

# 자바 기초 1

---

장서윤 [pineai@cnu.ac.kr](mailto:pineai@cnu.ac.kr)

# 수업 안내

---

- ▶ JAVA를 배운 적이 있는 학생을 대상으로 하는 기초 수업임
  - ▶ JAVA를 처음 배우는 학생의 경우 다른 특강이 개설 될 예정
- ▶ 강의자료
  - ▶ [winslab.cnu.ac.kr/lunar](http://winslab.cnu.ac.kr/lunar)

# JDK, JRE

---

## ➤ JDK(Java Development Kit)

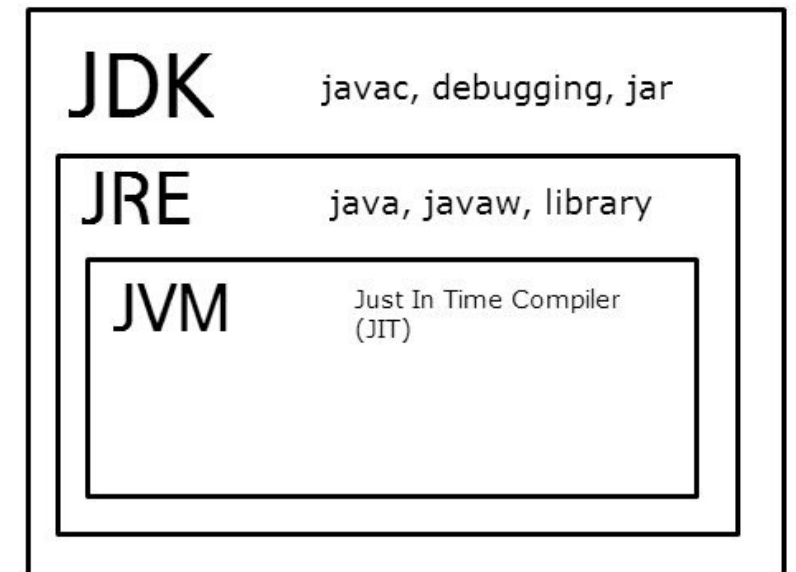
- Java SE의 표준안에 따라서 만들어진 구체적인 소프트웨어
- Java 코드를 컴파일하는 컴파일러
- 개발에 필요한 각종 도구
- JRE가 포함
- 개발자를 위한 자바 버전

## ➤ JRE(Java Runtime Environment)

- 자바가 실제로 동작하는 데 필요한 JVM, 라이브러리, 각종 파일들이 포함
- 자바로 만들어진 프로그램을 구동할 때 필요
- 일반인을 위한 자바 버전

## ➤ JVM(Java Virtual Machine)

- JVM은 자바가 실제로 구동하는 환경



# JAVA의 버전

---

- Java SE (Standard Edition)
  - 자바의 핵심
- Java EE (Enterprise Edition)
  - 기업용 시장에서 사용하는 자바 개발환경
- Java ME (Micro Edition)
  - 모바일 개발을 위해서 사용하는 자바 버전



# JAVA의 설치

---

## ➤ Java SE JDK를 설치

- <http://www.oracle.com/technetwork/java/javase/downloads/index.html>



The screenshot shows the Oracle Java SE Downloads page. At the top is the Oracle logo and navigation links: Sign In/Register, Help, Country, and Communities. Below this are links for Products, Solutions, and Downloads. A breadcrumb trail reads: Oracle Technology Network > Java > Java SE > Downloads. On the left is a sidebar menu with links to Java SE, Java EE, Java ME, Java SE Support, Java SE Advanced & Suite, Java Embedded, Java DB, Web Tier, Java Card, and Java TV. The main content area has tabs for Overview, Downloads (selected), Documentation, and Co. The title is 'Java SE Downloads'. Below the title is the Java logo and a large 'DOWNLOAD' button with a download icon. At the bottom, it specifies 'Java Platform (JDK) 8u111 / 8u112'.

ORACLE

Sign In/Register Help Country Communities

Products Solutions Downloads

Oracle Technology Network > Java > Java SE > Downloads

Java SE  
Java EE  
Java ME  
Java SE Support  
Java SE Advanced & Suite  
Java Embedded  
Java DB  
Web Tier  
Java Card  
Java TV

Overview Downloads Documentation Co

**Java SE Downloads**



**DOWNLOAD** ↓

Java Platform (JDK) 8u111 / 8u112

# JAVA의 설치

.....

## Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- [Java Developer Newsletter](#): From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- [Java Developer Day](#) hands-on workshops (free) and other events
- [Java Magazine](#)

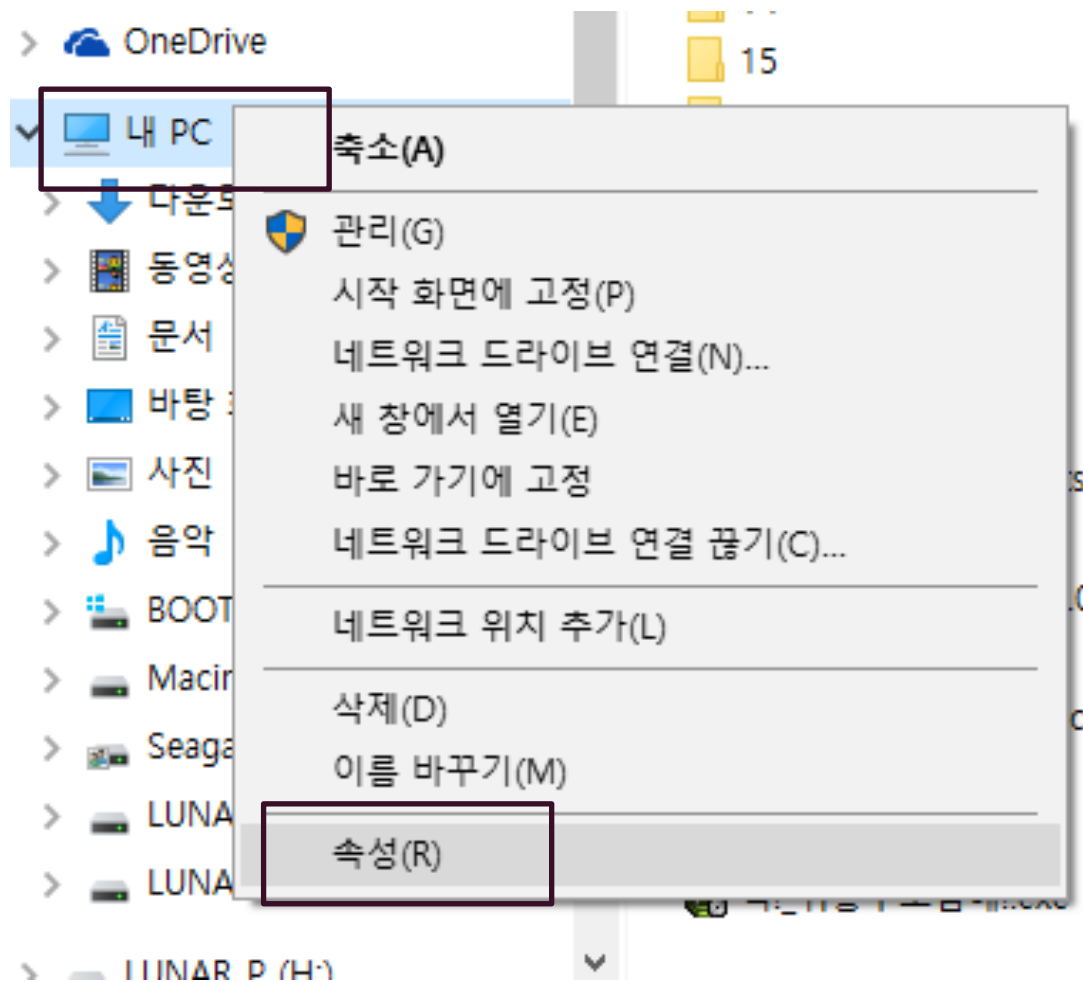
[JDK 8u111 Checksum](#)

[JDK 8u112 Checksum](#)

Java SE Development Kit 8u111		
You must accept the <a href="#">Oracle Binary Code License Agreement for Java SE</a> to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.78 MB	<a href="#">jdk-8u111-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM 64 Hard Float ABI	74.73 MB	<a href="#">jdk-8u111-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	160.35 MB	<a href="#">jdk-8u111-linux-i586.rpm</a>
Linux x86	175.04 MB	<a href="#">jdk-8u111-linux-i586.tar.gz</a>
Linux x64	158.35 MB	<a href="#">jdk-8u111-linux-x64.rpm</a>
Linux x64	173.04 MB	<a href="#">jdk-8u111-linux-x64.tar.gz</a>
Mac OS X	227.39 MB	<a href="#">jdk-8u111-macosx-x64.dmg</a>
Solaris SPARC 64-bit	131.92 MB	<a href="#">jdk-8u111-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	93.02 MB	<a href="#">jdk-8u111-solaris-sparcv9.tar.gz</a>
Solaris x64	140.38 MB	<a href="#">jdk-8u111-solaris-x64.tar.Z</a>
Solaris x64	96.82 MB	<a href="#">jdk-8u111-solaris-x64.tar.gz</a>
Windows x86	189.22 MB	<a href="#">jdk-8u111-windows-i586.exe</a>
Windows x64	194.64 MB	<a href="#">jdk-8u111-windows-x64.exe</a>

