모바일 프로그래밍 입문
(2018년도 2학기)

변수, 자료형, 연산자

수업 목표

- 식별자와 예약어
- 변수
- 자료형
- 연산과 형 변환
- 연산자와 수식

식별자와 예약어

- **자바에서 식별자**
  - 변수, 상수, 메소드, 배열, 문자열, 사용자가 정의하는 클래스나 메소드 등 구분 가능한 이름
- **식별자의 사용 원칙**
  - 문사, 숫자, 특수문자(_, $)로 구성될 수 있다.
  - 식별자의 첫 문자는 숫자를 사용할 수 없다.
  - 예약어를 식별자로 사용할 수 없다.
  - 식별자는 길이 제한을 두지 않는다.
  - 대소문자가 다른 이름은 서로 다른 식별자로 취급

식별자와 예약어

- 자바의 예약어

abstract	const	finally	interface	short	transient
assert	continue	float	long	static	try
boolean	default	for	native	strictfp*	void
break	do	goto	new	super	volatile
byte	double	if	package	switch	while
case	else	implements	private	synchronized	
catch	enum	import	protected	this	
char	extend	instanceof	public	throw	
class	final	int	return	throws	

식별자와 예약어

- 예약어를 변수로 사용?
  - 실습 예제 3.1

예제 3.1 ReservedWordTest.java

```
01: public class ReservedWordTest {
02:     public static void main(String[] args) {
03:         int if = 20;
04:         double for = 30.0;
05:     }
06: }
```

예약어를 변수로 사용

실행 결과

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
    Syntax error on token "if", invalid VariableDeclaratorId
    Syntax error on token "for", invalid VariableDeclaratorId
    Duplicate local variable $missing$

