortType {  
    public static void main(String[] args) {  
        byte bData1 = -128;  
        System.out.println(bData1);  
  
        //byte bData2 = 128;    // -128 ~ 127  
  
        short sData1 = 32767;    //-32768 ~ 32767  
  
        //short sData2 = 32768;  
        System.out.println(sData1);  
    }  
}
```



# 숫자 자료형 실습

## ■ 정수 자료형

```
public class IntLongType {  
    public static void main(String[] args) {  
        int iNum = 1234567890;    //-21억 ~ 21억  
        System.out.println(iNum);  
  
        long lNum = 123456789012L;  //'L' or 'l'을 끝에 붙임  
        System.out.println(lNum);  
    }  
}
```



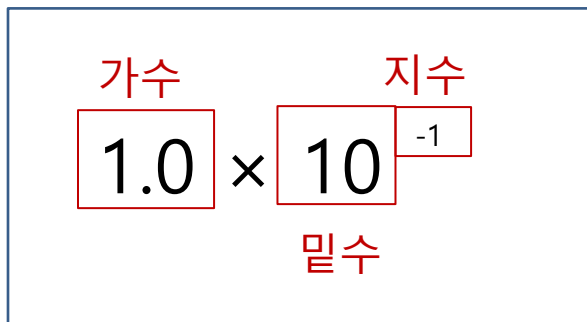
# 실수 자료형

## ■ 실수를 컴퓨터 내부에서 어떻게 나타내야 할지 생각해 보자

- 실수값 3.14를 표현하면 3이라는 정수부분과 .14라는 소수 부분을 따로 표현할 수 있고, 0과 1사이에는 무한개의 실수가 있다.
- 부동소수점 방식을 사용하면 실수를 좀 더 세밀하게 표현할 수 있다.

## ■ float 와 double

- 부동 소수점 방식 : 실수를 지수부와 가수부로 표현함  
무한의 실수를 표현하기 위한 방식
- 0.1을 표현하는 방식


$$\begin{array}{c} \text{가수} \qquad \qquad \text{지수} \\ \boxed{1.0} \times \boxed{10}^{\boxed{-1}} \\ \text{밑수} \end{array}$$

- float(4바이트) - 숫자 뒤에 F나 f를 써서 표시
- double(8바이트)  
예) `double num = 3.14;`  
`float num = 3.14F;`





# 숫자 자료형 실습

## ■ 실수 자료형

```
public class FloatDoubleType {  
    public static void main(String[] args) {  
        //실수 자료형  
        float fNum = 1.23456789F;  //'F' 나 'f'를 붙임  
        double dNum = 1.2345678901234567;  
  
        System.out.println(fNum);  
        System.out.println(dNum);  
    }  
}
```



# 문자 자료형

## ▪ char

- 자바에서는 문자를 2바이트로 처리
- 문자 1개를 표현할때 홑따옴표(' ')로 감싸준다.

## ▪ 문자 세트(charset)

- 문자세트 – 문자를 위한 코드 값(숫자 값) 들을 정해 놓은 세트
- 인코딩 – 각 문자에 따른 특정한 숫자 값(코드 값)을 부여
- 디코딩 – 숫자 값을 원래의 문자로 변환
- 아스키(ASCII) – 1 바이트로 영문자, 숫자, 특수 문자 등을 표현 함
- 유니코드(Unicode) – 한글과 같은 복잡한 언어를 표현하기 위한  
표준 인코딩 UTF-8, UTF-16이 대표적, 2바이트

<https://www.unicode.org/charts/PDF/UAC00.pdf>



# 문자 자료형

```
public class CharType {  
    public static void main(String[] args) {  
        //문자 자료형  
        char ch1 = 'A';  
        System.out.println(ch1);  
        System.out.println((int)ch1);  
  
        char ch2 = 66; 형변환  
        System.out.println(ch2);  
  
        int ch3 = 67;  
        System.out.println(ch3);  
        System.out.println((char)ch3);  
  