# JAVA 실행 확인

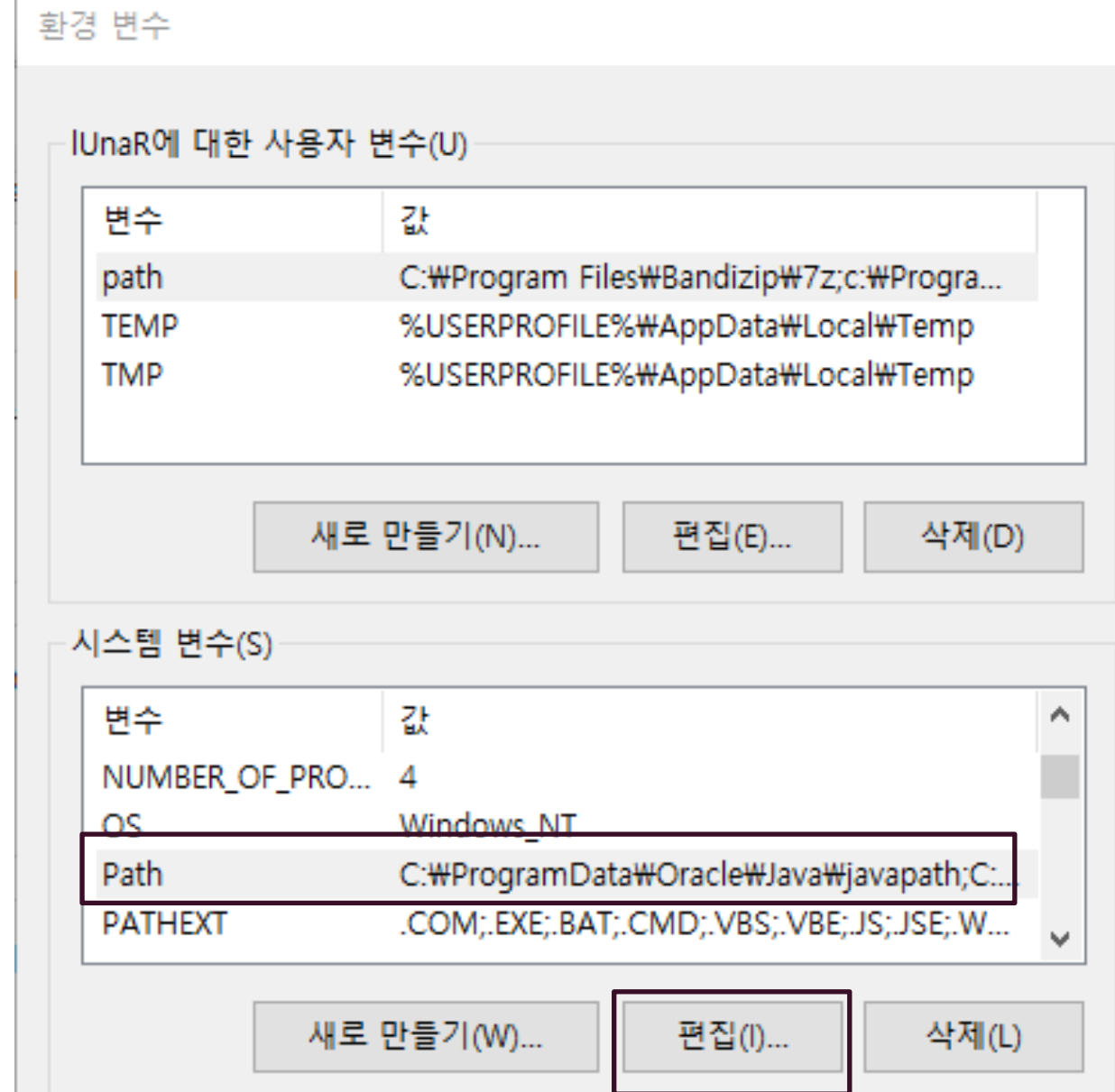
.....



설정 변경

제품 키 변경

# JAVA 실행 확인





JDK 설치 위치

BOOTCAMP (C:) > Program Files > Java > jdk1.8.0\_101 > bin

환경 변수 편집

C:\ProgramData\Oracle\Java\javapath  
%SystemRoot%\system32  
%SystemRoot%  
%SystemRoot%\System32\Wbem  
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\  
C:\Program Files (x86)\MATLAB\R2014a\bin  
C:\Program Files\Java\jdk1.8.0\_101\bin

새로 만들기(N)

편집(E)

찾아보기(B)...

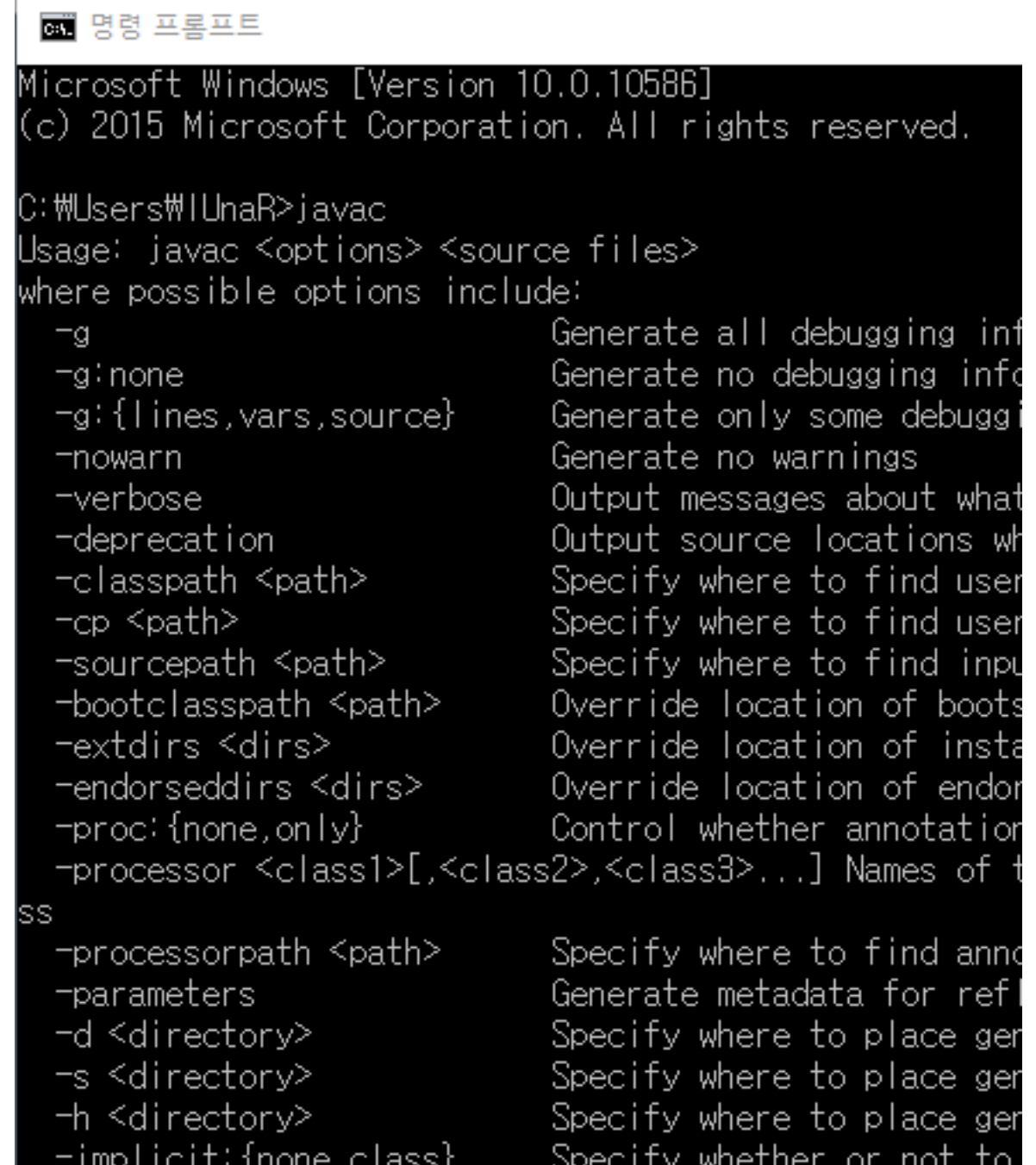
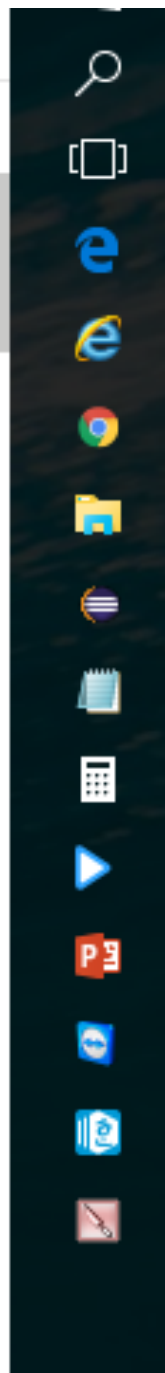
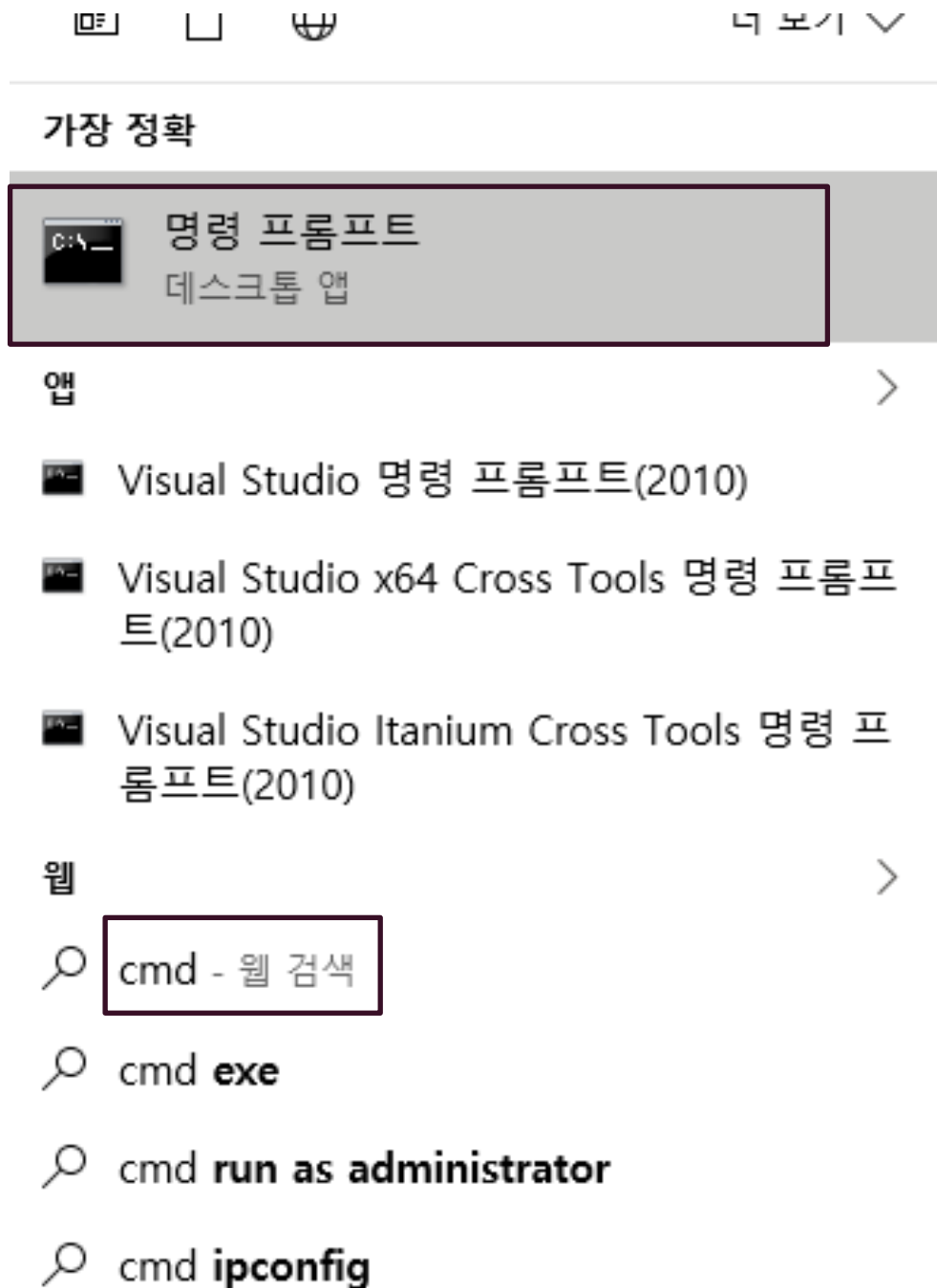
삭제(D)