    at ReservedWordTest.main(ReservedWordTest.java:3)
```

식별자와 예약어

■ 식별자의 사용에 대한 일반적인 관례

클래스 이름

JavaTest1, RuntimeErrorTest ◀---- 바람직한 형태, 단어의 첫 글자는 대문자로 쓰는 것이 좋습니다.

applicationtest, sampletest ◀---- 오류는 아니지만 관례에 어긋나고, 가독성이 떨어집니다.

메소드, 변수, 배열, 문자열의 이름

sum, sumAndSubstract, nameAddress ◀---- 바람직한 형태, 단어의 첫글자는 소문자로 쓰는 것이 좋습니다.

NameAndAge, Productname ◀---- 좋지 않은 형태, 클래스 이름과 혼동됩니다.

상수의 이름

PI, MAX_NUMBER ◀---- 상수는 모두 대문자로 쓰는 것이 관례입니다.

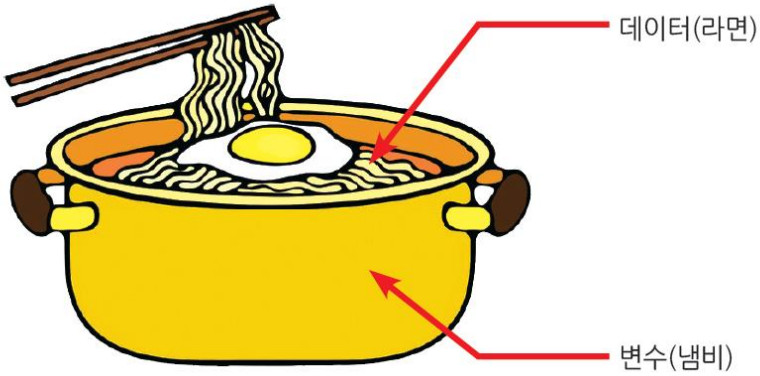
max, Max, Address ◀---- 좋지 않은 형태, 다른 이름과 혼동됩니다.

수업 목표

- 식별자와 예약어
- 변수
- 자료형
- 연산과 형 변환
- 연산자와 수식

변수

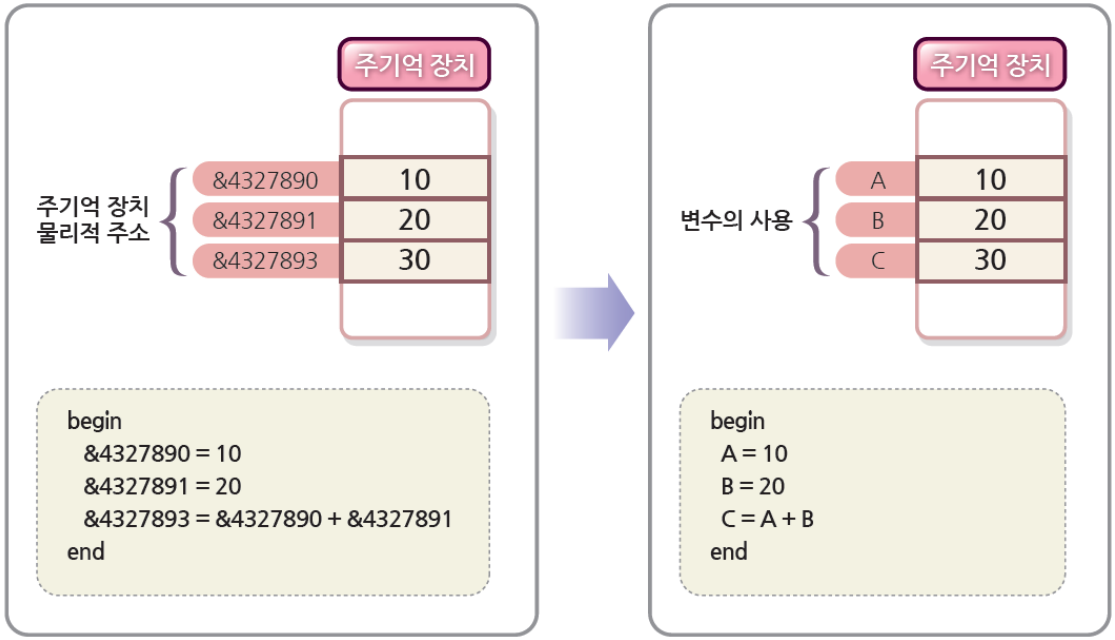
- 변수의 의미
  - 일상 생활에서의 변수는?



    - 우리가 먹을 수 있는 것? (라면? Or 냄비?)
    - 프로그램에서 처리하는 것은 라면, 라면을 저장하는 냄비가 바로 변수

변수

- 변수의 의미
  - 프로그래밍 언어에서 사용되는 변수
    - “값(value)이 저장된 메모리의 위치에 주어진 이름”
    - 변수에 값을 배정(assignment)할 때 “=” 기호를 사용

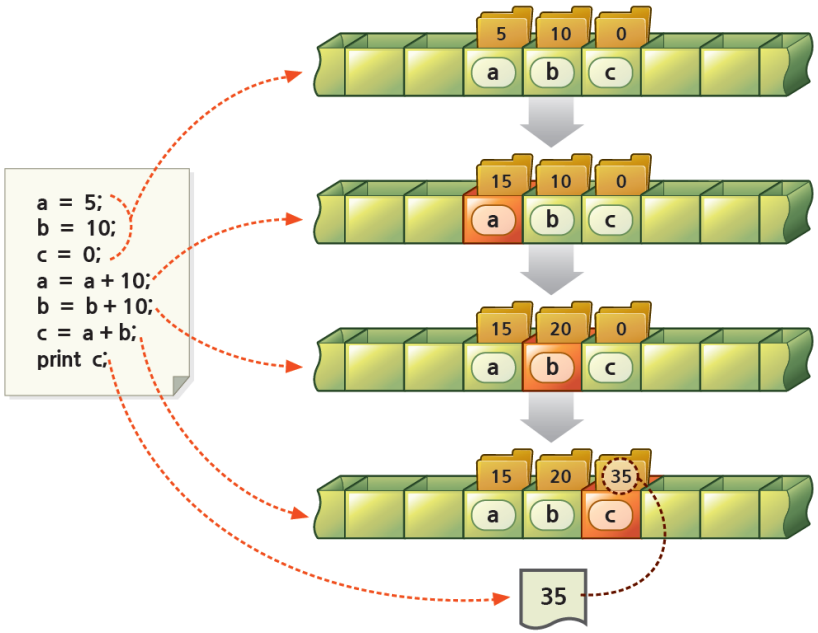


변수

- 변수의 선언과 사용
  - 변수명을 지정하는 규칙
    - 위의 규칙과 같음
  - 변수의 사전적 의미
    - 변화하는 것
  - 좋은 변수 명?
    - 의미를 가진 변수명

변수

- 변수의 선언과 사용
  - 메모리에 저장된 변수의 값 변화



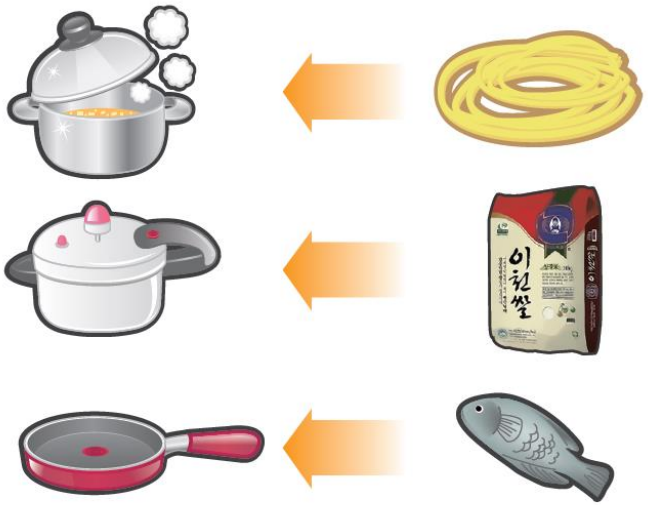
메모리에 저장된 변수값은 프로그램의 진행에 따라 변화됩니다.

수업 목표

- 식별자와 예약어
- 변수
- 자료형
- 연산과 형 변환
- 연산자와 수식

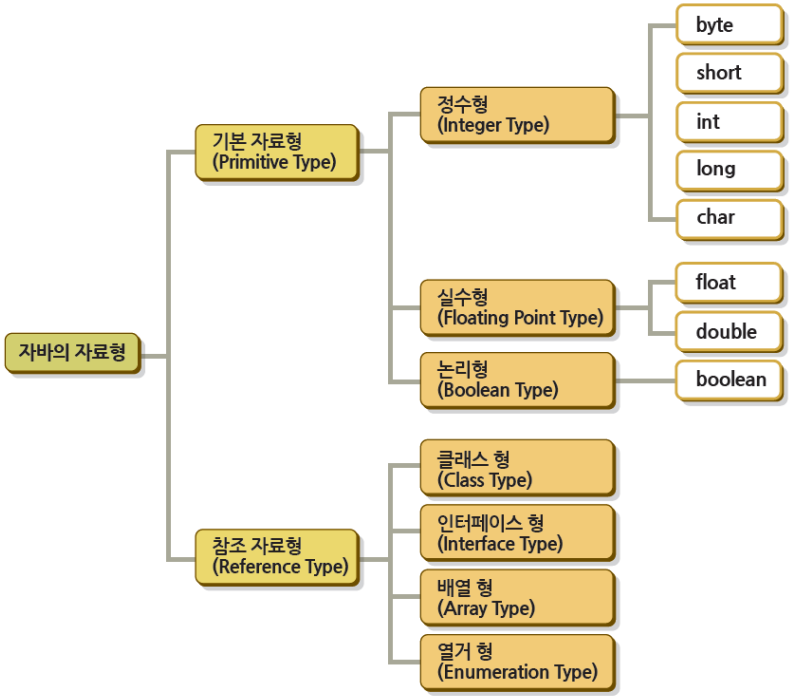
자료형

- **자료형(Data Type)**
  - 변수가 가질 수 있는 값의 형태
- **일상 생활에서의 자료형? 그릇마다 담을 수 있는 자료(데이터)가 다르다.**



자료형

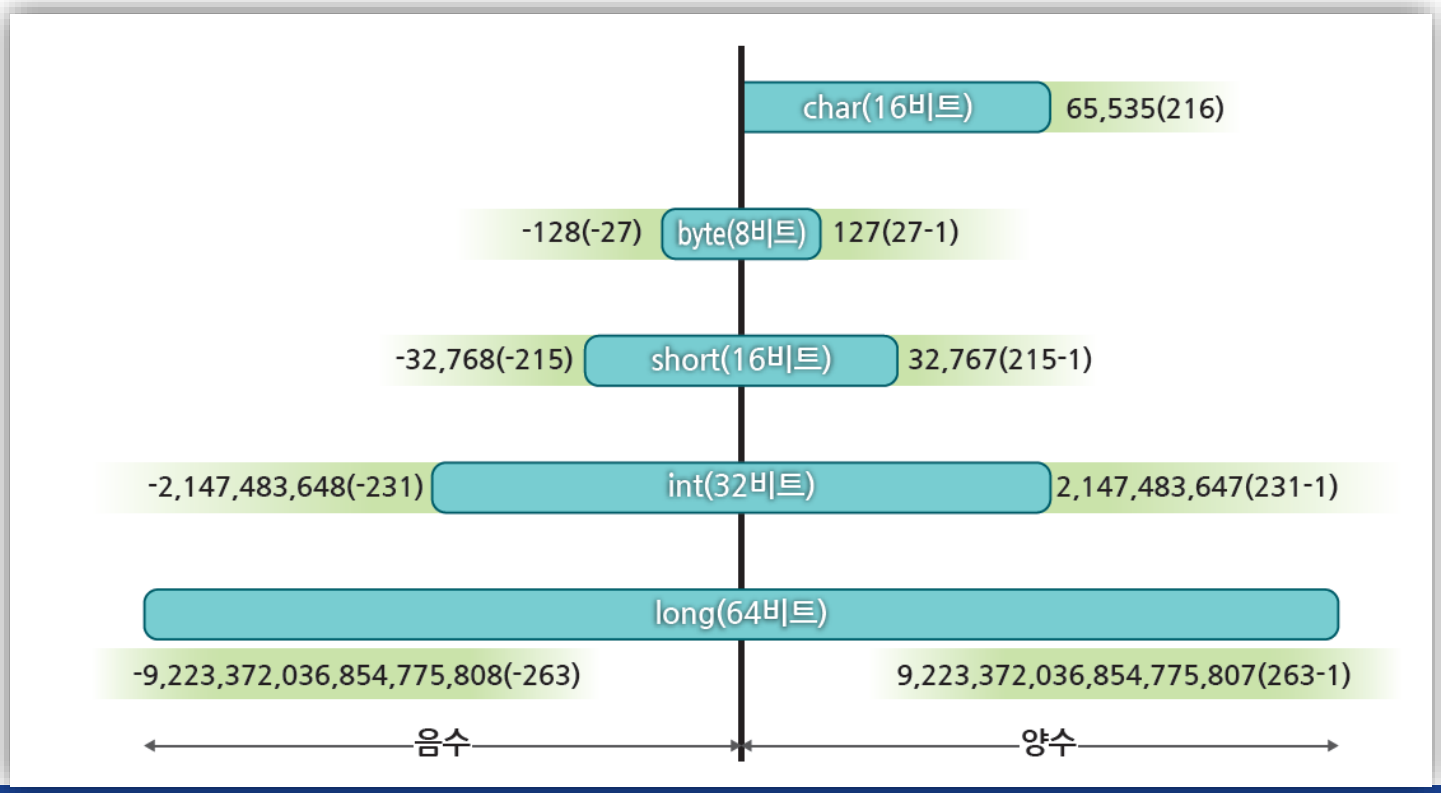
- 자바의 자료형은 크게
  - 기본 자료형 8가지와
  - 참조 자료형 4가지로 구분된다.



- 기본 자료형과 참조 자료형
  - 기본 자료형과 참조 자료형의 차이
    - 기본 자료형 : 값을 가진다.
    - 참조 자료형 : 참조(주소)를 가진다.

자료형

- **정수형**
  - 자바의 정수형 : 5가지(byte, short, int, long, char)



예제 3.3 : 값의 범위를 벗어나면 오류 발생

예제 3.3 ByteTestError.java

