
JAVA
변수, 연산자, 수식
Microsoft

학습 목표

식별자와 예약어를 알아보고 사용 관례에 관해 학습합니다.

변수의 의미와 변수명을 지정하는 방법에 관해 학습합니다.

자료형의 종류와 사용 방법을 알아봅니다.

연산의 의미와 서로 다른 자료형 간의 형 변환에 관해 학습합니다.

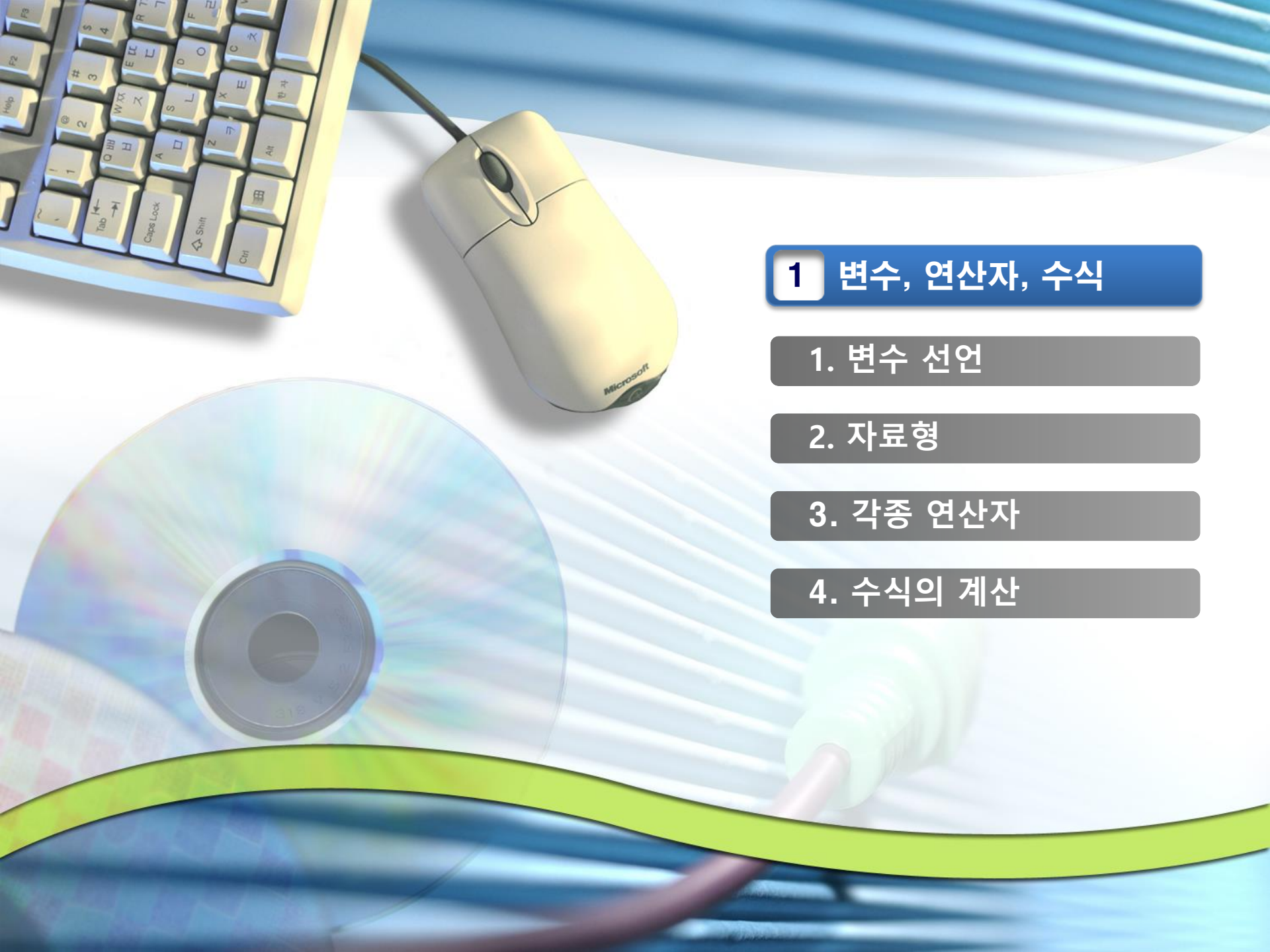
다양한 연산자들의 사용 방법에 관해 알아봅니다.

