

2장. 변수, 자료형, 연산자



Variable, Operator

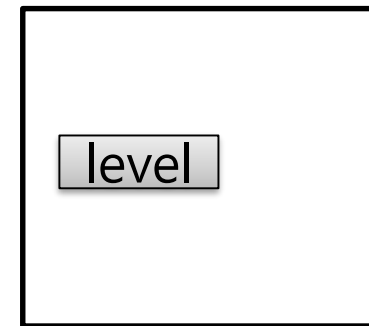


■ 변수란?

- 프로그램에서 사용되는 자료를 저장하기 위한 공간
- 할당받은 메모리의 주소 대신 부르는 이름
- 프로그램 실행 중에 값 변경 가능, variable 이라 함

■ 변수의 선언 및 초기화

- 변수 선언은 어떤 타입의 데이터를 저장할 것인지 그리고 변수이름은 무엇인지를 결정한다.
- 자료형 변수이름;
- 자료형 변수이름 = 초기값;
`int level;`
`double height;`



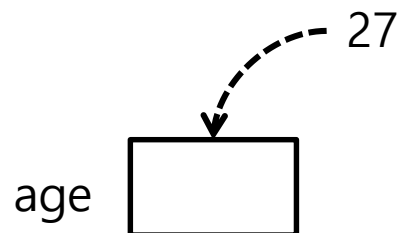
변수 사용하기

■ 변수의 초기화

```
int age = 27;
```

```
char c = 'k';
```

```
String fruit= "사과"
```



■ 변수 이름 선언시 유의점

- 변수의 이름은 알파벳, 숫자, _, \$로 구성된다.
- 대소문자를 구분한다.
- 숫자로 시작할 수 없고, 키워드(예약어)도 변수의 이름으로 사용할 수 없다.(문법 오류)
- 이름 사이에 공백이 있을 수 없다.(문법 오류)
- 변수의 이름을 정할 때는 변수의 역할에 어울리는, 의미있는 이름을 지어야 한다.

예약어(reserved word)
프로그래밍 구문에 사용되는 명령어

break, int, const, if, for, class, this 등



변수 사용하기

■ 변수 선언과 사용

```
package variable;

public class Variable {
    public static void main(String[] args) {

        String name;    //문자형 변수 선언
        int grade;       //정수형 변수 선언

        name = "이정우";
        grade = 1;

        //int class = 3; //예약어 이므로 사용 불가
        int schoolClass = 5;

        //'+'는 변수와 문자를 연결하는 연산자
        System.out.println(name + "는 " + grade +
                             "학년 " + schoolClass + "반 입니다.");

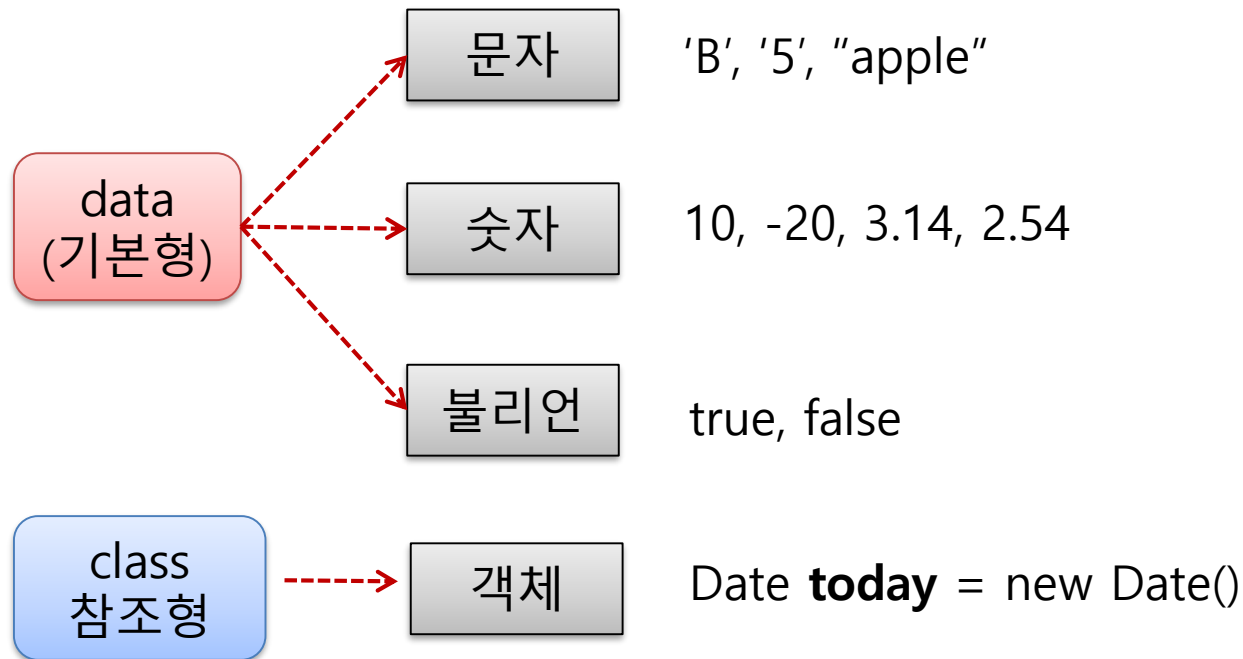
        //변수 이름 작성시 오류
        //int 4grade; //숫자로 시작 불가
        //int school Class; //이름에 공백 불가
        //예약어 - if, for, class, break 등
    }
}
```