```
01: public class ByteTestError {
02:     public static void main(String args[])
03:     {
04:         byte a = 128;
05:         System.out.println("128을 저장한 byte 값은: " + a);
06:     }
07:
08: }
```

정수를 사용하면 int 형으로 취급한다. 127까지는 자동으로 배정되지만, 범위를 벗어나면 오류 발생

실행 결과

```
오류 발생

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Type mismatch: cannot convert from int to byte

    at ByteTestError.main(ByteTestError.java:4)
```

자료형

- 예제 3.4 : 값의 범위를 벗어난 값을 강제 형 변환 하면 배정이 가능하다. 그러나 결과는?

예제 3.4 ByteTypeTest.java

```
01: public class ByteTypeTest {
02:     public static void main(String args[])
03:     {
04:         byte a = (byte)128;
05:         System.out.println("128을 저장한 byte 값은: " + a);
06:         byte b = (byte)256;
07:        System.out.println("256을 저장한 byte 값은: " + b);
08:     }
09: }
```

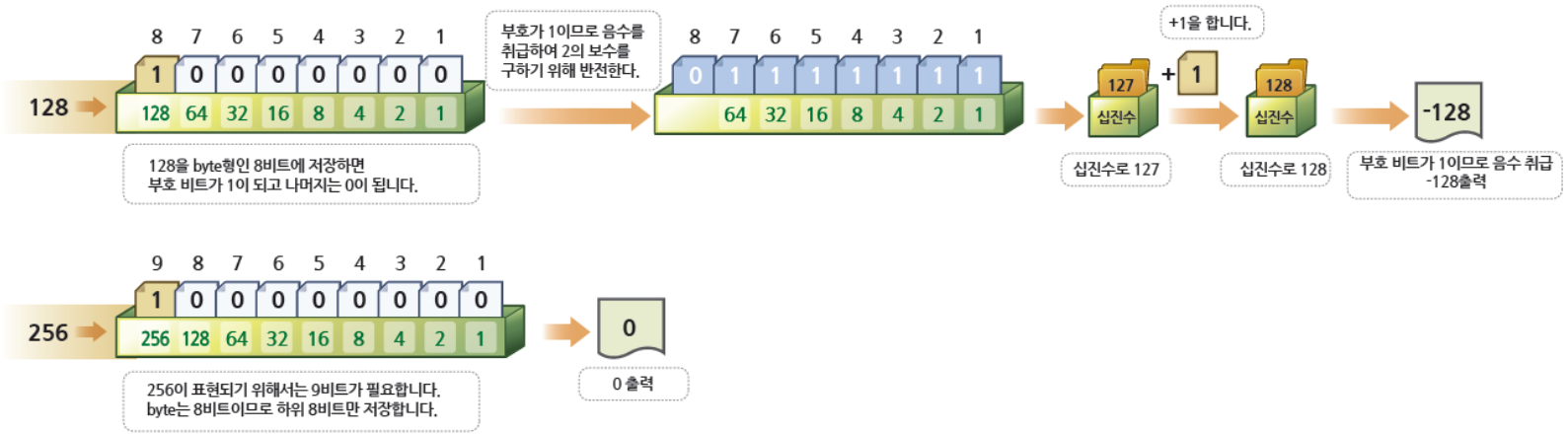
범위를 벗어났지만, byte 형으로 강제 형 변환을 하여 배정

실행 결과