수식에서 사용된 연산자의 우선순위에 관해 학습합니다.

문자열과의 사용 방법에 관해 학습합니다.

자바의 표준 출력과 키보드 입력에 관해 학습합니다.

자바의 주석과 오류에 관해 학습합니다.



1 변수, 연산자, 수식

1. 변수 선언
2. 자료형
3. 각종 연산자
4. 수식의 계산

변수와 자료형

1) 변수란? 프로그램에서 데이터 값들이 저장되는 공간을 변수(variable)라고 한다.

2) 자료형(data type)은 변수에 저장되는 자료의 타입을 의미한다

기초형(primitive type)

- 값을 다루는 자료형
- 정수형, 실수형, 문자형, 논리형

참조형(reference type)

- 주소를 다루는 자료형
- 클래스, 배열, 인터페이스

변수의 선언과 초기화

1) 변수 선언이란 컴파일러에게 어떤 변수를 사용하겠다고 미리 알리는 것이다.

```
자료형 변수명;
```

2) 변수의 선언과 동시에 초기화

```
자료형 변수명 = 값;
```

변수의 사용

❖변수의 사용

- 변수값 읽기
  - 변수는 초기화가 되어야 읽기 가능
  - 잘못된 코딩의 예

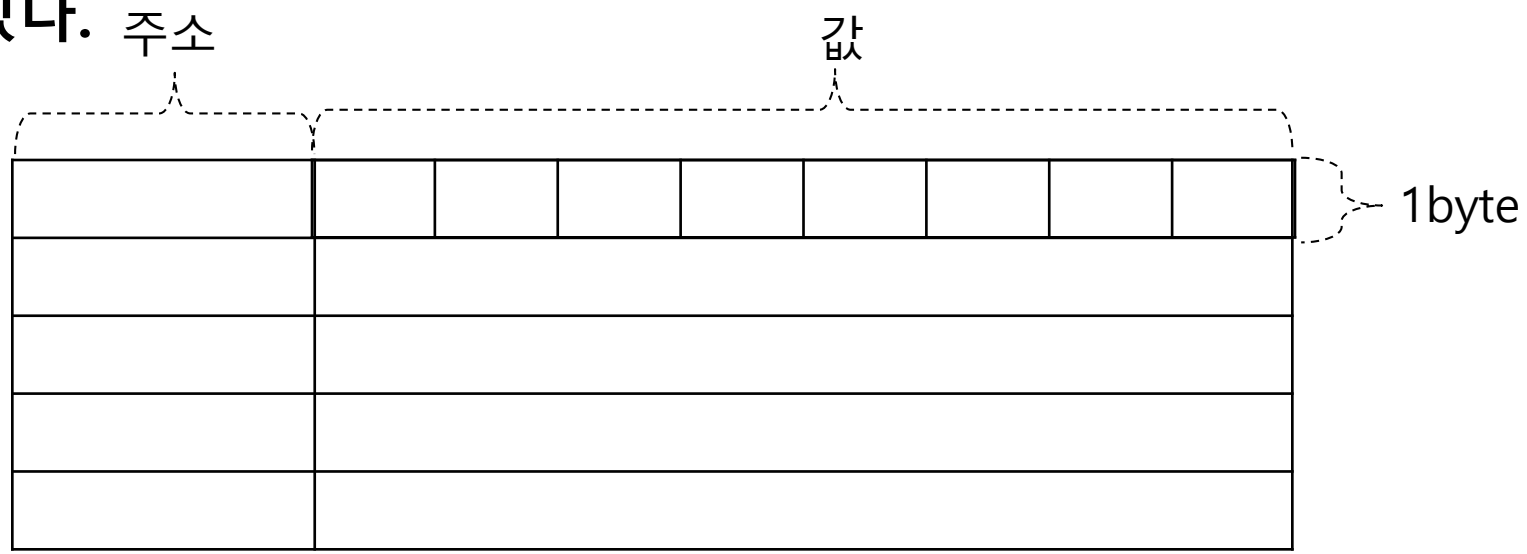
```
int value;
int result = value + 10;
```

  - 맞게 고친 후의 코드

```
int value = 30;
int result = value + 10;
```

메모리에 할당 명령 내리기

메모리는 데이터를 저장하기 위한 장치이다. 기본 단위인 바이트(byte, 8bit)로 나누어져 있다 이 단위로 문자 데이터를 표현할 수 있다.