        System.out.println("\n***** 유니코드 *****");  
  
        char uniCode1 = '한';  
        System.out.println(uniCode1);  
  
        char uniCode2 = '\uD55C';  
        System.out.println(uniCode2);  
  
        for(char c = 97; c < 123; c++) {    //영어 소문자 출력(아스키코드)  
            System.out.print(c + " ");  
        }  
    }  
}
```



# 아스키 코드 vs 유니코드

## ★ 아스키 코드(ASCII Code)

아스키 코드는 미국 [ANSI](#)에서 표준화한 정보교환용 7비트 부호체계이다. 000(0x00)부터 127(0x7F)까지 총 128개의 부호가 사용된다. 이는 영문 키보드로 입력할 수 있는 모든 기호들이 할당되어 있는 부호 체계이며, 2바이트 이상의 코드를 표현할 수 없기 때문에 국제표준의 위상은 [유니코드](#)에게 넘어갔다.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	<b>Space</b>	64	40	100	&#64;	<b>@</b>	96	60	140	&#96;	<b>`</b>
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	<b>!</b>	65	41	101	&#65;	<b>A</b>	97	61	141	&#97;	<b>a</b>
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	<b>"</b>	66	42	102	&#66;	<b>B</b>	98	62	142	&#98;	<b>b</b>
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	<b>#</b>	67	43	103	&#67;	<b>C</b>	99	63	143	&#99;	<b>c</b>
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	<b>\$</b>	68	44	104	&#68;	<b>D</b>	100	64	144	&#100;	<b>d</b>
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	<b>%</b>	69	45	105	&#69;	<b>E</b>	101	65	145	&#101;	<b>e</b>
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	<b>&amp;</b>	70	46	106	&#70;	<b>F</b>	102	66	146	&#102;	<b>f</b>
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	<b>'</b>	71	47	107	&#71;	<b>G</b>	103	67	147	&#103;	<b>g</b>
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	<b>(</b>	72	48	110	&#72;	<b>H</b>	104	68	150	&#104;	<b>h</b>
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	<b>)</b>	73	49	111	&#73;	<b>I</b>	105	69	151	&#105;	<b>i</b>
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	<b>*</b>	74	4A	112	&#74;	<b>J</b>	106	6A	152	&#106;	<b>j</b>
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	<b>+</b>	75	4B	113	&#75;	<b>K</b>	107	6B	153	&#107;	<b>k</b>
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	<b>,</b>	76	4C	114	&#76;	<b>L</b>	108	6C	154	&#108;	<b>l</b>
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	<b>-</b>	77	4D	115	&#77;	<b>M</b>	109	6D	155	&#109;	<b>m</b>
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	<b>.</b>	78	4E	116	&#78;	<b>N</b>	110	6E	156	&#110;	<b>n</b>
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	<b>/</b>	79	4F	117	&#79;	<b>O</b>	111	6F	157	&#111;	<b>o</b>
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	<b>0</b>	80	50	120	&#80;	<b>P</b>	112	70	160	&#112;	<b>p</b>
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	<b>1</b>	81	51	121	&#81;	<b>Q</b>	113	71	161	&#113;	<b>q</b>
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	<b>2</b>	82	52	122	&#82;	<b>R</b>	114	72	162	&#114;	<b>r</b>
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	<b>3</b>	83	53	123	&#83;	<b>S</b>	115	73	163	&#115;	<b>s</b>
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	<b>4</b>	84	54	124	&#84;	<b>T</b>	116	74	164	&#116;	<b>t</b>
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	<b>5</b>	85	55	125	&#85;	<b>U</b>	117	75	165	&#117;	<b>u</b>



# 아스키 코드 vs 유니코드

## ★ 유니 코드(Uni code)

전 세계의 모든 문자를 다루도록 설계된 표준 문자 전산 처리 방식. 주요 구성 요소는 ISO/IEC 10646 Universal Character Set과 UCS, UTF 등의 인코딩 방식, 문자 처리 알고리즘 등이다. 유니코드를 사용하면 한글과 간체자, 아랍 문자 등을 통일된 환경에서 깨뜨리지 않고 사용할 수 있다.

초창기에는 문자 코드는 ASCII의 로마자 위주 코드였고, 1바이트의 남은 공간에 각 나라가 자국 문자를 할당하였었다.

하지만 이런 상황에서 다른 국가에 이메일을 보냈더니 글자가 와장창 깨졌던 것. 이에 따라 2~3바이트의 넉넉한 공간에 세상의 모든 문자를 할당한 결과물이 이것이다.

흔히 우리가 웹 브라우저의 인코딩을 설정하면서 자주 보는 UTF-8이라는 말이 이것이고, 바로 유니코드에 기반한 인코딩 방식 중 하나를 가리키는 것이다



# 문자열 자료형

## ▪ String

- 문자열을 사용할 때는 String 자료형을 사용한다.
- 값은 쌍따옴표("")로 감싸준다.