설치 위치 입력

bin 검색

16KB  
17KB  
17KB  
34KB  
16KB  
17KB  
203KB  
16KB

# JAVA 실행 확인



# JAVA 코드 작성

---

## ➤ 코드

- 자바의 문법에 맞게 만들어진 텍스트 파일
- 확장자는 [이름].java

## ➤ 간단한 에디터를 사용하여 코딩 (coding - 코드를 만듦)이 가능

# JAVA 코드 작성

---

## ➤ Helloworld.java

```
class Helloworld {  
    public static void main(String[] args){  
        System.out.println("Hello world");  
    }  
}
```

# JAVA 컴파일(COMPILE)

---

- ▶ 코드를 컴퓨터가 이해할 수 있는 상태로 변환해주는 과정
- ▶ 컴파일러 (compiler) - 컴파일을 해주는 소프트웨어
  - ▶ 자바의 컴파일러는 javac

```
javac Helloworld.java
```

# JAVA 실행

---

- ▶ 실행 파일 - [이름].class
- ▶ 실행시 .class는 입력하지 않는다.

```
java HelloWorld
```

# JAVA 실행

---



# 개발 도구

---

- ▶ 이클립스

- ▶ 가장 대표적인 자바의 개발도구

- ▶ IDE - **I**ntegrated **D**evelopment **E**nvironment

- ▶ 오픈소스

- ▶ 무료

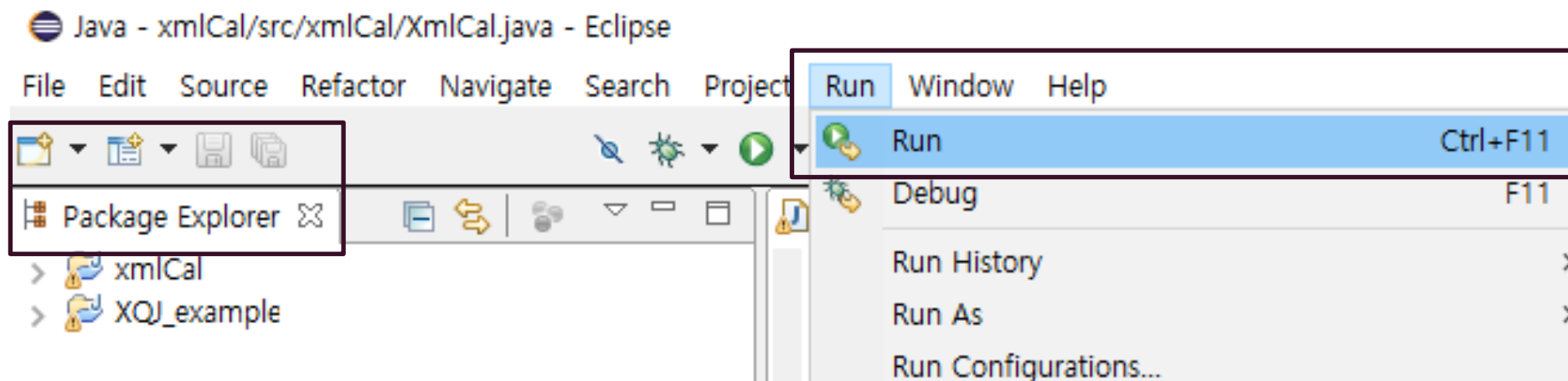
- ▶ 모든 운영 체제를 지원

- ▶ <http://www.eclipse.org/downloads/>



# 이클립스

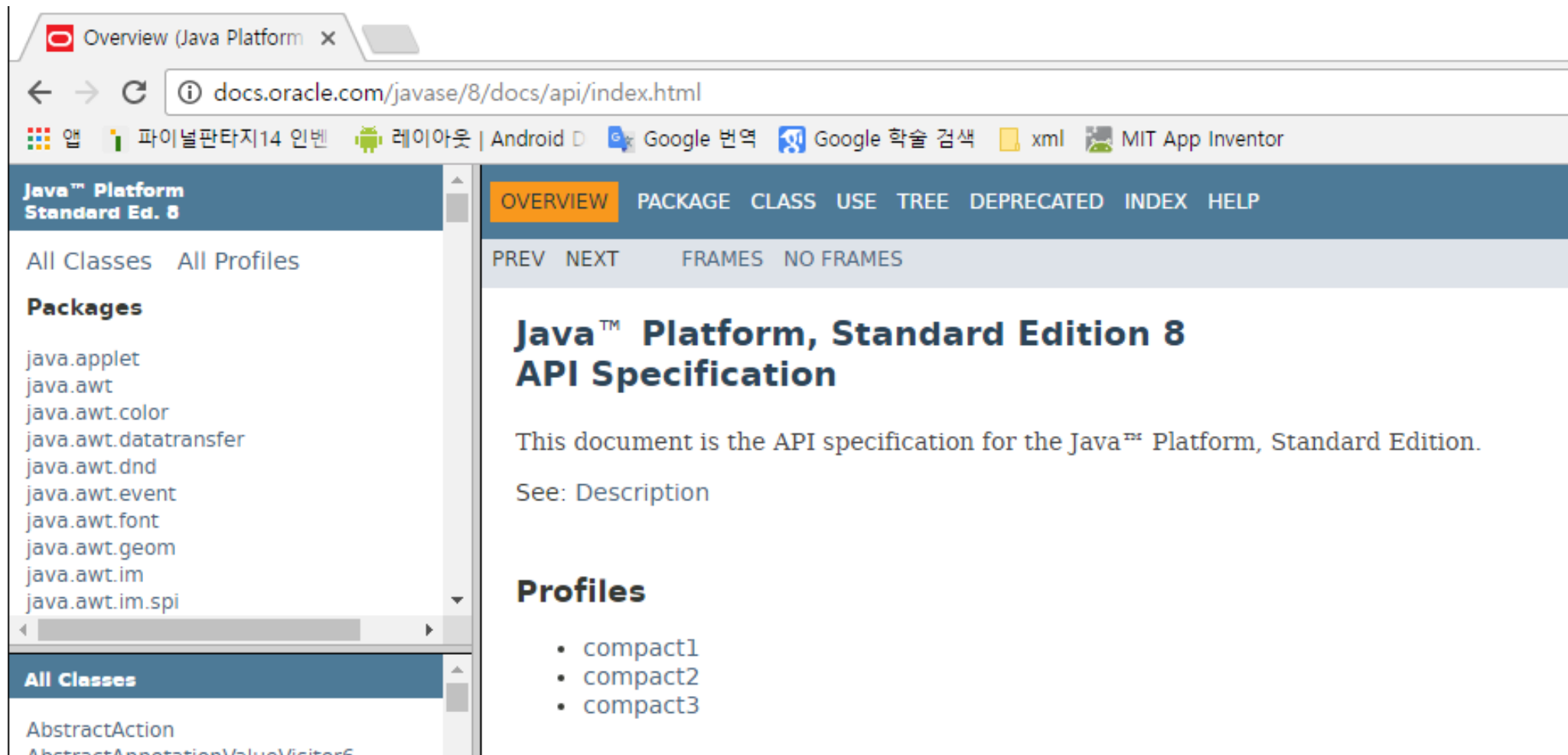
---



# 자바 DOCUMENT / API SPECIFICATION

---

➤ <http://docs.oracle.com/javase/8/docs/api/index.html>



# 변수

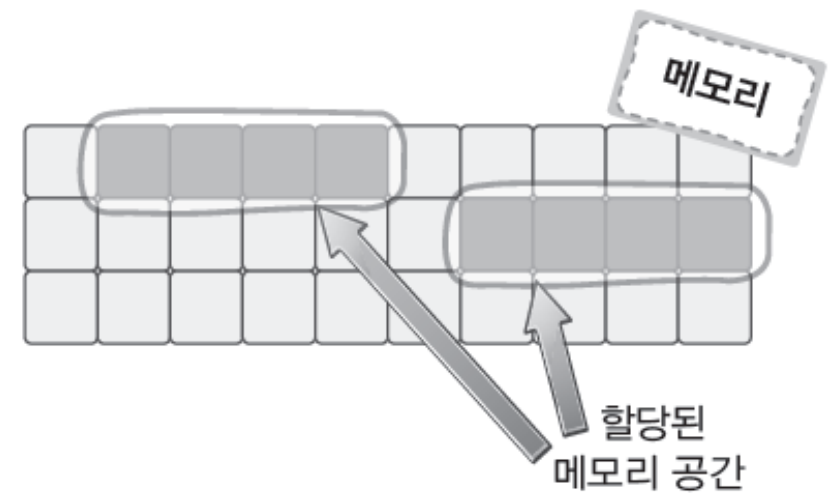
---

## ✓ 메모리 공간의 활용에 필요한 두 가지 요소

- 데이터 저장을 위한 메모리 공간의 할당 방법
- 할당된 메모리 공간의 접근(저장 및 참조) 방법



'변수(Variable)'라는 것을 통해 이 두가지 방법을 모두 제공



# 변수(VARIABLE)에 대한 간단한 이해

---

"난 10진수 정수의 저장을 위한  
메모리 공간을 할당하겠다."  
"그리고 그 메모리 공간의 이름을  
num이라 하겠다."