```
128을 저장한 byte 값은: -128
256을 저장한 byte 값은: 0
```

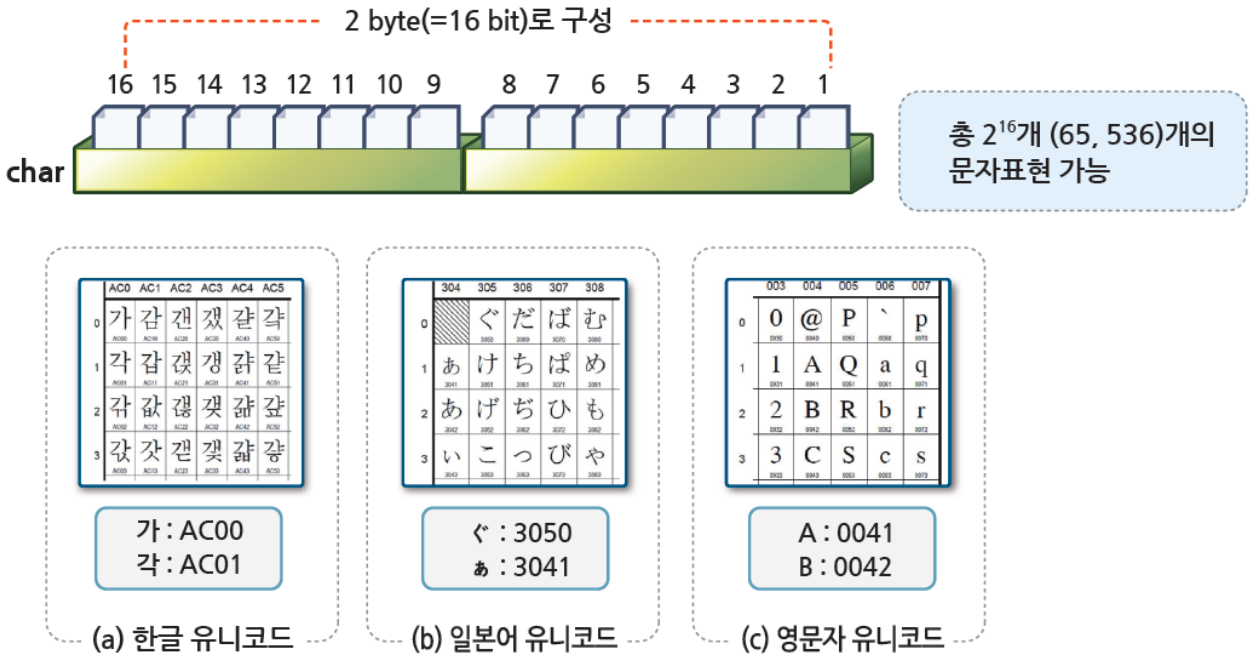
자료형

- 예제 3.4 : 값의 범위를 벗어난 값을 강제 형 변환 하면 배정이 가능하다. 그러나 결과는?



자료형

- **정수형**
  - 문자 정수형 : char (하나의 문자를 나타 냄)
    - 자바의 문자는 16비트 유니코드로 구성



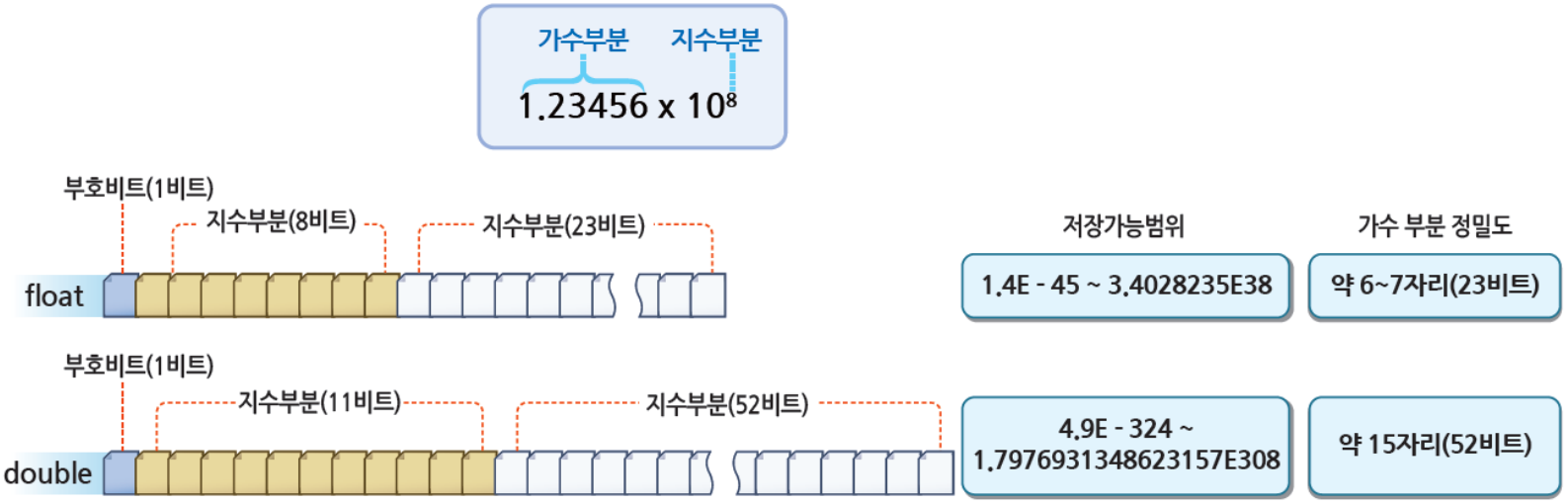
자료형

- **정수형**
  - 문자 정수형 : char (하나의 문자를 나타 냄)
    - 하나의 문자를 나타내기 위해 따움표(‘’)를 사용
      - 대문자 ‘A’를 표시하는 방법 : 문자, 10진수, 8진수, 16진수, 유니코드로 표현 가능

```
char ch1 = 'A';
char ch2 = 65;      <-- 10진수 값으로도 지정 가능
char ch3 = 0101;    <-- 8진수 값으로도 지정 가능
char ch4 = 0x41;    <-- 16진수 값으로도 지정 가능
char ch5 = ₩u0041   <-- 유니코드값으로도 지정 가능
```

자료형

- **실수형**
  - 부호와 지수부분, 가수 부분으로 구성
  - 저장할 수 있는 크기에 따라 (float, double) 구분
  - 묵시적(default) 데이터형은 double형



자료형

- 실수형
  - Float를 사용하려면 반드시 f를 분여야 한다.

```
float f = 3.14f ;  <-------- 오류 없음
float f = 3.14;    <-------- 구문 오류 발생. 실수는 묵시적으로 double로 취급하므로 배정 불가
double d = 3.14;   <-------- 오류 없음
double d = 3.14d;  <-------- 오류 없음
```

논리형

논리형 변수 : Boolean

예제 3.10 : 참(true) 또는 거짓(false)을 저장하는 변수

예제 3.10 BooleanTypeTest.java

```
01: public class BooleanTypeTest {
02:     public static void main(String args[])
03:     {
04:         boolean a = true;   <-- 논리형 변수에 true값을 저장
05:         System.out.println("boolean 변수 a의 값은 : " + a);
06:         boolean b = 10 > 20;   <-- 논리형 변수에 (10 > 20)의 결과를 저장
07:         System.out.println("boolean 변수 b의 값은 : " + b);
08:         boolean c = a;   <-- 논리형 변수에 논리형 변수값을 저장
09:         System.out.println("boolean 변수 c의 값은 : " + c);
10:     }
11: }
```

실행 결과

```
boolean 변수 a의 값은 : true
boolean 변수 b의 값은 : false
boolean 변수 c의 값은 : true
```

자료형

- 상수와 리터럴
  - 상수 : 변하지 않는 값을 저장하는 변수
  - 리터럴 : 값 자체를 의미
  - 상수 선언의 예

```
final int MAX = 100; <---------- 상수변수로 MAX를 선언하고 100으로 초기화
final double PI; <---------- 상수변수 PI를 선언
PI = 3.14159; <---------- PI 초기화
PI = 3.14; <----------오류발생. 상수는 한번 값이 결정되면 변할 수 없음
final boolean FLAG = true; <---------- 상수변수로 boolean 형의 변수 FLAG를 선언하고 true 값으로 초기화
```

- 상수와 리터럴
  - 프로그램에서 상수를 사용하는 이유
    - 같은 리터럴이 여러 번 사용되는 경우 효율성을 위해