```
public class StringType {  
    public static void main(String[] args) {  
        //문자열 자료형  
        String s = "k"; //문자열일 경우 "" 붙임.  
  
        String name = "Ja" + "va";  
        String str = name + 8.0;  
  
        System.out.println(s);  
        System.out.println(name);  
        System.out.println(str);  
        System.out.println(3 + "4");  
    }  
}
```



# 논리형

## ■ boolean

- 논리값 true(참), false(거짓)을 표현하는 자료형
- 프로그램 수행이 잘되었는지 여부, 값이 존재하는지 여부 등
- boolean으로 선언     예) **boolean** isMerried = true

```
// 회원 정보
String name = "추신수";
int age = 38;
boolean isMerried = true;
int numberOfChildren = 3;

System.out.println("이름 : " + name);
System.out.println("나이 : " + age);
System.out.println("결혼 유무 : " + isMerried);
System.out.println("자녀수 : " + numberOfChildren);
```

Member.java

이름 : 추신수  
나이 : 38  
결혼 유무 : true  
자녀수 : 3



# 형 변환(Type Conversion)

## ■ 형 변환

- 자료형은 각각 사용하는 메모리 크기와 방식이 다름
- 정수와 실수를 더할때 하나의 자료형으로 통일한 후 연산을 해야함.

### 묵시적 형변환

- 작은 자료형에서 큰 자료형으로 변환

```
int iNum = 20;  
float fNum = iNum;
```

- 연산 중 자동변환

```
double dNum = fNum + iNum
```

### 명시적 형변환

- 큰 자료형에서 작은 자료형으로 변환  
변환 자료형을 명시 : ( ) 괄호 사용

```
double dNum = 12.34;  
int iNum = (int)dNum;
```





# 형 변환(Type Conversion)

```
public class TypeConversion {  
    public static void main(String[] args) {  
        //묵시적 형 변환  
        int iNum = 20;  
        float fNum = iNum;  
        System.out.println(iNum);    //20  
        System.out.println(fNum);    //20.0  
  
        double dNum;  
        dNum = iNum + fNum;  
        System.out.println(dNum);    //40.0  
  
        //명시적 형변환  
        double dNum1 = 1.2;  
        float fNum2 = 0.9F;  
  
        int iNum3 = (int)dNum1 + (int)fNum2;  
        int iNum4 = (int)(dNum1 + fNum2);  
        System.out.println(iNum3);    //1  
        System.out.println(iNum4);    //2  
    }  
}
```



# 형 변환(Type Conversion)

```
public class TypeConversion2 {  
    public static void main(String[] args) {  
        //사칙연산  
        int x = 10, y = 20;  
  
        System.out.println(x + y);  
        System.out.println(x - y);  
        System.out.println(x * y);  
        System.out.println(x / y);  
  
        System.out.println(x / (double) y);  
    }  
}
```



# 상수(Constant)

## ■ 상수(constant)

- 상수 : 변하지 않는 값(1년은 12개월, 원주율은 3.14 등)
- 상수 선언하기 : **final** 키워드 사용, 대문자 사용