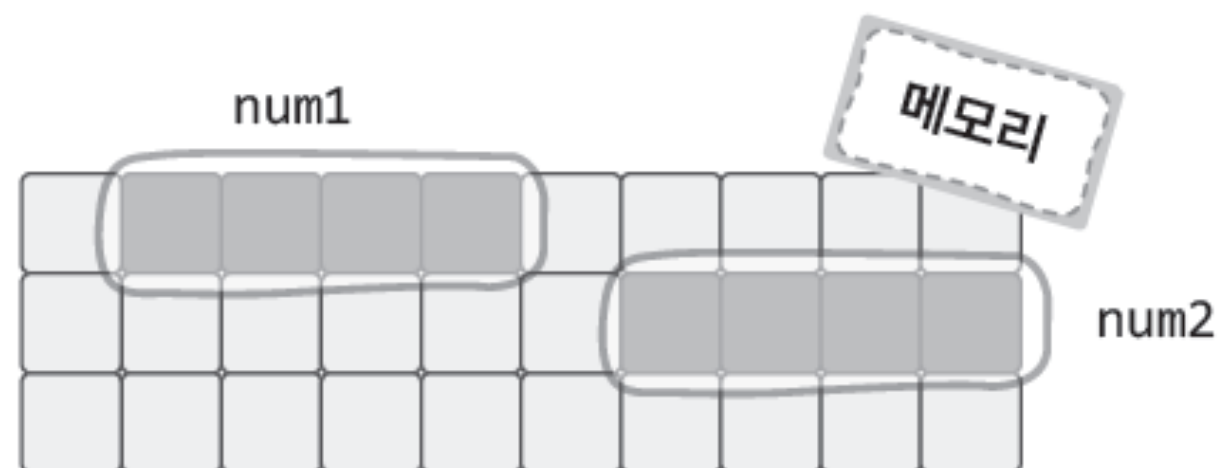


```
int num;
```

변수의 선언

int와 같이 변수의 특성을 결정짓는 키워드를 가리켜 자료형이라 한다.

```
int num1;  
int num2;
```



# 변수의 선언

---

```
class UseVariable
{
    public static void main(String[] args)
    {
        int num1;
        num1=10;

        int num2=20;
        int num3=num1+num2;
        System.out.println(num1+" "+num2+"="+num3);
    }
}
```

실행결과

10+20=30

# 자료형의 종류와 구분

자료형	데이터	메모리 크기	표현 가능 범위
boolean	참과 거짓	1 바이트	true, false
char	문자	2 바이트	모든 유니코드 문자
byte	정수	1 바이트	-128 ~ 127
short		2 바이트	-32768 ~ 32767
int		4 바이트	-2147483648 ~ 2147483647
long		8 바이트	-9223372036854775808 ~ 9223372036854775807
float	실수	4 바이트	$\pm(1.40 \times 10^{-45} \sim 3.40 \times 10^{38})$
double		8 바이트	$\pm(4.94 \times 10^{-324} \sim 1.79 \times 10^{308})$

- 정수 표현                    byte, short, int, long
- 실수 표현                    float, double
- 문자 표현                    char
- 참과 거짓의 표현        boolean

## 자료형의 분류

# 변수의 이름을 짓는 방법

---

## ✓ 변수 이름의 제약사항

- 숫자로 시작 불가
- \$와 \_ 이외의 다른 특수문자는 사용 불가
- 키워드는 변수의 이름으로 사용 불가

## 자바의 키워드

<code>boolean</code>	<code>if</code>	<code>interface</code>	<code>class</code>	<code>true</code>
<code>char</code>	<code>else</code>	<code>package</code>	<code>volatile</code>	<code>false</code>
<code>byte</code>	<code>final</code>	<code>switch</code>	<code>while</code>	<code>throws</code>
<code>float</code>	<code>private</code>	<code>case</code>	<code>return</code>	<code>native</code>
<code>void</code>	<code>protected</code>	<code>break</code>	<code>throw</code>	<code>implements</code>
<code>short</code>	<code>public</code>	<code>default</code>	<code>try</code>	<code>import</code>
<code>double</code>	<code>static</code>	<code>for</code>	<code>catch</code>	<code>synchronized</code>
<code>int</code>	<code>new</code>	<code>continue</code>	<code>finally</code>	<code>const</code>
<code>long</code>	<code>this</code>	<code>do</code>	<code>transient</code>	<code>enum</code>
<code>abstract</code>	<code>super</code>	<code>extends</code>	<code>instanceof</code>	<code>null</code>

# 상수?

---

## ✓ 표현되는 데이터는 상수 아니면 변수

- `int num = 1 + 5;`
- `System.out.println(2.4 + 7.5);`

## ✓ 상수와 변수의 비교

- 변수와 마찬가지로 상수도 메모리 공간에 저장이 된다.
- 다만 이름이 존재하지 않으니 값의 변경이 불가능하다.
- 상수는 존재 의미가 없어지면 바로 소멸된다.



# 자료형의 변환

```
int main(String[] args)
{
    short num1=10;
    short num2=20;
    short result = num1 + num2;
    . . . .
}
```

✓ 자료형의 변환은 표현방법의 변환

num1(10) → 00000000 00001010

num2(20) → 00000000 00010100

↓ short to int

int형 정수 10 → 00000000 00000000 00000000 00001010

int형 정수 20 → 00000000 00000000 00000000 00010100

int형 정수 1 → 00000000 00000000 00000000 00000001

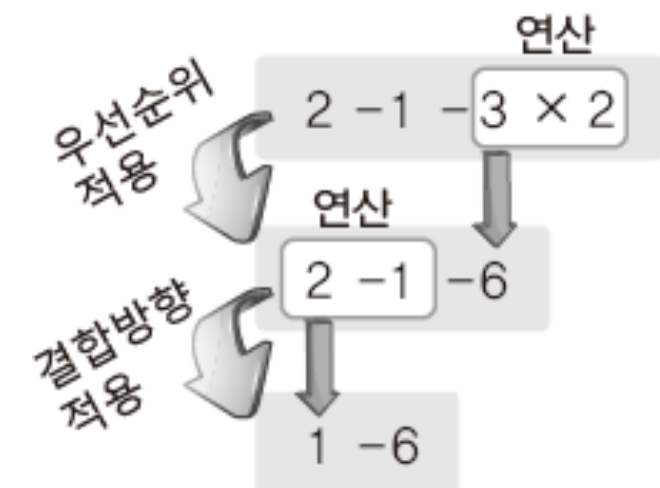
↓ short to int

float형 실수 1.0 → 00111111 10000000 00000000 00000000

# 자바의 연산자와 연산의 과정

연산기호	결합방향	우선순위
[ ], .	➡	1(높음)
expr++, expr--	⬅	2
++expr, --expr, +expr, -expr, ~, !, (type)	⬅	3
*, /, %	➡	4
+, -	➡	5
<<, >>, >>>	➡	6
<, >, <=, >=, instanceof	➡	7
==, !=	➡	8
&	➡	9
^	➡	10
	➡	11
&&	➡	12
	➡	13
? expr : expr	⬅	14
=, +=, -=, *=, /=, %=, &=, ^=,  =, <<=, >>=, >>>=	⬅	15(낮음)

## 연산의 과정



# 대입 연산자(=)와 산술 연산자(+, -, \*, /, %)

.....

연산자	연산자의 기능	결합방향
=	연산자 오른쪽에 있는 값을 연산자 왼쪽에 있는 변수에 대입한다. 예) val = 20;	←
+	두 피연산자의 값을 더한다. 예) val = 4 + 3;	→
-	왼쪽의 피연산자 값에서 오른쪽의 피연산자 값을 뺀다. 예) val = 4 - 3;	→
*	두 피연산자의 값을 곱한다. 예) val = 4 * 3;	→
/	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눈다. 예) val = 7 / 3;	→
%	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눴을 때 얻게 되는 나머지를 반환한다. 예) val = 7 % 3	→

# 대입연산과 산술연산의 예

---

```
class ArithOp
{
    public static void main(String[] args)
    {
        int n1=7;
        int n2=3;

        int result=n1+n2;
        System.out.println("덧셈 결과 : " + result);

        result=n1-n2;
        System.out.println("뺄셈 결과 : " + result);
        System.out.println("곱셈 결과 : " + n1*n2);
        System.out.println("나눗셈 결과 : " + n1/n2);
        System.out.println("나머지 결과 : " + n1%n2);
    }
}
```

## 실행결과

```
덧셈 결과 : 10
뺄셈 결과 : 4
곱셈 결과 : 21
나눗셈 결과 : 2
나머지 결과 : 1
```

# 복합대입 연산자

---

$a = a + b$	⇐ 동일 연산 ⇒	$a += b$
$a = a - b$	⇐ 동일 연산 ⇒	$a -= b$
$a = a * b$	⇐ 동일 연산 ⇒	$a *= b$
$a = a / b$	⇐ 동일 연산 ⇒	$a /= b$
$a = a \% b$	⇐ 동일 연산 ⇒	$a \% = b$

해석의 원칙은 동일

$\&=$ ,  $\wedge=$ ,  $|=$ ,  $\ll=$ ,  $\gg=$ ,  $\gg\gg=$

# 복합대입 연산자

---

```
class Comp
{
    public static void main(String[] args)
    {
        double e=3.1;
        e+=2.1;
        e*=2;
        int n=5;
        n*=2.2;
        System.out.println(e);
        System.out.println(n);
    }
}
```

실행결과

10.4

11

# 관계 연산자

.....

연산자	연산자의 기능	결합방향
<	예) $n1 < n2$ n1이 n2보다 작은가?	→
>	예) $n1 > n2$ n1이 n2보다 큰가?	→
<=	예) $n1 <= n2$ n1이 n2보다 같거나 작은가?	→
>=	예) $n1 >= n2$ n1이 n2보다 같거나 큰가?	→
==	예) $n1 == n2$ n1과 n2가 같은가?	→
!=	예) $n1 != n2$ n1과 n2가 다른가?	→

연산의 결과로 true or false 반환

# 관계연산

---

```
class CmpOp
{
    public static void main(String[] args)
    {
        int A=10, B=20;
        if(true)
            System.out.println("참 입니다!");
        else
            System.out.println("거짓 입니다!");

        if(A>B)
            System.out.println("A가 더 크다!");
        else
            System.out.println("A가 더 크지 않다!");

        if(A!=B)
            System.out.println("A와 B는 다르다!");
        else
            System.out.println("A와 B는 같다!");
    }
}
```

## 실행결과

```
참 입니다!
A가 더 크지 않다!
A와 B는 다르다!
```



# 논리 연산자

.....