```
01: final int MAX = 100;
02: int i = 1;
03: double avg, sum=0.0;
04: while(i <= MAX){
05:     //....
06: }
07: avg = sum / MAX ;
08: System.out.println(MAX+"까지의 합은 : " + sum);
09: System.out.println(MAX+"까지의 평균은 : " + avg);
```

상수 MAX를 100으로 선언. MAX의 값이 변경될 경우 상수 값만 변경

상수 MAX 사용.
상수를 사용하지 않은 경우
값이 변경된 경우 MAX가
사용된 모든 곳을 수정

- **형식 지정자를 사용한 출력 : printf()**
  - 출력문 : System.out.printf()문을 제공
    - printf() 출력문은 C언어의 출력문과 유사항 형식

[형식]

```
System.out.printf([형식제어문자],[변수 또는 리터럴]);
```

"%[옵션지정자][width][.precision]형식지정자"

- 정렬방법이나 부호 표시 등을 지정
- 나타낼 숫자의 폭을 지정 width는 전체 폭을 의미 precision은 소수점 이하의 폭을 의미
- 나타낼 숫자의 형식을 지정

그림 3-11 형식 제어 문자의 구성

형식 지정자를 사용한 출력 : printf()

형식 지정자

형식 지정자	데이터 타입	출력 형식
%d	byte, short, int, long	부호가 있는 10진수 정수 출력
%X, %x	int	16진수를 출력(양수만 가능). 대문자 X의 경우 16진수 알파벳을 대문자로 표시
%o	int	양수의 8진수를 출력(양수만 가능).
%c	char	한 개의 문자 출력
%f	double	고정 소수점 실수 출력
%E, %e	double	부동 소수점 실수 출력. 대문자 E의 경우 지수 문자로 'E'를 사용
%G, %g	double	소수점 이하 자리수가 고정 또는 부동소수점으로 출력. 자리수가 짧은 것을 기준으로 선택함. 대문자 G의 경우 지수 문자로 'E'를 사용
%s	String	문자열 출력
%s	String	문자열 출력
%%		%출력

수업 목표

- 식별자와 예약어
- 변수
- 자료형
- 연산과 형 변환
- 연산자와 수식

연산자 형 변환

- 연산과 자료형
  - 정수 리터럴은 int 형, 실수 리터럴은 double 형
  - 자료형이 다른 경우 자동으로 확대 형 변환 수행

byte 〉〉 short/char 〉〉 int 〉〉 long 〉〉 float 〉〉 double

- 확대 형 변환의 순서 -

[코드]

```
int avg = 9 / 2 ;      <---------- int = int / int  정수 연산이 수행되어 결과로 4가 저장됨
double avg = 9 / 2     <---------- double = int / int 정수 연산이 수행된 결과 4가 double 변수에 저장될 때 4.0으로 변환되어 저장
double avg = 9 / 2.0   <---------- double = int / double 앞쪽의 피연산자가 double로 변환되어 실수 연산이 수행됨.
                                   결과로 4.5가 double 변수에 저장됨
double avg = 9.0 / 2   <---------- double = double / int 뒤쪽의 피연산자가 double로 변환되어 실수 연산이 수행됨.
                                   결과로 4.5가 double 변수에 저장됨
double avg = 9.0 / 2.0 <---------- double = double / double 모든 연산이 실수 연산으로 수행되어 4.5가 저장됨
int avg = 9.0 / 2.0    <---------- 구문 오류 발생됨. 연산은 double로 수행되어 결과도 double, double형 변수값을 정수로
                                   자동 변환 불가
```

연산과 형 변환

◻ 예제 3.14 : 자동 형 변환

예제 3.14 DatatypeOperation1.java

```
01: import java.util.Scanner;
02: public class DatatypeOperation1 {
03:     public static void main(String[] args) {
04:         Scanner stdin = new Scanner(System.in);
05:         System.out.print("첫 번째 정수를 입력 : ");
06:         int first = stdin.nextInt();   <-- 첫 번째로 입력된 정수 저장
07:         System.out.print("두 번째 정수를 입력 : ");
08:         int  second = stdin.nextInt();   <-- 두 번째로 입력된 정수 저장
09:         double avg1 = (first+second) / 2 ;   <-- 정수로 평균을 구하는 연산을 수행
10:         System.out.println("정수 연산 : 평균은("+first+"+"+second+")/2 =
    "+avg1+" 입니다");
11:         double avg2 = (first+second) / 2.0 ;   <-- 실수로 평균을 구하는 연산을 수행
12:         System.out.println("실수 연산 : 평균은("+first+"+"+second+")/2.0 =
    "+avg2+" 입니다");
13:     }
14: }
```

실행 결과