```
final double PI = 3.14;
```

```
final int MAX_NUM = 100;
```

final로 선언된 상수는 다른 값을 대입할 수 없음

```
PI = 3.15 //에러
```



# 상수와 리터럴

## ■ 상수 실습 예제

```
public class ConstantEx {  
    public static void main(String[] args) {  
        //상수로 선언하기  
        final int MAX_NUM = 100;  
        final int MIN_NUM;  
  
        MIN_NUM = 0;  
        //MAX_NUM = 1000;  //final로 선언하면 변경할 수 없음  
  
        System.out.println(MAX_NUM);  
        System.out.println(MIN_NUM);  
  
        // 원의 넓이 구하기(반지름*반지름*3.14)  
        final double PI = 3.14;  
        int radius = 5;  
        double area = PI * radius * radius;  
        System.out.println("원의 넓이는 " + area + "입니다.");  
    }  
}
```



# 컴퓨터에서 데이터 표현하기

- 컴퓨터는 0과 1로만 데이터를 저장함(0-> 신호꺼짐, 1-> 신호켜짐)
- Bit(비트) : 컴퓨터가 표현하는 데이터의 최소 단위로 2진수 하나의 값을 저장할 수 있는 메모리의 크기
- 숫자, 문자 표현 – 컴퓨터 내부에서는 숫자뿐만 아니라 문자도 2진수로 표현  
예) 'A'는 65로 정함(아스키코드) → 이진수로 01000001, 한글 'ㄱ'은 12593(유니코드)

10진수	2진수
0	00000000
1	00000001
2	00000010
3	00000011
...	...
9	00001001
10	00001010

문자	코드값	2진수
A	65	01000001
B	66	01000010
C	67	01000011
...	...	...
0	48	00110000
1	49	00110001
2	50	00110010



# 컴퓨터에서 데이터 표현하기

## ■ 비트로 표현할 수 있는 수의 범위

비트수	표현할 수 있는 범위(십진수)	
1bit	0, 1(0~1)	$2^1$
2bit	00, 01, 10, 11(0~3)	$2^2$
3bit	000, 001, 010, 011, 100, 101, 110, 111(0~7)	$2^3$

16진수	2진수
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111
10	10000

※ 10진수를 2진수로 바꾸기

가중치 방식

$$10 = 1010_{(2)}$$

8 4 2 1

$$1\ 0\ 1\ 0 (1 \times 2^3 + 1 \times 2^1 \rightarrow 8 + 2)$$

자리 올림 발생



# 이진수, 16진수 표기

```
public class BinHexPrint {  
    public static void main(String[] args) {  
        //2진수, 16진수 표기  
        int num = 10;  
        int bNum = 0b1010;  
        int hNum = 0xA;  
  
        System.out.println(num);    //10  
        System.out.println(bNum);   //10  
        System.out.println(hNum);   //10  
    }  
}
```



# 부호 있는 수를 표현하는 방법

## ● 음의 정수는 어떻게 표현할까?

- 정수의 가장 왼쪽에 존재하는 비트는 부호비트입니다.  
MSB(Most Significant Bit) 가장 의미있는 비트라는 뜻
- 음수를 만드는 방법은 2의 보수(1의 보수에 1을 더함)를 취한다.
- 1의 보수 표기법 - 0을 1로, 1을 0으로 표기

두 수를 더하면 0이 됨  
~~1~~00000000  
맨앞 1은 제거됨

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

10진수 5

1의 보수를 취한다.

1	1	1	1	1	0	1	0
---	---	---	---	---	---	---	---

1을 더한다.

+ 1

1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

10진수 -5





# 부호 있는 수를 표현하는 방법

```
public class BinaryTest {  
  
    public static void main(String[] args) {  
        //2진수를 음의 정수로 만드는 방법 - 2의 보수 법  
        int num1 = 0b00000000000000000000000000000000101;  
        int num2 = 0b11111111111111111111111111111111011; //보수에 1을 더함  
  