## 연산의 결과로 true or false 반환

연산자	연산자의 기능	결합방향
&&	예) A && B A와 B 모두 true이면 연산결과는 true (논리 AND)	➡
	예) A    B A와 B 둘 중 하나라도 true이면 연산결과는 true (논리 OR)	➡
!	예) !A 연산결과는 A가 true이면 false, A가 false이면 true (논리 NOT)	⬅

# 논리 연산자

---

```
class LogicOp
{
    public static void main(String[] args)
    {
        int num1=10, num2=20;
        boolean result1=(num1==10 && num2==20);
        boolean result2=(num1<=12 || num2>=30);

        System.out.println("num1==10 그리고 num2==20 : " + result1);
        System.out.println("num1<=12 또는 num2>=30 : " + result2);

        if(!(num1==num2))
            System.out.println("num1과 num2는 같지 않다.");
        else
            System.out.println("num1과 num2는 같다.");
    }
}
```

## 실행결과

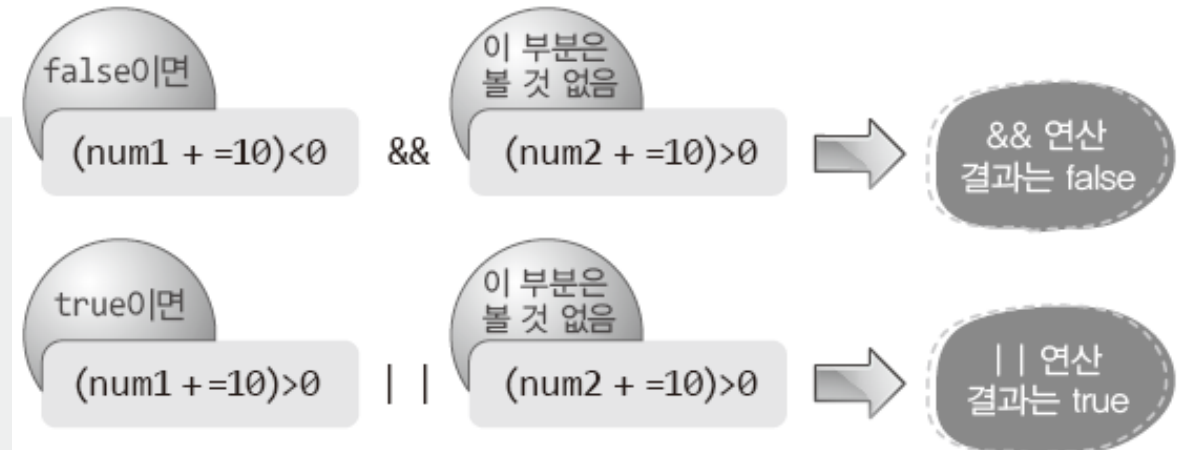
```
num1==10 그리고 num2==20 : true
num1<=12 또는 num2>=30 : true
num1과 num2는 같지 않다.
```

# 논리 연산자와 SCE(SHORT-CIRCUIT EVALUATION)

```
class SCE
{
    public static void main(String[] args)
    {
        int num1=0, num2=0;
        boolean result;

        result = (num1+=10)<0 && (num2+=10)>0;
        System.out.println("result="+result);
        System.out.println("num1="+num1+", num2="+num2);

        result = (num1+=10)>0 || (num2+=10)>0;
        System.out.println("result="+result);
        System.out.println("num1="+num1+", num2="+num2);
    }
}
```



```
result=false
num1=10, num2=0
result=true
num1=20, num2=0
```

실행결과

# 부호 연산자로서의 +와 -

---

## ✓ 연산자의 기능

- 단항 연산자로서 -는 부호를 바꾸는 역할을 한다.
- 단항 연산자로서 +는 특별히 하는 일이 없다.

```
class UnaryAddMin
{
    public static void main(String[] args)
    {
        int n1 = 5;
        System.out.println(+n1);
        System.out.println(-n1);

        short n2 = 7;
        int n3 = +n2;
        int n4 = -n2;
        System.out.println(n3);
        System.out.println(n4);
    }
}
```

## 실행결과

```
5
-5
7
-7
```

# 증가, 감소 연산자

연산자	연산자의 기능	결합방향
++ (prefix)	피연산자에 저장된 값을 1 증가 예) val = ++n;	←
-- (prefix)	피연산자에 저장된 값을 1 감소 예) val = --n;	←
연산자	연산자의 기능	결합방향
++ (postfix)	피연산자에 저장된 값을 1 증가 예) val = n++;	←
-- (postfix)	피연산자에 저장된 값을 1 감소 예) val = n--;	←



# 증가 감소 연산

---

```
class PrefixOp
{
    public static void main(String[] args)
    {
        int num1 = 7;
        int num2, num3;

        num2 = ++num1;
        num3 = --num1;

        System.out.println(num1);
        System.out.println(num2);
        System.out.println(num3);
    }
}
```

실행결과

7  
8  
7

```
class PostfixOp
{
    public static void main(String[] args)
    {
        int num1 = 7;
        int num2, num3;

        num2 = num1++;
        num3 = num1--;

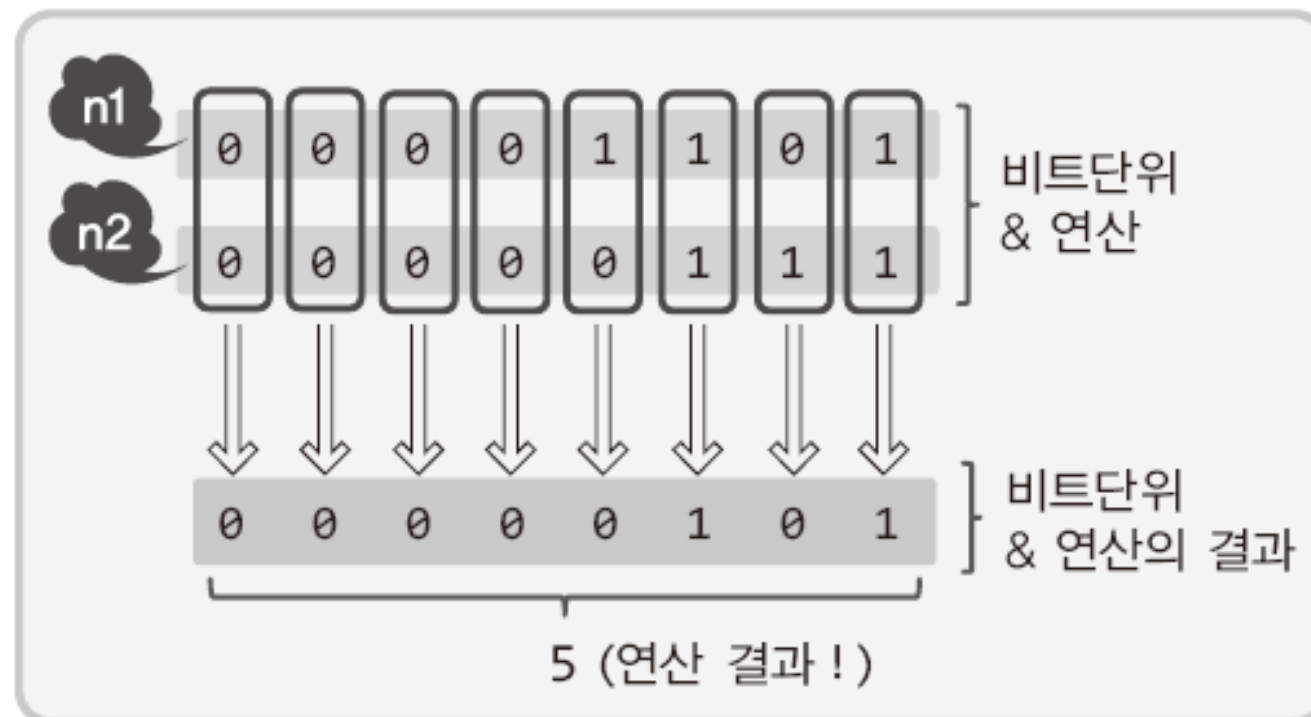
        System.out.println(num1);
        System.out.println(num2);
        System.out.println(num3);
    }
}
```

실행결과

7  
7  
8

# 비트 연산자

연산자	연산자의 기능	결합방향
&	비트단위로 AND 연산을 한다. 예) $n1 \& n2$ ;	→
	비트단위로 OR 연산을 한다. 예) $n1   n2$ ;	→
^	비트단위로 XOR 연산을 한다. 예) $n1 ^ n2$ ;	→
~	피연산자의 모든 비트를 반전시켜서 얻은 결과를 반환 예) $\sim n$ ;	←



# 비트연산 진리 표

비트 A	비트 B	비트 A & 비트 B
1	1	1
1	0	0
0	1	0
0	0	0

&(and)

비트 A	비트 B	비트 A   비트 B
1	1	1
1	0	1
0	1	1
0	0	0

| (OR)

비트 A	비트 B	비트 A ^ 비트 B
1	1	0
1	0	1
0	1	1
0	0	0

^(XOR)

비트	~비트
1	0
0	1

~(NOT)



# 비트 쉬프트(SHIFT) 연산자

연산자	연산자의 기능	결합방향
<<	<ul style="list-style-type: none"> <li>피연산자의 비트 열을 왼쪽으로 이동</li> <li>이동에 따른 빈 공간은 0으로 채움</li> <li>예) <math>n \ll 2</math>; → n의 비트 열을 두 칸 왼쪽으로 이동 시킨 결과 반환</li> </ul>	→
>>	<ul style="list-style-type: none"> <li>피연산자의 비트 열을 오른쪽으로 이동</li> <li>이동에 따른 빈 공간은 음수의 경우 1, 양수의 경우 0으로 채움</li> <li>예) <math>n \gg 2</math>; → n의 비트 열을 두 칸 오른쪽으로 이동 시킨 결과 반환</li> </ul>	→
>>>	<ul style="list-style-type: none"> <li>피연산자의 비트 열을 오른쪽으로 이동</li> <li>이동에 따른 빈 공간은 0으로 채움</li> <li>예) <math>n \ggg 2</math>; → n의 비트 열을 두 칸 오른쪽으로 이동 시킨 결과 반환</li> </ul>	→

## ✓ 비트연산의 특징

- 왼쪽으로의 비트 열 이동은 2의 배수의 곱
- 오른쪽으로의 비트 열 이동은 2의 배수의 나눗셈

- 정수 2 → 00000010 → 정수 2
- $2 \ll 1 \rightarrow 00000100 \rightarrow$  정수 4
- $2 \ll 2 \rightarrow 00001000 \rightarrow$  정수 8
- $2 \ll 3 \rightarrow 00010000 \rightarrow$  정수 16

# 비트 쉬프트(SHIFT) 연산

---

```
class BitShiftOp
{
    public static void main(String[] args)
    {
        System.out.println(2 << 1);    // 4 출력
        System.out.println(2 << 2);    // 8 출력
        System.out.println(2 << 3);    // 16 출력

        System.out.println(8 >> 1);    // 4 출력
        System.out.println(8 >> 2);    // 2 출력
        System.out.println(8 >> 3);    // 1 출력
        System.out.println(-8 >> 1);   // -4 출력
        System.out.println(-8 >> 2);   // -2 출력
        System.out.println(-8 >> 3);   // -1 출력

        System.out.println(-8 >>> 1);  // 2147483644 출력
    }
}
```

# IF문과 IF~ELSE문

---

```
if (true or false)
{
    /* true 시 실행되는 영역 */
}
```

```
if (true or false)
{
    /* true 시 실행되는 영역 */
}
else
{
    /* false 시 실행되는 영역 */
}
```

```
class IEUsage
{
    public static void main(String[] args)
    {
        int num=10;
        if(num>0)
            System.out.println("num은 0보다 크다.");
        if((num%2)==0)
            System.out.println("num은 짝수");
        else
            System.out.println("num은 홀수");
    }
}
```