메모리의 크기

비트(bit): 0과 1을 저장하는 최소 저장 단위.

바이트(byte): ASCII 코드와 같은 문자 데이터를 저장하는 단위.

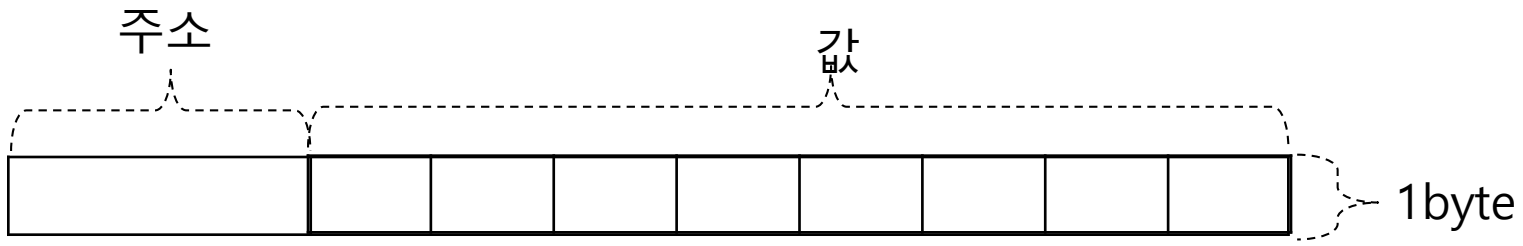
메모리에 할당 명령 내리기

1바이트(8비트)단위로 나눈 이유는 문자 하나를 기억하는데 필요한 최소 메모리 크기가 8비트이기 때문이다.

그리고 바이트 단위로 나누어진 각 영역을 구별하기 위해 바이트 단위로 주소가 할당된다. 메모리를 다루는 기본 단위는 바이트이며, 바이트로 나누어진 각 영역은 주소를 이용해 참조할 수 있도록 구조화 되어 있다,

메모리에 할당 명령 내리기

메모리의 중요한 특징이 여기에 있다. 바로 메모리의 주소를 이용해야만 해당 메모리에 저장된 값을 참조할 수 있다는 것이다



주기억 장치(메모리)의 특징 = 주솟값을 이용해서 값을 참조한다.

현재는 개발자가 메모리에 직접 관리하지 않기 때문에 메모리의 어디가 비었는지 알 수가 없다. 그래서 개발자는 CPU에 메모리 참조 명령을 내릴 수 없다. 현재 메모리는 운영체제의 커널에서 관리하고 개발자는 주솟값 대신 식별자로 메모리를 참조할 수 있다.

변수의 이름

1) 변수의 이름은 식별자(identifier)의 일종으로 다음과 같은 규칙을 따른다.

- 식별자는 문자와 숫자의 조합으로 만들어진다.
- 식별자의 첫 문자는 일반적으로 문자이거나 _, $이어야 하고 숫자로 시작할 수 없다.
- 식별자는 공백을 포함할 수 없다.
- 대문자와 소문자는 구별된다.
- 첫문자는 영어 소문자로 시작하되, 다른 단어와 연결하여 사용할 경우 첫글자를 대문자로 한다.
- 식별자의 이름으로 키워드(keyword)를 사용해서는 안 된다.

변수의 이름

1) 변수의 이름은 식별자(identifier)의 일종으로 다음과 같은 규칙을 따른다.

적합한 변수 선언 예를 들어보면 다음과 같다.

int speed;
long firstName;
int _count; // _로 시작할 수 있다.
long $value; // $로 시작할 수 있다.
int 반복횟수; // 유니코드를 지원하므로 한글 변수 이름도 가능
int num1; // 맨 처음이 아니라면 숫자도 넣을 수 있다.