자료형(Data Type)

● 자료형이란?

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



자료형(Data Type)

● 기본 자료형

자료형		용량 (bytes)	주요 용도	범위
정수형	byte	1	작은 정수 표현	-128~127
	short	2	정수 표현	-32768~32767
	int	4	큰 범위의 정수 표현	-2147483648~2147483647
	long	4	큰 범위의 정수 표현	$-2^{31} \sim (2^{31}-1)$
실수형	float	4	실수 표현	$10^{-38} \sim 10^{38}$
	double	8	정밀한 실수 표현	$10^{-380} \sim 10^{380}$
문자형	char	2	영문자, 한글 표현	-32768~32767
논리형	boolean	1	true(참) / false(거짓)	-128~127



정수 자료형

- 숫자(정수, 실수) 자료형의 종류 및 크기
 - **byte(1byte)**
 - 동영상, 음악, 이미지 파일등 이진(바이너리) 실행 파일의 자료
 - **short(2byte)**
 - 주로 C/C++ 언어와의 호환 시 사용됨
 - **int(4byte)**
 - 프로그램에서 사용하는 모든 정수는 기본적으로 int로 저장됨
 - **long(8byte)**
 - 가장 큰 정수 자료형으로 숫자 뒤에 'L' 또는 'l'을 써서 표시
 - **float(4byte)**
 - 실수 자료형으로 숫자 뒤에 'F' 또는 'f'를 써서 표시
 - **double(8byte)**
 - 프로그램에서 실수는 기본적으로 double로 저장됨



숫자 자료형 실습

● 숫자 자료형의 종류 및 범위

```
package datatype;

public class NumberType {
    public static void main(String[] args) {
        System.out.println("***** 정수 자료형 *****");
        byte bData1 = 127; //byte(1byte) : -128 ~ 127
        //byte bData2 = 128; //범위를 벗어나 오류

        System.out.println(bData1);

        int iNum = 220000000; //int(4byte) : -21억 ~ 21억
        long lNum = 3000000000L; //long(8byte)

        System.out.println(iNum);
        System.out.println(lNum);

        System.out.println("***** 실수 자료형 *****");
        //float(4byte), double(8byte) - 정밀도 차이
        float fNum = 1.23456789F; //소수 7자리까지 표현
        double dNum = 1.23456789012345678; //소수 16자리까지 표현

        System.out.println(fNum);
        System.out.println(dNum);
    }
}
```

```
***** 정수 자료형 *****
127
2200000000
3000000000
***** 실수 자료형 *****
1.2345679
1.2345678901234567
```



문자 자료형

- 문자 자료형의 종류

- **char**

- 자바에서는 문자를 2바이트로 처리
 - 문자 1개를 표기할때 홑따옴표(' ')로 감싸준다. ('A', 'a', '가')