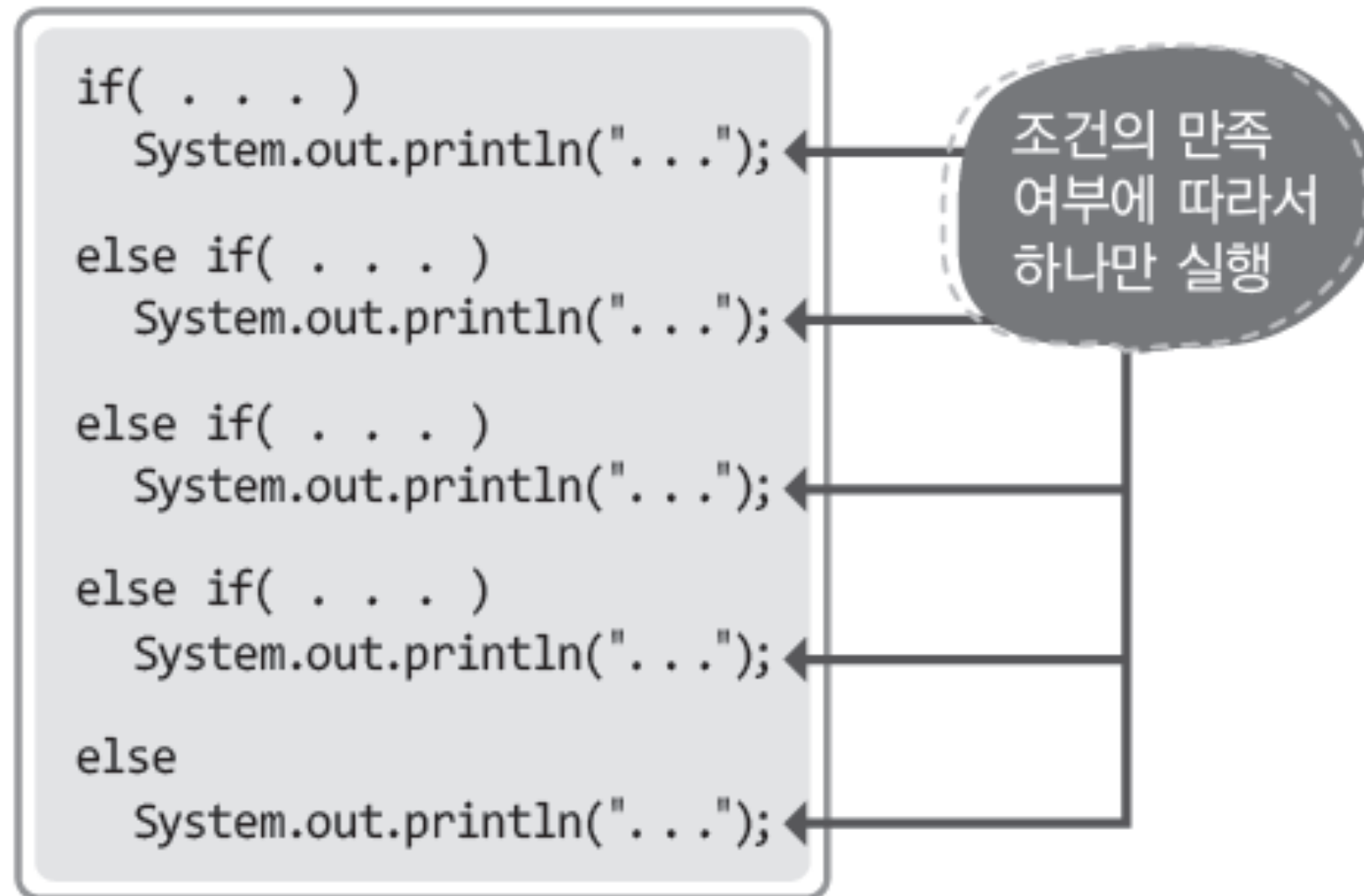
실행결과

```
num은 0보다 크다.
num은 짝수
```

중괄호는 하나의 문장일 때 생략 가능!  
if~else는 하나의 문장이다!

# IF~ELSE 중첩의 일반화

.....



중간에 else if가 추가되는만큼 if~else가 중첩된 형태이다!

# IF~ELSE와 유사한 성격의 조건 연산자

---

true or false? 숫자 1 : 숫자 2

```
class CondOp
{
    public static void main(String[] args)
    {
        int num1=50, num2=100;
        int big, diff;

        big = (num1>num2) ? num1 : num2;
        System.out.println(big);

        diff = (num1>num2)? num1-num2 : num2-num1;
        System.out.println(diff);
    }
}
```

실행결과

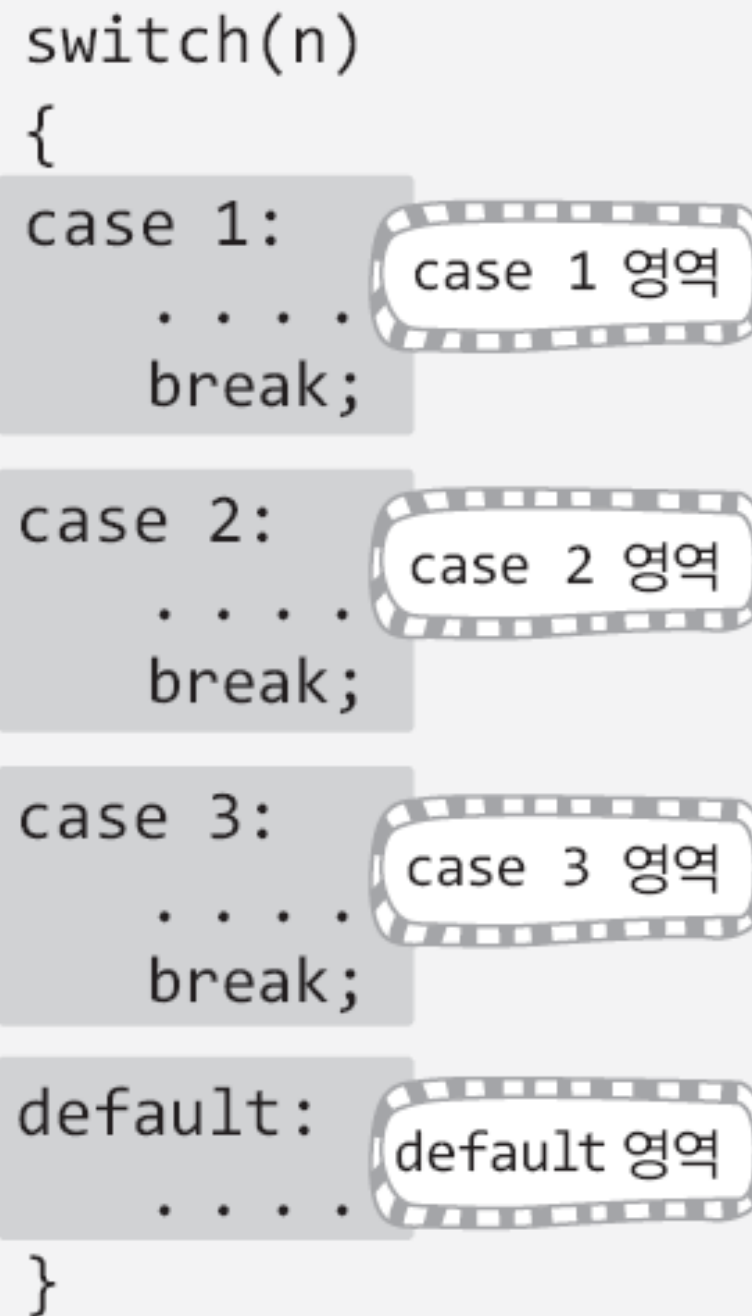
100

50

# SWITCH문 + BREAK문: SWITCH문의 일반적 사용 모델

.....

```
switch(n)
{
case 1:
    . . . . .
    break;
case 2:
    . . . . .
    break;
case 3:
    . . . . .
    break;
default:
    . . . . .
}
```



The diagram illustrates the general usage model of a switch statement. It shows a switch statement with four cases: case 1, case 2, case 3, and a default case. Each case is represented by a gray rectangular block containing the case label, a series of dots indicating code, and a break statement. To the right of each case block is a dashed rectangular box containing the text 'case 1 영역', 'case 2 영역', 'case 3 영역', and 'default 영역' respectively, indicating the execution area for each case.

# SWITCH문 + BREAK문의 예

---

```
class SwitchBreak
{
    public static void main(String[] args)
    {
        int n=3;

        switch(n)
        {
            case 1 :
                System.out.println("Simple Java");
                break;
            case 2 :
                System.out.println("Funny Java");
                break;
            case 3 :
                System.out.println("Fantastic Java");
                break;
            default :
                System.out.println("The best programming language");
        }

        System.out.println("Do you like coffee?");
    }
}
```

변수 n이 2일 때

Funny Java

Do you like coffee?

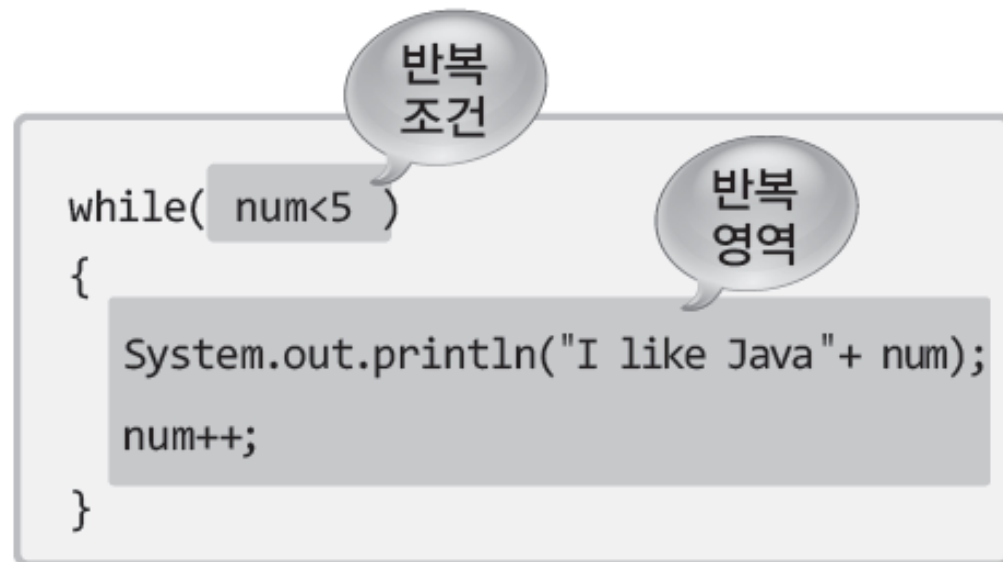
변수 n이 3일 때

Fantastic Java

Do you like coffee?