잘못된 변수 선언의 예는 다음과 같다.
int 1stPrizeMoney // 첫글자가 숫자 double super // 키워드 (true, false, null) int #ofComputer; // 첫글자가 허용되지 않는 기호

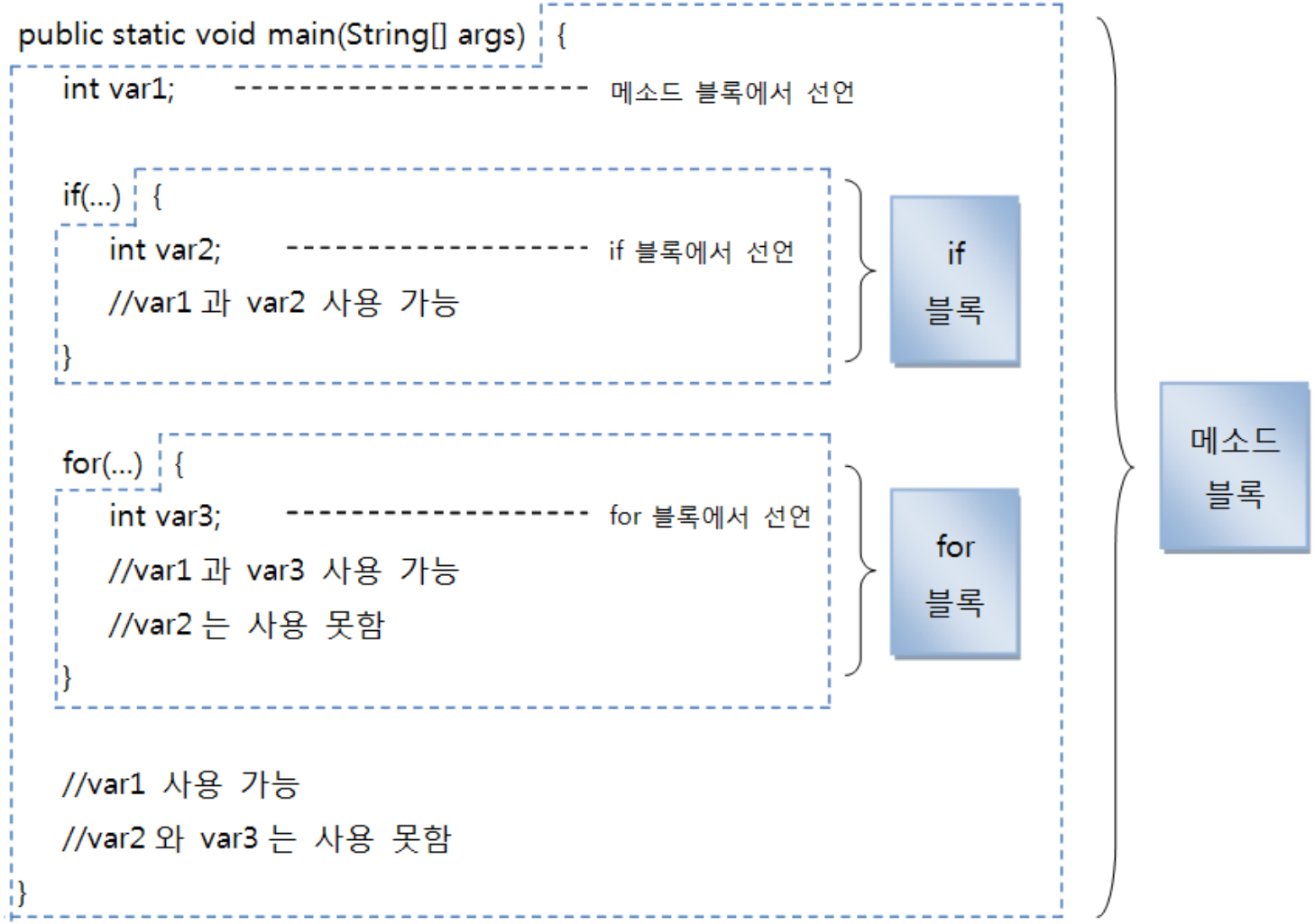
※ 유니코드 : 전 세계의 모든 문자를 컴퓨터에서 일관되게 표현하고 다룰 수 있도록 설계된 산업 표준이며, 유니코드 협회가 제정한다.

식별자 이름 작성 관례

종류	사용방법	예
클래스명	각 단어의 첫글자는 대문자로 한다.	Employee StaffMember, ItemProducer
변수명, 메서드명	소문자로 시작되어 2번째 단어의 첫글자는 대문자로 한다.	width, height, payRate, accountNumber, currentImage setColor(), fillRect()
static final 자료형 변수	변하지 않는 숫자를 나타내는 변수, 모든 글자를 대문자로 한다.	PI, MAX_NUMBER

변수

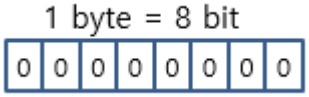
변수의 사용 범위 - 변수는 중괄호 블록 {} 내에서 선언되고 사용



기초형

1) 기초형(primitive data type)은 정수(int), 실수(float), 문자형(char), 논리형(boolean) 등의 일반적일 데이터를 나타내는 형으로 모두 8개가 있다.