- **String**

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



문자 자료형

- 문자 자료형의 종류 및 범위

```
public class CharStringType {  
    public static void main(String[] args) {  
        System.out.println("***** 문자 자료형 *****");  
        //char(2byte) - 1개 문자 표현  
        char ch1 = 'A';  
        System.out.println(ch1);  
        //(int) - 숫자형으로 변환  
        System.out.println((int)ch1); //아스키 코드값  
  
        char ch2 = 66;  
        System.out.println(ch2);  
  
        int ch3 = 67;  
        System.out.println(ch3);  
        System.out.println((char)ch3);  
  
        char kor1 = '가';  
        System.out.println(kor1);  
  
        char kor2 = '\uAC00'; //유니코드  
        System.out.println(kor2);  
    }  
}
```

형변환



문자 자료형

● 문자 자료형의 종류 및 범위

```
System.out.println("***** 문자열 자료형 *****");  
//String  
String jdk = "jdk";  
String version = "21";  
String java = jdk + version;  
  
System.out.println(jdk);  
System.out.println(version);  
System.out.println(java);  
  
String kor3 = "가"; //String은 쌍따옴표  
String cart = "라면";  
System.out.println(cart);  
System.out.println(kor3);  
  
//배열  
String[] carts = {"라면", "빵", "우유"};  
System.out.println(carts[2]); //우유  
}  
}
```

```
***** 문자 자료형 *****  
A  
65  
B  
67  
C  
가  
가  
***** 문자열 자료형 *****  
jdk  
21  
jdk21  
라면  
가  
우유
```



■ 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
- 아스키(ASCII) – 1 바이트로 영문자, 숫자, 특수 문자 등을 표현 함
- 유니코드(Unicode) – 2바이트로 한글과 같은 복잡한 언어를 표현하기 위한
표준 인코딩 UTF-8, UTF-16이 대표적,

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



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 체계이다.

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



아스키 코드 vs 유니코드

★ 유니 코드(Uni code)

전 세계의 모든 문자를 다루도록 설계된 표준 문자 전산 처리 방식. 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 할당하였다.

하지만 영어권 이외의 국가에 이메일을 보냈더니 글자가 깨졌던 것. 이에 따라 2~3바이트의 넉넉한 공간에 세상의 모든 문자를 할당한 결과물이 이 코드이다.

	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8	AC9	ACA	ACB	ACC	ACD	ACE	ACF
0	가 AC00	감 AC10	갸 AC20	갯 AC30	갈 AC40	각 AC50	갬 AC60	거 AC70	검 AC80	겐 AC90	겜 ACA0	겔 ACB0	결 ACC0	겟 ACD0	고 ACE0	곰 ACF0
1	각 AC01	갑 AC11	갸 AC21	갹 AC31	갈 AC41	갹 AC51	겜 AC61	걱 AC71	겁 AC81	겜 AC91	겜 ACA1	겜 ACB1	겜 ACC1	겜 ACD1	곡 ACE1	굽 ACF1
2	갹 AC02	갹 AC12	겜 AC22	갹 AC32	갹 AC42	갹 AC52	겜 AC62	겜 AC72	겜 AC82	겜 AC92	겜 ACA2	겜 ACB2	겜 ACC2	겜 ACD2	곡 ACE2	굽 ACF2



논리형 자료형

● 논리형(불리언) 자료형의 종류

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

```
public class BooleanType {  
  
    public static void main(String[] args) {  
        // boolean(1Byte) - 참/거짓  
        boolean value1 = true;  
        System.out.println(value1);  
  
        boolean value2 = (10 > 20);  
        System.out.println(value2);  
  
        System.out.println("***** 사원 정보 *****");  
        String name = "오상식";  
        int age = 41;  
        boolean isMerried = true;  
        int numberOfChildren = 3;  
  
        System.out.println("이름: " + name);  
        System.out.println("나이: " + age);  
        System.out.println("결혼유무: " + isMerried);  
        System.out.println("자녀수: " + numberOfChildren);  
    }  
}
```



형 변환(Type Conversion)

■ 형 변환

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

<묵시적 형변환(자동 형변환)>

- 작은 자료형에서 큰 자료형으로 변환
`int iNum = 20;`
`float fNum = iNum;`
- 연산 중 자동변환
`double dNum = fNum + iNum`

<명시적 형변환(강제 형변환)>

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

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



형 변환(Type Conversion)