# WHILE 반복문

---



```
while( num<5 )
{
    System.out.println("I like Java"+ num);
    num++;
}
```

The diagram shows a while loop structure. A callout bubble labeled '반복 조건' (loop condition) points to the condition 'num<5'. Another callout bubble labeled '반복 영역' (loop body) points to the block of code inside the curly braces.

while문은 한번도 실행되지 않을 수 있다!

```
class WhileBasic
{
    public static void main(String[] args)
    {
        int num=0;

        while(num<5)
        {
            System.out.println("I like Java " + num);
            num++;
        }
    }
}
```

실행결과

```
I like Java 0
I like Java 1
I like Java 2
I like Java 3
I like Java 4
```



# FOR 반복문과 WHILE 반복문의 비교

---

```
int num=0; 1.
while( num<5 ) 2.
{
    System.out.println("...");
    num++; 3.
}
```

```
for( 1. int num=0 ; 2. num<5 ; 3. num++ )
{
    System.out.println("...");
}
```

1. → 반복의 횟수를 세기 위한 변수

2. → 반복의 조건

3. → 반복의 조건을 무너뜨리기 위한 연산

# FOR 반복문의 실행흐름

첫 번째 루프의 흐름

1 → 2 → 3 → 4 [i=1]

두 번째 루프의 흐름

2 → 3 → 4 [i=2]

세 번째 루프의 흐름

2 → 3 → 4 [i=3]

네 번째 루프의 흐름

2 [i=3] 따라서 탈출!

```
for( 1. int i=0 ; 2. i<3 ; 4. i++ )  
{  
  3. System.out.println("...");  
}
```

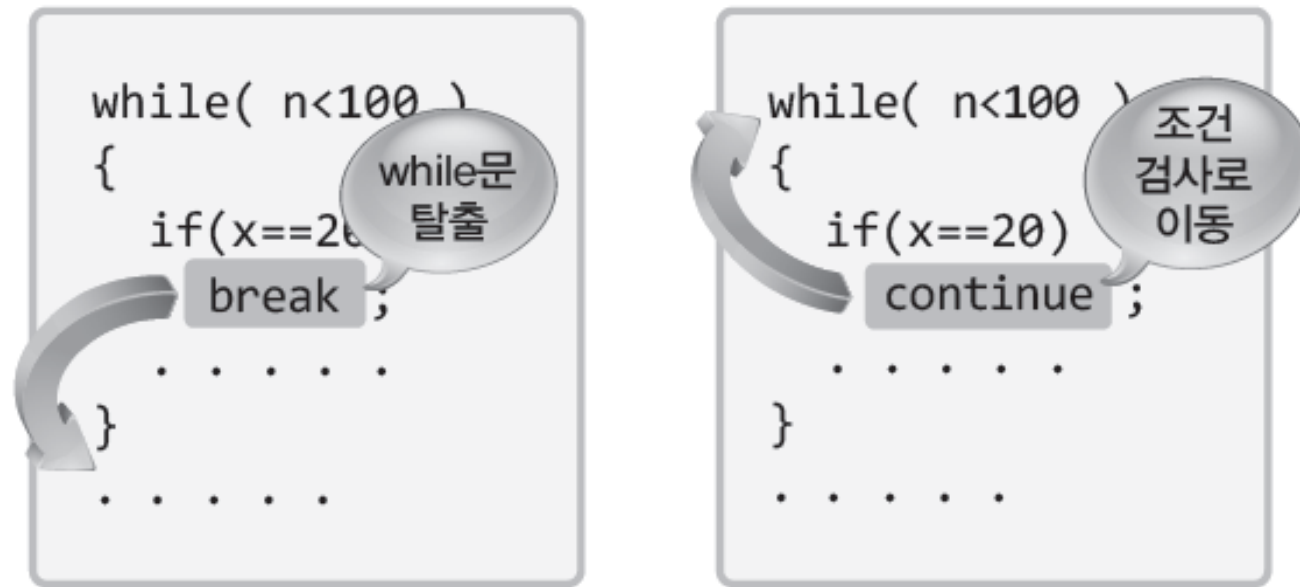
```
class ForBasic  
{  
    public static void main(String[] args)  
    {  
        for(int i=0; i<3; i++)  
            System.out.println("I love Java " + i);  
    }  
}
```

실행결과

```
I love Java 0  
I love Java 1  
I love Java 2
```

# CONTINUE & BREAK문

.....



```
while(num<100)
{
    if(num%5==0 && num%7==0)
    {
        search=true;
        break;
    }
    num++;
}
```

break문의 예

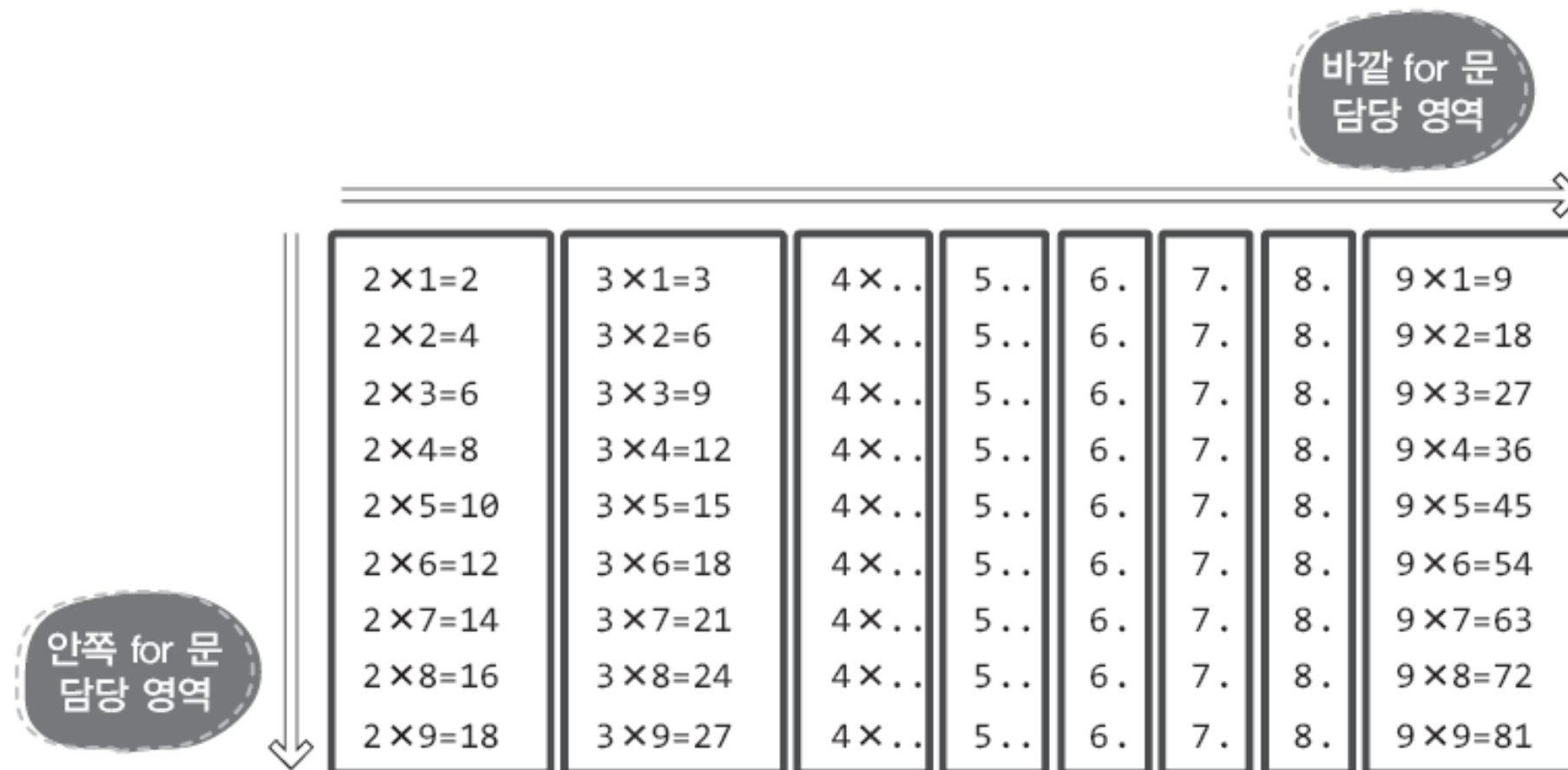
continue문의 예

```
while(num++<100)
{
    if(num%5!=0 || num%7!=0)
        continue;

    count++;
    System.out.println(num);
}
```

# 가장 많이 등장하는 FOR문의 중첩

.....



# 가장 많이 등장하는 FOR문의 중첩

---

```
class ByTimes
{
    public static void main(String[] args)
    {
        for(int i=2; i<10; i++)
        {
            for(int j=1; j<10; j++)
                System.out.println(i + " x " + j + " = " + i*j);
        }
    }
}
```