        System.out.println(num1); //5  
        System.out.println(num2); //-5  
  
        int sum = num1 + num2;  
        System.out.println(sum);  
    }  
}
```



# 항과 연산자

## ■ 항(operand)

- 연산에 사용되는 값

## ■ 연산자(operator)

- 연산에 사용되는 기호

예)  $3 + 7$  (3과 7은 항, '+'는 연산자)



## ■ 항의 개수에 따른 연산자 구분

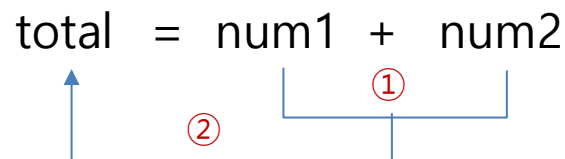
연산자	설명	연산 예
단항 연산자	항이 한 개인 연산자	$++num$
이항 연산자	항이 두 개인 연산자	$num1 + num2$
삼항 연산자	항이 세 개인 연산자	$(5 > 3) ? 1 : 0$



# 대입 및 부호 연산자

## ■ 대입 연산자

- 변수에 값을 대입하는 연산자
- 연산의 결과를 변수에 대입
- 왼쪽 변수(lvalue)에 오른쪽 값(rvalue)를 대입



## ■ 부호 연산자

- 양수/음수의 표현, 값의 부호를 변경
- 변수에 +, -를 사용한다고 해서 변수의 값이 변하는 것이 아님.
- 변수의 값을 변경하려면 대입연산자를 사용해야함

```
int num = 10;
System.out.println(num);
System.out.println(-num); //부호만 바뀜
System.out.println(num);

num = -num; //값이 바뀜
System.out.println(num);
```



# 대입 연산자

## ■ 실습 예제

-----

아래의 실행 결과대로 변수의 값을 서로 바꾸는 프로그램을 작성 하세요  
(파일 이름 : swap.java)

-----

👉 실행 결과

```
교환 전
x = 0, y = 1
-----
교환 후
x = 1, y = 0
```



# 산술 연산자

## ■ 산술 연산자

연산자	기 능	연산 예
+	두 항을 더합니다.	$5+3$
-	앞 항에서 뒤 항을 뺍니다.	$5-3$
*	두 항을 곱합니다.	$5*3$
/	앞 항에서 뒤 항을 나누어 몫을 구합니다.	$5/3$
%	앞 항에서 뒤 항을 나누어 나머지를 구합니다.	$5\%3$



# 증감 연산자

## ■ 증가 감소 연산자

- 1만큼 더하거나 1만큼 뺄 때 사용하는 연산자

연산자	기 능	연산 예
++	항의 값에 1을 더합니다.	<code>val = ++num; // num = num+1; val=num</code> <code>val = num++; //val=num; num=num+1;</code>
--	항의 값에서 1을 뺍니다.	<code>val = --num // num = num+1; val=num</code> <code>val = num--; //val=num; num=num+1;</code>



# 산술 연산자 연습문제

```
public class OperationEX2 {  
    public static void main(String[] args) {  
        //증가, 감소 연산자  
        int num = 10;  
        int val = 0;  
  
        //val = num++; //val = num 이후 num = num + 1;  
        val = ++num;    //num = num + 1 이후 val = num  
        System.out.println(val);  
        System.out.println(num);  
  
        //val = num--; //val = num 이후 num = num - 1;  
        val = --num;    //num = num - 1 이후 val = num  
        System.out.println(val);  
        System.out.println(num);  
  