▪ 자료형의 크기 및 형 변환

```
public class TypeConversion {  
    public static void main(String[] args) {  
        //자료형의 크기 확인  
        System.out.printf("byte형의 크기 ==> %dbit\n", Byte.SIZE);  
        System.out.printf("short형의 크기 ==> %dbit\n", Short.SIZE);  
        System.out.printf("int형의 크기 ==> %dbit\n", Integer.SIZE);  
        System.out.printf("long형의 크기 ==> %dbit\n", Long.SIZE);  
        System.out.printf("float형의 크기 ==> %dbit\n", Float.SIZE);  
        System.out.printf("double형의 크기 ==> %dbit\n", Double.SIZE);  
        System.out.printf("char형의 크기 ==> %dbit\n", Character.SIZE);  
  
        //묵시적 형변환(자동 형변환)  
        int iNum = 20;  
        float fNum = iNum;  
  
        System.out.println(iNum); //20  
        System.out.println(fNum); //20.0  
    }  
}
```



형 변환(Type Conversion)

▪ 자료형의 크기 및 형 변환

```
double dNum;  
dNum = iNum + fNum;  
  
System.out.println(dNum); //40.0  
  
//명시적 형변환(강제 형변환)  
double dNum2 = 1.5;  
float fNum2 = 0.7F;  
  
int iNum2 = (int)dNum2; //실수형 -> 정수형  
int iNum3 = (int)fNum2;  
int iNum4 = (int)(dNum2 + fNum2);  
  
System.out.println(iNum2); //1  
System.out.println(iNum3); //0  
System.out.println(iNum4); //2  
}  
}
```

byte형의 크기	==> 8bit
short형의 크기	==> 16bit
int형의 크기	==> 32bit
long형의 크기	==> 64bit
float형의 크기	==> 32bit
double형의 크기	==> 64bit
char형의 크기	==> 16bit

20
20.0
40.0
1
0
2



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



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

■ 진수 표기 및 음의 정수 구현하기


[illegible]

대입 및 부호 연산자

■ 대입 연산자

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

int x = 10



```
public class SwapTest {  
    public static void main(String[] args) {  
        // 변수의 값 교환하기  
        int x = 0;  
        int y = 1;  
  
        System.out.println("**** 교환 전 ****");  
        System.out.println("x = " + x + ", y = " + y);  
  
        int temp; //교환을 위한 임시 변수  
  
        //값 교환  
        temp = x;  
        x = y;  
        y = temp;  
  
        System.out.println("**** 교환 전 ****");  
        System.out.println("x = " + x + ", y = " + y);  
    }  
}
```



산술 연산자

■ 산술 및 증감 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

연산자	기 능	연산 예
++	항의 값에 1을 더합니다.	num++; // num = num+1(후치) ++num; // num = num+1(전치)
--	항의 값에서 1을 뺍니다.	num--; // num = num-1(후치) --num // num = num-1(전치)



■ 산술 연산

```
package operators;

public class Operator1 {

    public static void main(String[] args) {
        // 산술 연산
        System.out.println("*** 산술 연산 ***");
        int n1 = 4;
        int n2 = 7;

        System.out.println(n1 + n2);
        System.out.println(n1 - n2);
        System.out.println(n1 * n2);
        System.out.println((double)n1 / n2); //형 변환
        //소수 자리수 처리 : String.format() - 서식 문자(%d-정수, %f-실수)
        System.out.println(String.format("%.2f", (double)n1 / n2));
        //printf() 사용
        System.out.printf("%.2f\n", (double)n1 / n2);
        System.out.println(n1 % n2); //나머지
    }
}
```



■ 산술 연산

```
//증감 연산
System.out.println("*** 증감 연산 ***");
int a = 99;

System.out.println(a++); //99
System.out.println(a); //100

System.out.println(++a); //101
System.out.println(a); //101

System.out.println(a--); //101
System.out.println(a); //100

System.out.println(--a); //99
System.out.println(a); //99
}
```



관계(비교) 및 논리 연산자

■ 비교 및 논리 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

연산자	기 능	연산 예
&&	두 항이 모두 참인 경우에만 결과 값이 참 입니다.	(7<3) && (5>2)
	두 항중 하나의 항만 참이면 결과 값이 참 입니다.	(7<3) (5>2)
!	참인 경우는 거짓으로, 거짓은 참으로 바꿉니다.	!(7>3)