- 메모리의 최소 기억단위인 bit가 모여 byte 형성



데이터형	설명	크기(비트)	기본값	최소값	최대값
byte	부호있는 정수	8비트	0	-128	127
short	부호있는 정수	16비트	0	-32768	32767
int	부호있는 정수	32비트	0	-2147483648	2147483647
long	부호있는 정수	64비트	0L	-9223372036854775808	9223372036854775807
float	실수	32비트	0.0f	약 $\pm 3.4 \times 10^{-38}$ (유효숫자 7개)	약 $\pm 3.4 \times 10^{+38}$ (유효숫자 7개)
double	실수	64비트	0.0d	약 $\pm 1.7 \times 10^{-308}$ (유효숫자 15개)	약 $\pm 1.7 \times 10^{+308}$ (유효숫자 15개)
char	문자(유니코드)	16비트	null	'\u0000'(0)	'\uFFFF'(65535)
boolean	true 또는 false	8비트	false	해당없음	해당없음

정수형

이는 저장하고자 하는 값의 범위 따라서 적절한 자료형을 선택할 수 있도록 하기 위함이다. 바이트 크기가 크면 그만큼 표현할 수 있는 값의 범위도 커지기 마련이다.

1) 정수형 정수 타입일 경우 $-2^{n-1} \sim 2^{n-1}-1$의 값을 저장. n이 메모리 사용 크기(bit 수)를 의미.

- byte는 8비트 정수로서 -128에서 +127까지의 정수를 표현
- short는 16비트를 이용하여 -32,768에서 +32767사이의 정수를 표현
- int는 32비트를 이용하여 약 -21억에서 21억 정도의 정수를 표현
- long은 64비트를 이용

2) 정수형 리터럴

리터럴(literal)이란 x = 100;에서 100과 같이 소스 코드에 쓰여 있는 값을 의미한다.

정수형(byte형)

최상위 부호 비트								[이진수]	[십진수]
0	1	1	1	1	1	1	1	→	127
0	1	1	1	1	1	1	0	→	126
									⋮
0	0	0	0	0	0	1	0	→	2
0	0	0	0	0	0	0	1	→	1
0	0	0	0	0	0	0	0	→	0
1	1	1	1	1	1	1	1	→	-1
1	1	1	1	1	1	1	0	→	-2
									⋮
1	0	0	0	0	0	0	1	→	-127
1	0	0	0	0	0	0	0	→	-128

정수형

- 최상위 부호 비트(MSB: Most Significant Bit)는 정수값의 부호를 결정한다. 최상위 비트가 0이면 양의 정수, 1이면 음의 정수를 뜻한다. 실제 정수값은 나머지 7개의 bit로 결정한다.
- 1의 보수(n의 보수란 어떤 수가 n이 되려면 보충해야 하는 수를 의미)1의 보수는 각 자리를 1에서 빼어 나타낸 것이다. 즉 0 →1, 1 →0으로 바꾸면 1의 보수를 구할 수 있다.
- 2의 보수는 1의 보수의 최하위 비트에 +1을 해서 구할 수 있다.

예를 들어 -2는 다음과 같이 계산된다.

정수형 리터럴

0~7까지 8개의 수를 사용하여 표현하는 숫자

리터럴 : 프로그램에서 쓰는 구체적인 데이터의 표현.

변수와 상수에 대입되는 값 자체를 말한다.