        //산술 연산자  
        int mathScore = 90;  
        int engScore = 75;  
  
        int totalScore = mathScore + engScore;  
        System.out.println(totalScore);  
  
        double avgScore = (double)totalScore / 2;  
        System.out.println(avgScore);  
    }  
}
```



# 관계(비교) 연산자

## ■ 관계(비교) 연산자

연산자	기 능	연산 예
>	왼쪽 항이 크면 참을, 아니면 거짓을 반환합니다.	num > 3;
<	왼쪽 항이 작으면 참, 아니면 거짓을 반환합니다.	num < 3;
>=	왼쪽 항이 크거나 같으면 참, 아니면 거짓을 반환합니다.	num >= 3;
<=	왼쪽 항이 작거나 같으면 참, 아니면 거짓을 반환합니다.	num <= 3;
==	두 개의 항 값이 같으면 참, 아니면 거짓을 반환합니다.	num == 3;
!=	두 개의 항 값이 다르면 참, 아니면 거짓을 반환합니다.	num != 3





# 논리 연산자

## ■ 논리 연산자

연산자	기 능	연산 예
&&	두 항이 모두 참인 경우에만 결과 값이 참 입니다.	<code>boolean = (7&lt;3) &amp;&amp; (5&gt;2)</code>
	두 항중 하나의 항만 참이면 결과 값이 참 입니다.	<code>boolean = (7&lt;3)    (5&gt;2)</code>
!	단항 연산자, 참인 경우는 거짓으로, 거짓인 경우는 참으로 바꿉니다.	<code>boolean = !(7&gt;3)</code>



# 논리 연산자

```
public class OperationEx3 {  
    public static void main(String[] args) {  
        //관계(비교) 연산자  
        System.out.println(7 < 3);  
        System.out.println(7 > 3);  
        System.out.println(7 == 3);  
        System.out.println(7 != 3);  
  
        //논리 연산자  
        System.out.println((7 > 3) && (5 > 2));  
        System.out.println((7 > 3) || (5 < 2)); //단락 회로  
        System.out.println(!(7 > 3));  
    }  
}
```



# 논리 연산자

## ■ 논리 연산자 – 단락 회로

- 앞 조건이 거짓이면 뒤 조건은 연산하지 않음( &&인 경우)
- 앞 조건이 참이면 뒤 조건은 연산하지 않음(||인 경우)

```
public class LogicalOperator {  
    public static void main(String[] args) {  
        //단락회로 평가 실습  
        int n = 10;  
        int i = 2;  
        boolean value = ((n = n + 10) < 10) && ((i = i + 2) < 10);  
        System.out.println(value); //false  
        System.out.println(n);    //20  
        System.out.println(i);    //2  
  
        System.out.println("-----");  
        value = ((n = n + 10) > 10) || ((i = i + 2) < 10);  
        System.out.println(value); //true  
        System.out.println(n);    //30  
        System.out.println(i);    //2  
    }  
}
```



# 복합대입 연산자

## ■ 복합대입 연산자

연산자	기 능	연산 예
+=	두 항의 값을 더해서 왼쪽 항에 대입합니다.	num += 2; num=num+2와 같음
-=	왼쪽 항에서 오른쪽 항을 빼서 그 값을 왼쪽 항에 대입합니다.	num -= 2; num=num-2와 같음
*=	두 항의 값을 곱해서 왼쪽 항에 대입합니다.	num *= 2; num=num*2와 같음
/=	왼쪽 항을 오른쪽 항으로 나누어 그 몫을 왼쪽 항에 대입합니다.	num /= 2; num=num/2와 같음
%=	왼쪽 항을 오른쪽 항으로 나누어 그 나머지를 왼쪽 항에 대입합니다.	num %= 2; num=num%2와 같음



# 조건 연산자

## ■ 조건 연산자

삼항 연산자 -> 제어문중 조건문을 간단히 표현할 때 사용할 수 있음

연산자	기 능	연산 예
조건식?결과1:결과2;	조건식이 참이면 결과1, 조건식이 거짓이면 결과2가 선택됩니다.	int num = (5>3)?10:20;



# 조건 연산자

```
public class OperationEx4 {  
  
    public static void main(String[] args) {  
        //복합대입 연산자  
        int num = 10;  
        System.out.println(num += 1);  
        System.out.println(num %=10);  
        num -= 1;  
        System.out.println(num);  
  