비교 및 논리 연산

■ 비교 및 논리 연산

```
public class Operator2 {  
  
    public static void main(String[] args) {  
        // 비교 연산자  
        System.out.println("*** 비교 연산 ***");  
        int n1 = 10;  
        int n2 = 20;  
  
        System.out.println(n1 >= n2); //false  
        System.out.println(n1 <= n2); //true  
        System.out.println(n1 == n2); //false  
        System.out.println(n1 != n2); //true  
  
        // 논리 연산자  
        System.out.println("*** 논리 연산 ***");  
        System.out.println((n1 < n2) && (n1 == n2)); //false  
        System.out.println((n1 < n2) || (n1 == n2)); //true  
        System.out.println(!(n1 == n2)); //true  
    }  
}
```



복합대입 및 조건 연산자

■ 복합대입 및 조건 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

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



복합 대입 및 조건 연산

■ 복합 대입 및 조건 연산

```
System.out.println("*** 복합 대입 연산 ***");
int num = 10;

System.out.println(num += 2); //num = num + 2
System.out.println(num -= 2); //num = num - 2
System.out.println(num *= 2); //num = num * 2
System.out.println(num /= 2); //num = num / 2

System.out.println("*** 조건 연산 ***");
//부모 나이 비교
int fatherAge = 55;
int motherAge = 57;

//결과 - true/false
boolean result1 = (fatherAge > motherAge) ? true : false;
System.out.println(result1);

//결과 - T/F
char result2 = (fatherAge > motherAge) ? 'T' : 'F';
System.out.println(result2);
```



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 a를 2비트 만큼 왼쪽으로 이동
>>	오른쪽 shift	$A \gg 2$ 변수 a를 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

■ 비트 연산

```
// 비트 논리 연산자
int num1 = 5;    //00000101
int num2 = 10;   //00001010
int result1, result2;

result1 = num1 & num2; //00000000
System.out.println(result1); //0

result2 = num1 | num2; //00001111
System.out.println(result2); //15

//비트 이동 연산자
int value = 3; //00000011
int result3, result4;

result3 = value << 2;
System.out.println(result3); //00001100, 12

result4 = value >> 1;
System.out.println(result4); //00000001, 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

```
package input;

import java.util.Scanner;

public class ScannerEx1 {

    public static void main(String[] args) {
        //입력 처리 - Scanner 클래스 필요함
        //scanner 객체 생성
        Scanner scanner = new Scanner(System.in);

        System.out.print("당신의 이름은 무엇입니까? ");
        String name = scanner.nextLine(); //문자열 입력
        System.out.println("당신의 이름은 " + name + "이군요!");

        System.out.print("당신의 나이는 몇 살입니까? ");
        int age = scanner.nextInt(); //정수 입력
        System.out.println("당신의 나이는 " + age + "세 이군요!");

        scanner.close(); //scanner 닫기
    }
}
```



상수(Constant)

■ 상수(constant)

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

final double PI = 3.14;

final로 선언된 상수는 다른 값을 대입할 수 없음

PI = 3.15 //에러



상수와 리터럴

■ 상수 실습 예제

```
package constant;

public class FinalEx1 {

    public static void main(String[] args) {
        //상수 사용하기
        final int MIN_NUM = 0;
        final int MAX_NUM = 100;

        //MAX_NUM = 1000; 수정할 수 없음

        System.out.println(MIN_NUM);
        System.out.println(MAX_NUM);

        //원의 넓이 계산하기
        final double PI = 3.1415;
        int radius = 5;
        double area = PI * radius * radius;

        //System.out.println("원의 넓이 : " + area);
        //서식 문자 - %d(정수), %f(실수), %s(문자열)
        System.out.printf("원의 넓이: %.3f", area);
    }
}
```



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.



실습문제 1 – 몫과 나머지

빵 25개를 4명이 나눠 가질 경우 각자의 몫과 남은 빵의 개수를 구하시오.
(파일 이름 : Bread.java)

☞ 실행 결과

```
각자의 몫: 6  
남은 빵의 개수: 1
```



실습문제 2 – 조건 연산자 사용

숫자를 입력받아 홀수/짝수를 판별하는 프로그램을 작성하세요.

(힌트: 조건 연산자 활용, 파일이름: EvenOdd.java)

👉 실행 결과

```
숫자를 입력하세요: 11  
입력한 11은(는) 홀수입니다.
```