```
첫 번째 정수를 입력 : 3
두 번째 정수를 입력 : 6
정수 연산 : 평균은(3+6)/2 = 4.0 입니다
실수 연산 : 평균은(3+6)/2.0 = 4.5 입니다
```

연산과 형 변환

- **형 변환(Casting)**
  - 확대형 변환과 축소형 변환으로 구분
    - 확대 형 변환
      - 치역이 정의역 보다 더 넓어 값의 손실이 발생되지 않고 저장
      - 자동으로 형 변환이 발생
    - 축소 형 변환
      - 확대 형 변환의 반대의 경우
      - 명시적인 형 변환 구문을 사용

[형 변환 구문]

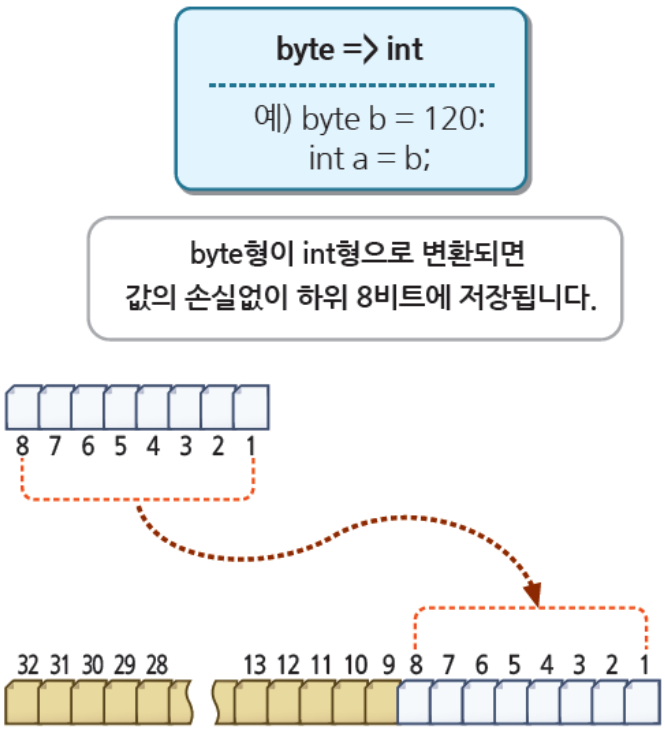
(type) 식 또는 변수

[예]

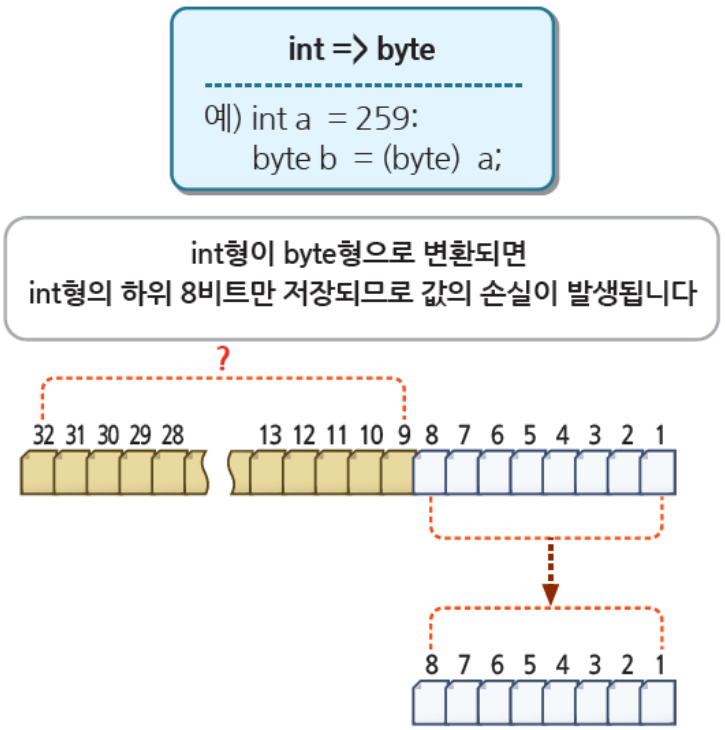
```
(double) 2                                  // 정수 2를 2.0으로 형 변환
double avg1 = (double)(first+second) / 2    // first+second의 결과가 실수로 형 변환
byte b = (byte) 700                         // 정수 700이 바이트 형으로 형 변환
```

연산과 형 변환

형 변환(Casting)



(a) 확대 형 변환



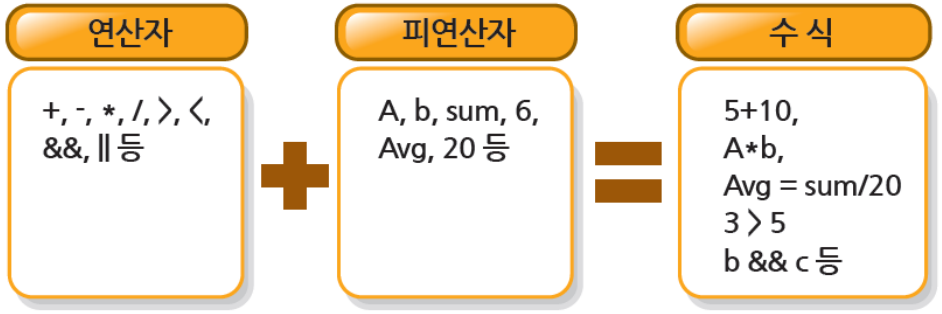
(b) 축소 형 변환

수업 목표

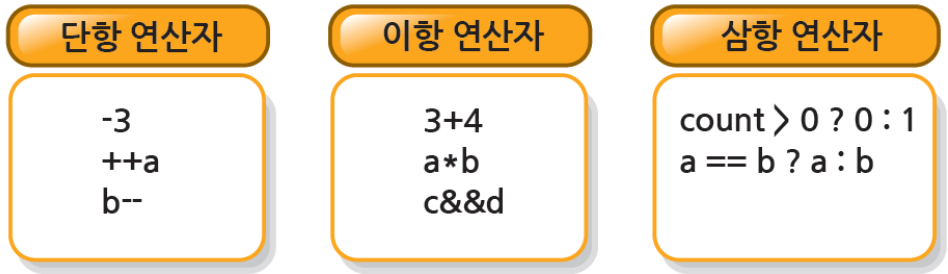
- 식별자와 예약어
- 변수
- 자료형
- 연산과 형 변환
- 연산자와 수식

연산자와 수식

- 수식은 연산자와 피연산자로 구성



- 연산자의 사용 형태는 3가지



연산자와 수식

자바의 산술 연산자

- 단항 연산자는 피연산자로 변수만 사용

연산자	사용법	설명	비고
+	op1+op2	op1과 op2를 더한다.	단항 및 이항
–	op–op2	op1에서 op2를 뺀다.	단항 및 이항
*	op1*op2	op1과 op2를 곱한다.	이항
/	op1/op2	op1을 op2로 나눈다.	이항
%	op1%op2	op1을 op2로 나눈 나머지를 구한다.	이항
+ +	var++ ++var	var 값 1 증가. var 값을 증가시키기 전에 평가 var 값 1 증가. var 값을 증가시킨 다음 평가	단항 단항
– –	var– – – –var	var 값 1 감소. var 값을 감소시키기 전에 평가 var 값 1 감소. var 값을 감소시킨 다음 평가	단항 단항

연산자와 수식

□ 예제 3.18 : 단항 연산자와 이항 연산자

예제 3.18 ArithmeticOPTest1.java

```
01: public class ArithmeticOPTest1 {
02:     public static void main(String args[])
03:     {
04:         int a=5, b=2 ;
05:         int sum=a+b;
06:         System.out.println("a+b=" + sum);
07:         int sub=a-b;
08:         System.out.println("a-b=" + sub);
09:         int mul=a*b;
10:         System.out.println("a*b=" + mul);
11:         int div=a/b;
12:         System.out.println("a/b=" + div);
13:         int mod=a%b;   <-- 두 값을 나눈 나머지를 구하는 연산자
14:         System.out.println("a%b=" + mod);
15:         int c = ++a;   <-- 단항 연산자값을 증가시킨 후에 배정
16:         System.out.println("a의 전위 증가 연산(prefix)="+c);
17:         System.out.println("a 변수의 값 : "+a);
18:         int d = b++;   <-- 단항 연산자값을 배정한 후에 1을 증가
19:         System.out.println("b의 후위 증가 연산(postfix)="+d);
20:         System.out.println("b 변수의 값 : "+b);
21:     }
22: }
```

실행 결과

```
a+b=7
a-b=3
a*b=10
a/b=2
a%b=1
a의 전위 증가 연산(prefix)=6
a 변수의 값 : 6
b의 후위 증가 연산(postfix)=2
b 변수의 값 : 3
```

연산자와 수식

◻ 예제 3.20 : 전위 연산과 후위 연산 방법

예제 3.17 ArithmeticOPTest3.java

```
01: public class ArithmeticOPTest3 {
02:     public static void main(String aa[])
03:     {
04:         int a = 10;
05:         System.out.println("(++a + ++a)의 결과는 : " + (++a + ++a));
06:         a = 10;                                          // 11+12가 되어 23
07:         System.out.println("(++a - ++a)의 결과는 : " + (++a - ++a));
08:         a = 10;                                          // 11-12가 되어 -1
09:         System.out.println("(a++ + a++)의 결과는 : " + (a++ + a++));
10:         a = 10;                                          // 10+11이 되어 21
11:         System.out.println("(a++ - a++)의 결과는 : " + (a++ - a++));
12:         a = 10;                                          // 10-11이 되어 -1
13:         System.out.println("(++a + a++)의 결과는 : " + (++a + a++));
14:         a = 10;                                          // 11+11이 되어 22
15:         System.out.println("(++a - a++)의 결과는 : " + (++a - a++));
16:         a = 10;                                          // 11-11이 되어 0
17:         System.out.println("(a++ + ++a)의 결과는 : " + (a++ + ++a));
18:         a = 10;                                          // 10+12가 되어 22
19:         System.out.println("(a++ - ++a)의 결과는 : " + (a++ - ++a));
20:     }                                                    // 10-12가 되어 -2
21: }
```

실행 결과