10진수(decimal): 14, 16, 17

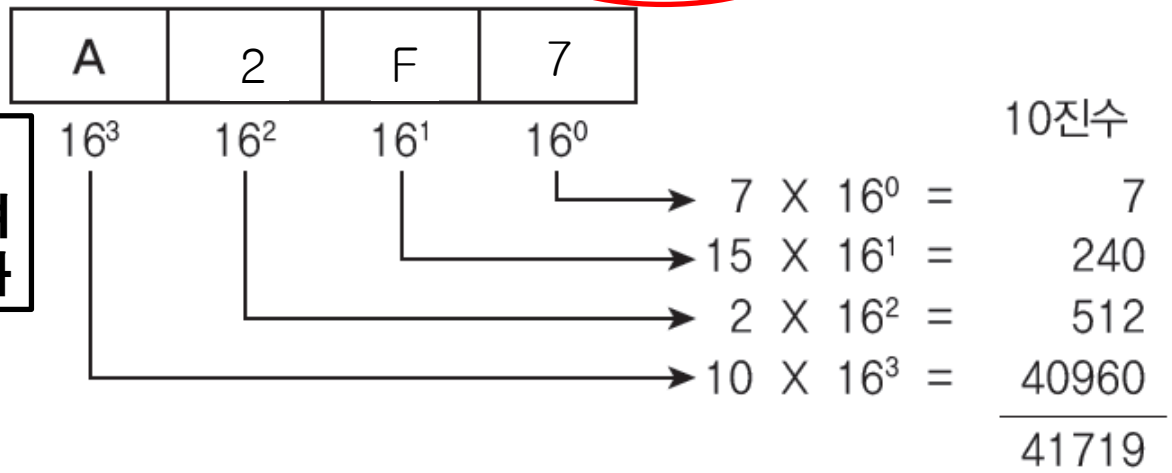
8진수(octal): 016, 012, 011

16진수(hexadecimal): 0xe, 0x10, 0x11

2진수(binary): 0b1100

0~9, A~F까지 16개의 수를 사용하여 표현하는 숫자

0,1 2개의 수를 사용하여 표현하는 숫자



int 타입이 수용할 수 있는 범위를 벗어나는 값은 int가 아니라 long 타입을 사용하도록 표현 방법을 바꿔야 한다. 이를 해결하려면 숫자 끝에 소문자 l 또는 L을 표기해주어야 한다.

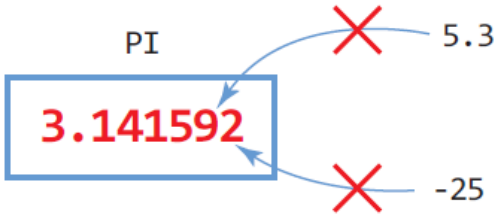
상수

상수 선언

- final 키워드 사용
- 선언 시 초기값 지정
- 실행 중 값 변경 불가

```
final double PI = 3.141592;
```

상수 선언 | 데이터 타입 | 상수 이름 | 초기화



상수는 일단 초기화되어 값이 정해지면 그 값을 변경할 수 없다.

논리형

논리 타입은 "참/거짓"처럼 둘 중 하나의 값을 표현할 때 사용한다. 자바에서 논리 타입의 데이터를 저장하려면 **boolean** 타입을 사용하며, 이 타입은 **1바이트(byte)만큼의 메모리 공간을 차지**한다. 논리 타입에는 **true 또는 false 값**을 저장한다.

[논리 타입 변수 선언]

```
boolean 변수명;
```

[논리 타입 변수에 데이터 저장]

```
boolean checkValue = true;
boolean checkValue = false;
```

실수형

1) 정밀도(값을 정확히 표현할 수 있는 능력)포기하고 표현의 범위를 넓힌 자료형.

- float는 32비트(소수점 이하 6자리의 정밀도)를 이용하여 실수를 표현
- double은 64비트(소수점 이하 15자리의 정밀도)를 이용하여 실수를 표현

자료형	메모리 크기	저장 가능 범위	리터럴 표기
float	4byte	1.4E-45 ~ 3.4028235E38	f와 F로 표기
double	8byte	4.9E-324 ~ 1.79769313486231570E308	실수형에서 디폴트 자료형.

- 소수점 형태나 지수 형태로 표현한 실수
  12. 12.0 .1234 0.1234 1234E-4

실수형

2) 실수형 리터럴

일반 표기법	과학적 표기법	지수 표기법
146.91	1.4691×10^{2}	1.4691E+2
0.00081	8.1×10^{-4}	8.1E-4
1,800,000	1.8×10^{6}	1.8E+6

[실수 타입 변수에 데이터 저장]

```
int var1 = 3000000;   //3000000
double var2 = 3e6;    //3000000
float var3 = 3e6f;    //3000000
```

실수형

3) 부동 소수점

실수는 부동 소수점(floating-point) 방식으로 저장된다. 부동 소수점 방식은 실수를 다음과 같은 형태로 표현한 것을 말한다.