        //조건 연산자  
        System.out.println("-----");  
        //부모님의 나이 비교  
        boolean bool = (5 > 3) ? true : false;  
        System.out.println(bool);  
  
        int fatherAge = 45;  
        int motherAge = 47;  
  
        char ch;  
        ch = (fatherAge > motherAge) ? 'T' : 'F';  
        System.out.println(ch);  
    }  
}
```



# 비트 연산자

## ■ 비트 연산자

연산자	기 능	연산 예
~	비트의 반전(1의 보수)	$a = \sim a$
&	비트 단위 AND	$1 \& 1 \rightarrow 1$ 을 반환, 그 외는 0
	비트 단위 OR	$0   0 \rightarrow 0$ 을 반환, 그 외는 1
<<	왼쪽 shift	$a \ll 2$ 변수 $a$ 를 2비트 만큼 왼쪽으로 이동
>>	오른쪽 shift	$a \ll 2$ 변수 $a$ 를 2비트 만큼 오른쪽으로 이동



# 비트 연산자

## ■ 비트 논리연산자

```
int num1 = 5;  
int num2 = 10;  
int result = num1 &  
num2;
```



```
num1 : 0 0 0 0 0 1 0 1  
& num2 : 0 0 0 0 1 0 1 0  
-----  
result : 0 0 0 0 0 0 0 0
```

## ■ 비트 이동 연산자

왼쪽으로 2자리 이동

```
int num = 5;  
num << 2;
```



```
num      : 0 0 0 0 0 1 0 1  
num << 2 : 0 0 1 0 1 0 0 0
```





# 비트 연산자

```
public class OperationEx5 {  
    public static void main(String[] args) {  
        //비트 논리 연산자  
        int num1 = 5;    //00000101  
        int num2 = 10;   //00001010  
        int result = num1 & num2; //00000000  
        System.out.println(result);  
  
        result = num1 | num2;    //00001111  
        System.out.println(result);  
  
        //비트 이동 연산자  
        int val = 2;    //00000010 -> 10진수 2  
        System.out.println(val << 1);    //00000100 -> 10진수 4  
        System.out.println(val << 2);    //00001000 -> 10진수 8  
        System.out.println(val);  
        System.out.println(val >> 1);    //00000001 -> 10진수 1  
    }  
}
```



# 연산자 우선 순위

## ■ 연산자 우선 순위

위쪽과 왼쪽이 우선 순위가 높음

우선순위	형태	연산자
1	일차식	( ) [ ]
2	단항	++ -- !
3	산술	% * / + -
4	비트이동	<< >>
5	관계	< > == !=
6	비트 논리	&   ~
7	논리	&&    !
8	조건	? :
9	대입	= += -= *=



# 입력 처리 – Scanner 클래스

## 화면에서 입력하기 - Scanner 클래스

- 문자, 숫자등의 자료를 표준 입력으로부터 읽어 들일 수 있다.
- Java.util 패키지에 속해있어서 import해야 한다.
- 종료시에 close() 메서드를 사용한다. -> scanner.close()

```
Scanner scanner = new Scanner(System.in)
```

메서드	설명
int nextInt()	int 자료형을 읽습니다.
double nextDouble()	double 자료형을 읽습니다.
String next()	문자열 String을 읽습니다.(\\n 버퍼에 남음)
String nextLine()	문자열 String을 읽습니다.(\\n을 포함)



## ■ Java Document -> Scanner 클래스 찾기

프로그램에서 데이터를 주고받기 위한 방법 등을 기술한 문서

**Module** java.base

**Package** java.util

**Class** Scanner

java.lang.Object  
java.util.Scanner

All Implemented Interfaces:

Closeable, AutoCloseable, Iterator<String>

```
public final class Scanner
extends Object
implements Iterator<String>, Closeable
```

A simple text scanner which can parse primitive types and strings using regular expressions.

A `Scanner` breaks its input into tokens using a delimiter pattern, which by default matches whitespace.

For example, this code allows a user to read a number from `System.in`:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```



# 입력 처리 – Scanner 클래스

- Scanner 클래스 사용하기