```
(++a + ++a)의 결과는 : 23
(++a - ++a)의 결과는 : -1
(a++ + a++)의 결과는 : 21
(a++ - a++)의 결과는 : -1
(++a + a++)의 결과는 : 22
(++a - a++)의 결과는 : 0
(a++ + ++a)의 결과는 : 22
(a++ - ++a)의 결과는 : -2
```

연산자와 수식

관계 및 논리 연산자

관계 연산자

- 두 개의 피연산자 값들을 비교하여 t 또는 f 값 반환
- 선택문과 반복문의 조건식에 사용
- 피연산자가 서로 다른 형인경우 : 큰 쪽으로 자동 변환

연산자	사용법	설명
〉	op1 〉 op2	op1이 op2보다 큰 경우
〉=	op1 〉= op2	op1이 op2보다 크거나 같은 경우
〈	op1 〈 op2	op1이 op2보다 작은 경우
〈=	op1 〈 op2	op1이 op2보다 작거나 같은 경우
==	op1 == op2	op1과 op2가 같은 경우
!=	op1 != op2	op1과 op2가 같지 않은 경우
instanceof	op1 instanceof op2	op1이 op2의 인스턴스(객체)인 경우

연산자와 수식

□ 예제 3.21 : 관계 연산자 사용 및 결과 확인

예제 3.18 RelationalOPTest.java

```
01: public class RelationalOPTest {
02:     public static void main(String args[])
03:     {
04:        byte a = 20;
05:         double d = 3.14;
06:         boolean flag;
07:         flag = a > d;   ← 관계 연산자 사용. 결과를 이진 변수에 저장
08:         System.out.println("a가 d보다 큰가? " + flag);
09:         flag = a == 20.0f;   ← byte형이 float형으로 자동 변환되어 비교
10:         System.out.println("a가 20.0f와 같은가? " + flag);
11:         flag = 10 != 10.0;   ← 정수 리터럴 10이 실수로 변환되어 비교
12:         System.out.println("10이 10과 같지 않은가? " + flag);
13:         flag = 10 <= 20;   ← 관계 연산자 사용
14:         System.out.println("10이 20보다 작거나 같은가? " + flag);
15:         System.out.println("10이 20보다 작은가? " + (10 < 20));
16:         System.out.println("10이 20보다 크거나 같은가? " + (10 >= 20));
17:     }
18: }
```

관계 연산자를 출력문에 직접 사용.

실행 결과

```
a가 d보다 큰가? true
a가 20.0f와 같은가? true
10이 10과 같지 않은가? false
10이 20보다 작거나 같은가? true
10이 20보다 작은가? true
10이 20보다 크거나 같은가? false
```

연산자와 수식

관계 및 논리 연산자

논리 연산자

- 두 개의 피연산자 값을 평가하여 t 또는 f 값 반환
- 두 개의 피연산자가 반드시 t 또는 f 값을 가져야 함
- 이항 논리 연산자 : &&(AND), !!(OR)

x	y	x||y	x && y
true	true	true	true
ture	false	true	false
false	true	true	false
false	false	false	false

- 단항 논리 연산자 : !(NOT)

x	!x
true	false
false	true

연산자와 수식

관계 및 논리 연산자

관계 연산자와 논리 연산자의 사용 예

```
boolean flag;
flag = 30 < 20;                  <-- 사용 가능
flag = 30 && 20;                 <-- 사용 불가능
flag = 30 < 20 || 50 > 20;       <-- 사용 가능
flag = 30 < 20 && 80;            <-- 사용 불가능
flag = 20 + 30 < 20 * 2;         <-- 사용 가능
flag = 20 + 30 || 20 * 2;        <-- 사용 불가능
```

연산자와 수식

□ 예제 3.22 : 관계 연산자와 논리 연산자 사용

예제 3.19 LogicalOPTest.java

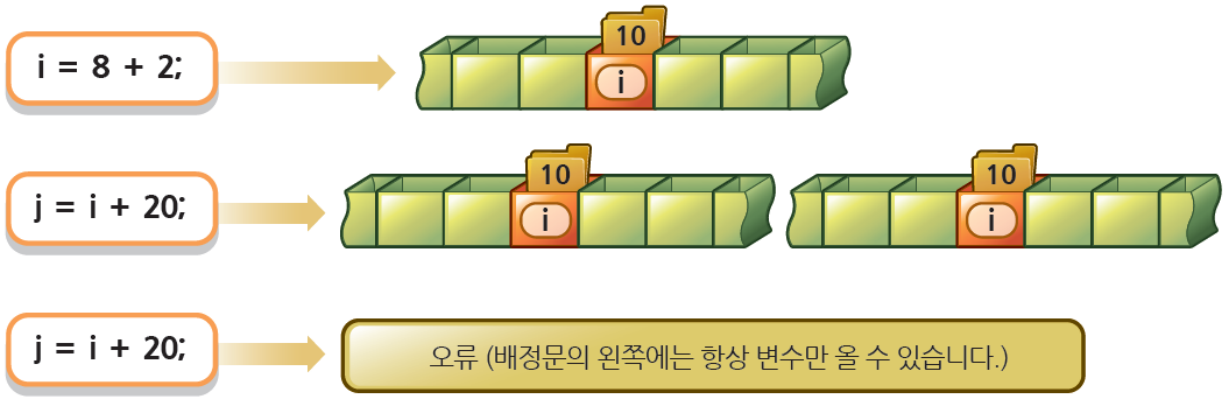
```
01: public class LogicalOPTest {
02:     public static void main(String args[])
03:     {
04:         boolean a;
05:         a = (20 > 10) || (30 > 40);   <-- OR 논리 연산자 수행
06:         System.out.println("20이 10보다 크거나 또는(논리합 ||) 30이 40보다 큰가? " +
    a);
07:         a = (20 > 10) && (30 > 40);   <-- AND 논리 연산자 수행
08:         System.out.println("20이 10보다 크고 그리고(논리곱 &&) 30이 40보다 큰가? " +
    a);
09:         a = ! true;   <-- NOT 논리 연산자 수행
10:         System.out.println("ture의 !(not)은? " + a);
11:         System.out.println("20이 10보다 크거나 또는(논리합 ||) 30이 40보다
    큰가? " + ((20 > 10) || (30 > 40)));
12:         System.out.println("20이 10보다 크고 그리고(논리곱 &&) 30이 40보다
    큰가? " + ((20 > 10) && (30 > 40)));
13:         System.out.println("ture의 !(not)은? " + (! true));
14:     }                                   출력문에 논리 연산자를 직접 사용
15: }
```

실행 결과