$$+-(1.m)x10^{2}$$

부호 **가수** **지수**

실수를 표현 할 때 소수점의 위치를 고정하지 않고 그 위치를 수치로 표현하며, **유효숫자**를 나타내는 **가수**와 **소수점의 위치**를 나타내는 **지수**로 표현하는 방식.

float: 부호(1bit)+지수(8bit)+가수(23bit) = 32bit = 4byte

1	지수(8bit)	가수(23bit)

double: 부호(1bit)+지수(11bit)+가수(52bit) = 64bit = 8byte

1	지수(11bit)	가수(52bit)

문자형

1) 아스키 코드가 아니라 유니 코드(unicode: 한 문자를 2byte로 표현하는 문자체계)를 사용.

```
char ch1 = '가';
char ch2 = '₩uac00';    // '가'를 나타낸다
```

(유니코드)
컴퓨터에서 세계 각국의 언어를 통일된 방법으로 표현할 수 있게 제안된 국제적인 문자 코드 규약.

유니코드는 사용중인 운영체제, 프로그램, 언어에 관계없이 문자마다 고유한 코드 값을 제공하는 새로운 개념의 코드다. 언어와 상관없이 모든 문자를 16비트로 표현하므로 최대 65,536자를 표현할 수 있다.

문자형

유니코드는 하나의 문자에 대해 하나의 코드값을 부여하기 때문에 영문 'A' 및 한글 '가'도 하나의 코드값을 갖는다. 유니코드는 0~65535 범위의 2byte 크기를 가진 정수값이다. 0~127까자는 아스키(ASCII) 문자(특수기호 및 영어 알파벳)가 할당되어 있고, 44032~55203까지는 한글 11172자가 할당되어 있다. (http://www.unicode.org)

char 타입에 저장할 수 있는 값은 0~65535까지이다.

```
char var1 = 'A'   유니코드: 0x0041 2진수: 00000000 01000001
char var2 = 'B'   유니코드: 0x0042 2진수: 00000000 01000010

char var3 = '가' 유니코드: 0xAC00 2진수: 10101100 00000000
char var3 = '가' 유니코드: 0xAC01 2진수: 10101100 00000001
```

유니코드는 16진수로 저장할 경우에는 유니코드라는 의미에서 '₩u+16진수값' 형태로 값을 저장하면 된다.

문자형

- 단일 인용부호(' ')로 문자 표현
  - 사례) 'w', 'A', '가', '*', '3', '글', ₩u0041
  - ₩u다음에 4자리 16진수(2바이트의 유니코드)

```
char a = 'A';
char b = '글';
char c = \u0041; // 문자 'A'의 유니코드 값(0041) 사용
char d = \uae00; // 문자 '글'의 유니코드 값(ae00) 사용
```

- 특수문자 리터럴은 백슬래시(₩)로 시작

종류	의미	종류	의미
'\b'	백스페이스(backspace)	'\r'	캐리지 리턴(carriage return)
'\t'	탭(tab)	'\"'	이중 인용부호(double quote)
'\n'	라인피드(line feed)	'\''	단일 인용부호(single quote)
'\f'	폼피드(form feed)	'\\'	백슬래시(backslash)

기본 타입 이외의 리터럴

- **null 리터럴**
  - **레퍼런스에 대입 사용**

오류

```
int n = null;          // 기본 타입에 사용 불가
String str = null;
```

- **문자열 리터럴(스트링 리터럴)**
  - **이중 인용부호로 묶어 표현**
    - **사례) "Good", "Morning", "자바", "3.19", "26", "a"**
  - **문자열 리터럴은 String 객체로 자동 처리**

```
String str = "Good";
```

var 키워드를 사용하여 변수 타입 생략

- var 키워드
  - Java 10부터 도입된 키워드
    - var와 동일한 기능으로 C++(2011년 표준부터)의 auto 키워드
  - 지역 변수의 선언에만 사용
  - 변수 타입 선언 생략 : 컴파일러가 변수 타입 추론
- 사용 예

```
var price = 200;                    // price는 int 타입으로 결정
var name = "kitae";                 // name은 String 타입으로 결정
var pi = 3.14;                      // pi는 double 타입으로 결정
var point = new Point();            // point는 Point 타입으로 결정(추후 정리)
var v = new Vector<Integer>();      // v는 Vector<integer> 타입으로 결정(추후 정리)
```

- 변수 선언문에 반드시 초기값 지정

오류

```
var name;                   // 컴파일 오류. 변수 name의 타입을 추론할 수 없음
```