```
import java.util.Scanner;
```

```
public class ScannerEx {
```

```
    public static void main(String[] args) {
```

```
        //Scanner 클래스 사용
```

```
        Scanner scanner = new Scanner(System.in);
```

← Scanner 객체 생성

```
        System.out.print("당신의 이름은 무엇입니까? ");
```

```
        String name = scanner.nextLine();
```

← nextLine()로 이름입력

```
        System.out.println("당신의 이름은 " + name + "이군요.");
```

```
        System.out.print("당신의 나이는 몇 살입니까? ");
```

```
        int age = scanner.nextInt();
```

← nextInt()로 나이입력

```
        System.out.println("당신의 나이는 " + age + "세 이군요.");
```

```
        scanner.close();
```

← close()로 종료한다.

```
    }
```

```
}
```



# 구매 포인트 계산 프로그램

## ● 고객의 구매 포인트 계산 프로그램

```
public class BonusPoint {  
  
    public static void main(String[] args) {  
        Scanner scan = new Scanner(System.in);  
  
        System.out.print("고객의 이름을 입력하세요 : ");  
        String name = scan.nextLine(); //고객 이름  
  
        System.out.print("구매 금액을 입력하세요 : ");  
        int price = scan.nextInt(); //가격  
        int bonusPoint = 0;          //보너스 포인트  
        double bonusRatio = 0.05;    //보너스적립율 : 5%  
  
        //보너스포인트 = 가격 x 보너스적립율  
        bonusPoint = (int)(price * bonusRatio);  
        System.out.println(name + " 님의 보너스 포인트는 " + bonusPoint + "점입니다. ");  
  
        scan.close();  
    }  
}
```

고객의 이름을 입력하세요 : 김검소  
구매 금액을 입력하세요 : 10000  
김검소님의 구매 포인트는 500점입니다.



# 속도 KM를 마일로 변환하는 프로그램



메이저리그는 점점 더 빠른 구속을 추구하고 있다. 올 시즌 메이저리그 포심 패스트볼 평균 구속은 시속 93.2마일(150.0km)에 달한다. 이제는 100마일(160.9km)이 넘는 공도 어렵지 않게 볼 수 있게 됐다.

투수에게 있어 구속이 가장 중요한 요소는 아니다. 구종, 제구, 구위 등 수 많은 요소들이 어우러져야 비로소 뛰어난 투구를 할 수 있다. 하지만 같은 조건이라면 당연히 구속이 빠를수록 유리하다. 구속이 빠를수록 타자들이 공에 대처할 수 있는 물리적인 시간이 줄어들기 때문이다.



# 속도 KM를 마일로 변환하는 프로그램

## ◆ 속도 KM를 마일로 바꾸는 프로그램

```
public class KmToMile {  
    public static void main(String[] args) {  
        Scanner scan = new Scanner(System.in);  
  
        final double RATE_KPH_MPH = 1.609344; //변환 상수  
        double kph = 0.0; //km/h  
        double mph = 0.0; //mile/h  
  
        System.out.print("당신의 구속을 입력하세요(km/h) : ");  
        kph = scan.nextDouble();  
  
        //mile = km / 변환 상수  
        mph = kph / RATE_KPH_MPH;  
  
        //printf("문자열 포맷", 객체)  
        System.out.printf("공의 속도는 %.2f[MPH]입니다.", mph);  
        scan.close();  
    }  
}
```

당신의 구속을 입력하세요(km/h) : 140.5  
공의 속도는 87.30[MPH]입니다.





# 입력 처리 – 연습 예제

- 실습 예제

숫자를 입력받아 홀수/짝수를 판별하는 프로그램을 작성하세요  
(조건 연산자 활용 – 예)

☞ 실행 결과

```
숫자 입력 : 11  
홀수
```