```
20이 10보다 크거나 또는(논리합 ||) 30이 40보다 큰가? true
20이 10보다 크고 그리고(논리곱 &&) 30이 40보다 큰가? false
ture의 !(not)은? false
20이 10보다 크거나 또는(논리합 ||) 30이 40보다 큰가? true
20이 10보다 크고 그리고(논리곱 &&) 30이 40보다 큰가? false
ture의 !(not)은? false
```

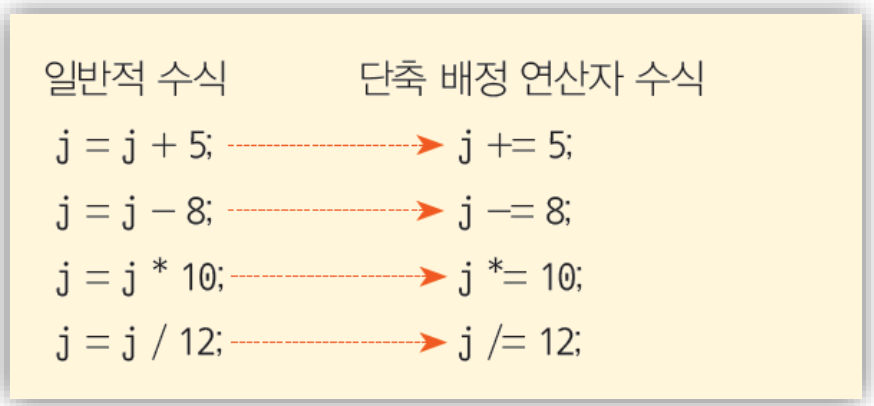
연산자와 수식

- 배정 연산자와 단축 배정 연산자
  - 자바의 배정 연산자 "="
    - 배정 연산자의 왼쪽에는 반드시 변수만 올 수 있다.



연산자와 수식

- 단축 배정 연산자 : 배정 연산자와 다른 연산자를 같이 사용할 수 있다.



연산자와 수식

- 단축 배정

연산자	사용법	의미
+=	op1 += op2	op1 = op1 + op2
-=	op1 -= op2	op1 = op1 - op2
*=	op1 *= op2	op1 = op1 * op2
/=	op1 /= op2	op1 = op1 / op2
%=	op1 %= op2	op1 = op1 % op2
&=	op1 &= op2	op1 = op1 & op2
|=	op1 |= op2	op1 = op1 | op2
^=	op1 ^= op2	op1 = op1 ^ op2
〈〈=	op1 〈〈= op2	op1 = op1 〈〈 op2
〉〉=	op1 〉〉= op2	op1 = op1 〉〉 op2
〉〉〉=	op1 〉〉〉= p2	op1 = op1 〉〉〉 op2

첫 번째 프로그램

예제 프로그램 1

```java
//Scanner 클래스를 사용하기 위한 Import
import java.util.Scanner;

class FirstMain {

    public static void main(String[] args) {
        // 변수 선언
        Scanner s = new Scanner(System.in);

        // 사용자 보여주는 출력 메세지
        System.out.println("사과의 개수는 몇 개입니까?");

        // 정수값으로 입력받음.
        int index = s.nextInt();

        //
        System.out.println("입력하신 사과의 개수는 "+ index + " 입니다.");
    }
}
```

결과

```
사과의 개수는 몇 개입니까?
10
입력하신 사과의 개수는 10 입니다.
```

두 번째 프로그램

- 첫 번째 프로그램을 우측의 결과가 나오도록 수정하기

예제 프로그램 2

```java
//Scanner 클래스를 사용하기 위한 Import
import java.util.Scanner;

class FirstMain {

    public static void main(String[] args) {
        // 변수 선언
        Scanner s = new Scanner(System.in);

        // 사용자 보여주는 출력 메세지
        System.out.println("사과의 개수는 몇 개입니까?");

        // 정수값으로 입력받음.
        int index = s.nextInt();

        //
        System.out.println("입력하신 사과의 개수는 "+ index + " 입니다.");
    }
}
```

```
사과의 개수는 몇 개입니까?
5
사과의 가격은 얼마입니까?
1000
사과의 가격은 5000원 입니다.
```

실습 문제 1 – 나이 계산

실습 문제 (1/6)

- 태어난 년도를 입력 받아 나이를 계산하고 출력하라. 단, 나이 = “현재 년도” – “태어난 년도” + 1로 계산한다.
  변수는 다음과 같이 사용하라.

```
birth_year          // 태여난 년도
age                 // 나이
```

결과 (2019년 기준)
태어난 년도를 입력하세요. **1995** 당신의 나이는 25 입니다.

실습 문제 2 – 온도 변환

실습 문제 (2/6)

- 섭씨 온도를 입력 받아, 화씨 온도로 변환한 후 출력하라.
  단, 화씨 온도 = 섭씨 온도 * 1.8 + 32로 계산한다.
  변수는 다음과 같이 사용하라.

```
c_degree        // 섭씨 온도
f_degree        // 화씨 온도
```

결과
섭씨 온도를 입력하세요. **20.5** 화씨 온도는 68.9도 입니다.

실습 문제 3 – 직사각형 넓이 계산

실습 문제 (3/6)

- 직사각형의 가로/세로 길이를 입력 받아 넓이를 계산하고 출력하라. 단, 직사각형의 넓이 = 가로 길이 * 세로 길이 로 계산한다.
  변수는 다음과 같이 사용하라.

```
width       // 가로 길이
height      // 세로 길이
area        // 직사각형의 넓이
```

결과
직사각형의 가로 길이를 입력하세요. **10** 직사각형의 세로 길이를 입력하세요. **20** 직사각형의 넓이는 200 입니다.

실습 문제 4 – 아파트 평형 계산

실습 문제 (4/6)

- 아파트의 분양 면적을 제곱미터()로 입력 받아 평형 단위의 값으로 변환하여 출력하라. 단, 평형 수 = 제곱미터 / 3.305로 계산한다.
  변수는 다음과 같이 사용하라.

```
m2_area          // 면적 (제곱미터)
pyung_area       // 면적 (평수)
```

결과
아파트의 분양 면적을 입력하세요. **105.5** 아파트의 평형은 31.9 입니다.

실습 문제 5 – 날짜 계산

실습 문제 (5/6)

- 날 수를 입력 받아 초로 변환하여 출력하라. 단, 초 = 날 수 * 24 * 60 * 60으로 계산한다.
  변수는 다음과 같이 사용하라.

```
days                    // 날 수
seconds                 // 초 단위 시간
```

결과
날 수를 입력하세요. **25** 날 수에 해당되는 시간은 모두 2160000초 입니다.

실습 문제 6 – 점수 계산

실습 문제 (6/6)

- 국어, 영어, 수학 점수를 입력받아 총점과 평균을 계산하고 출력하라. 단, 총점 = 국어+영어+수학, 평균 = 총점/3.0으로 계산하라.
  변수는 다음과 같이 사용하라.

```
kor       // 국어 점수
eng       // 영어 점수
math      // 수학 점수
total     // 총점
avg       // 평균 점수
```

결과

```
국어 점수를 입력하세요. 85
영어 점수를 입력하세요. 95
수학 점수를 입력하세요. 80
입력하신 점수의 총점은 260이고,
평균은 86.7 입니다.
```

Q & A

- Thank you for your attention