2 x 1 = 2

2 x 2 = 4

2 x 3 = 6

/\* ~중간생략~ \*/

9 x 7 = 63

9 x 8 = 72

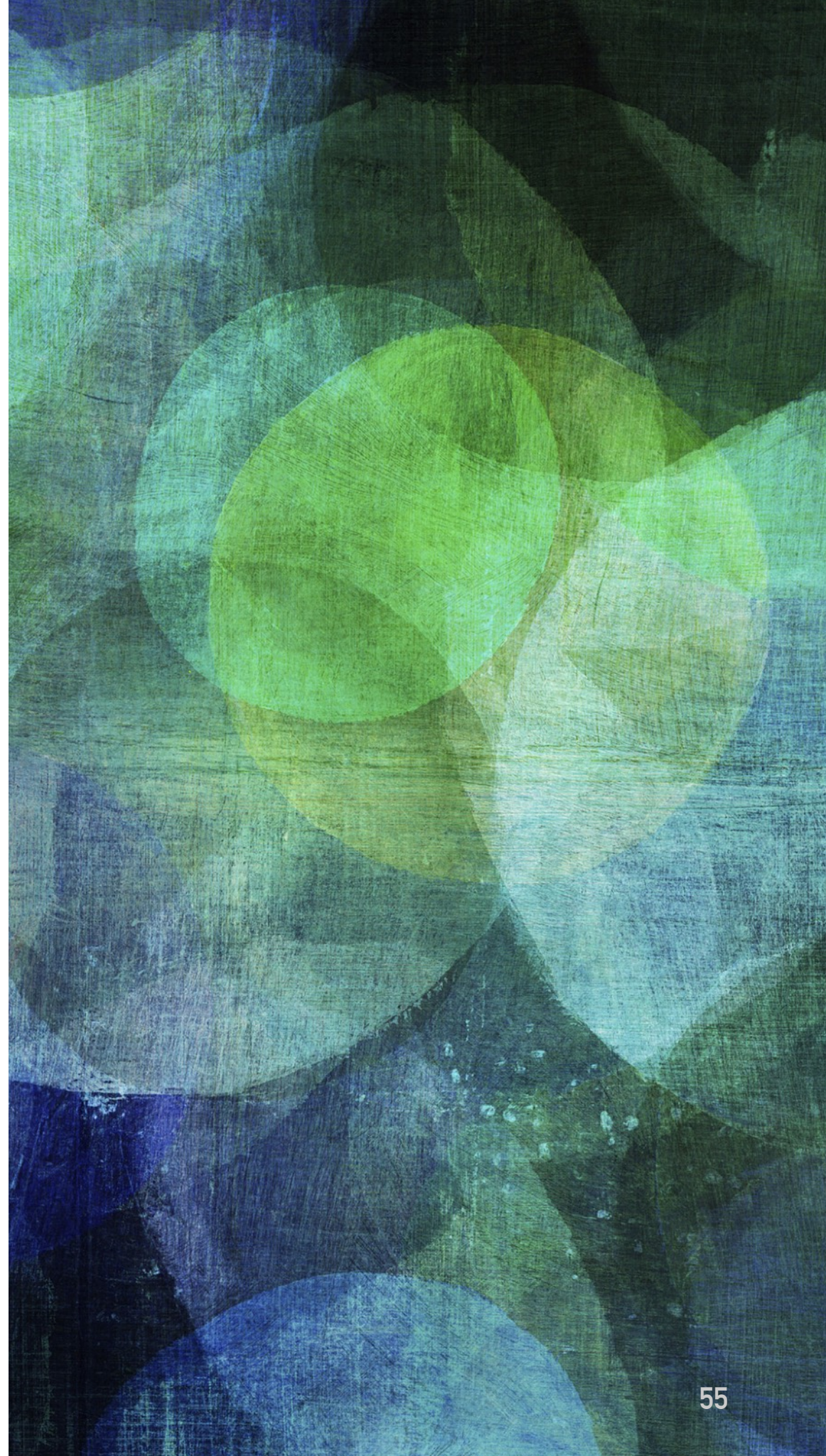
9 x 9 = 81

실습



# 오류, 디버깅

---





# 오류의 종류

---

- ▶ 변수 n2에는  $n^2$ , 변수 n3에는  $n^3$ , 변수 m에는  $n/d$ 을 대입하는 예제

```
01 public class Debug1Demo {  
02     public static void main(String[] args) {  
03         int d, m, n, n2, n3;  
04         d = 0;  
05         n = 2;  
06         n2 = n * n;  
07         n3 = n2 * n2;  
08         m = n / d;  
09     }  
10 }
```

실행문을 세미콜론으로 마쳐야 한다. 문법 오류이다.

n3는  $n^3$ 을 의미하는데,  $n^4$ 을 대입하므로 내용 오류이다.

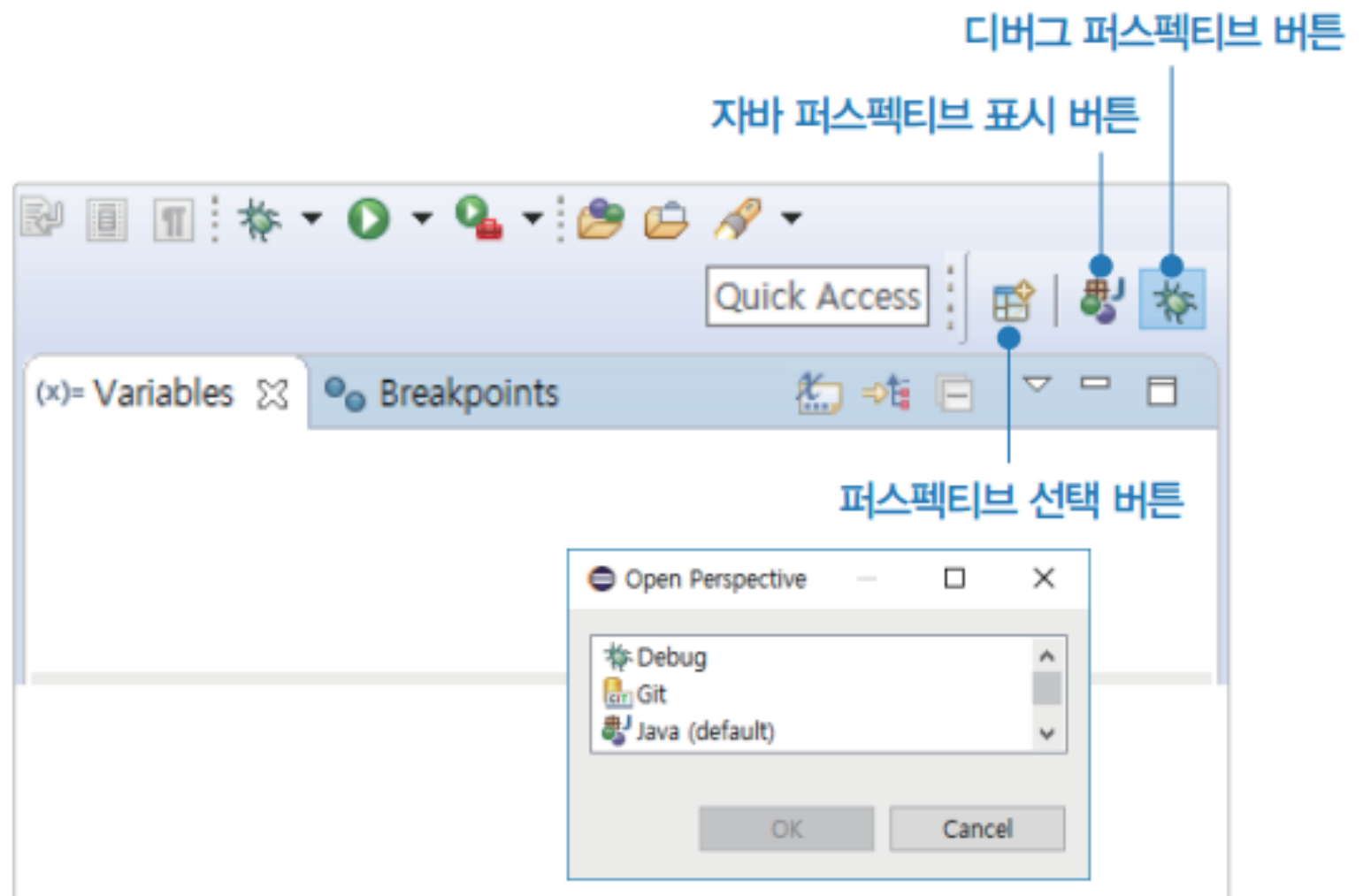
0으로 나눌 수 없으므로 실행 오류가 발생한다.



# 디버깅 과정(1)

---

- ▶ 이클립스는 에디터 뷰에 문법 오류는 알려주지만 논리 오류는 알려주지 않음
- ▶ 논리 오류는 프로그램 실행 도중에 변수 상태를 추적하는 것이 가장 기본적인 오류 점검 방식
- ▶ 디버그 퍼스펙티브로 이동



## 디버깅 과정(2)

---

### ▶ 디버깅 명령어와 이클립스 단축키

명령어	단축키	설명
Step Into	<span>F5</span>	한 행씩 실행하며, 메서드를 만나면 내부로 진입한다.
Step Over	<span>F6</span>	한 행씩 실행하며, 메서드를 만나도 내부로 진입하지 않는다.
Run to Line	<span>Ctrl</span> + <span>R</span>	다음 중단점까지 프로그램을 실행한다.
Resume	<span>F8</span>	중단된 프로그램을 다시 실행한다.
Terminate	<span>Ctrl</span> + <span>F2</span>	프로그램을 종료한다.

# 디버깅 과정(3)

.....

[예제 5-15] 간단한 자바 프로그램

sec04/Debug2Demo.java

```
01 package sec05;
02
03 public class Debug2Demo {
04     public static void main(String[] args) {
05         int year = 2018;
06         ++year;
07         System.out.println(year++ + "년 새해 복 많이 받으세요!");
08     }
09 }
```

2019년 새해 복 많이 받으세요!

# 디버깅 과정(4)

## ▶ 디버그 퍼스펙티브와 중단점 설정

디버그 퍼스펙티브  
상태임을 나타낸다.

정상 모드 실행 도구

디버그 모드 실행 도구

클릭해 디버그 퍼스펙티브로 전환

workspace - Debug - chap05/src/sec05/Debug2Demo.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Debug

C:\Program Files\Java\jdk1.8.0\_121\bin\javaw.exe (2017. 5. 30. 오후 4)

<terminated> Debug2Demo [Java Application]

<terminated> sec05.Debug2Demo at localhost:52384

<terminated, exit value: 1> C:\Program Files\Java\jdk1.8.0\_121\bin\java

(x)= Variables

Breakpoints

Name	Value
args	String[0] (id=16)
year	2018

Outline

sec05

Debug2Demo

main(String[]) : void

```
1 package sec05;
2
3 public class Debug2Demo {
4     public static void main(String[] args) {
5         int year = 2018;
6         ++year;
7         System.out.println(year++ + "년 새해 복 많이 받으세요!");
8     }
9 }
```

중단점

가터

Writable Smart Insert 1 : 1

# 디버깅 과정(5)

## ▶ 디버그 퍼스펙티브와 각종 실행 버튼

Resume Terminate

Step Into Step Over

workspace - Debug - chap05/src/sec05/Debug2Demo.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Debug

sec05.Debug2Demo at localhost:52300

Thread [main] (Suspended)

Debug2Demo.main(String[]) line: 6

C:\Program Files\Java\jdk1.8.0\_121\bin\javaw.exe (2017. 5. 30. 오후 4)

Debug2Demo [Java Application]

(x) Variables Breakpoints

Name	Value
args	String[0] (id=16)
year	2018

year 변수의 현재 값은 2018이다.

Outline

sec05

Debug2Demo

main(String[]) : void

Debug2Demo.java

```
1 package sec05;
2
3 public class Debug2Demo {
4     public static void main(String[] args) {
5         int year = 2018;
6         ++year;
7         System.out.println(year++ + "년 새해 복 많이 받으세요!");
8     }
9 }
10
```

중단점에서 실행을 멈춘다.

프로그램이 실행 중임을 나타낸다.

Console Tasks

Debug2Demo [Java Application] C:\Program Files\Java\jdk1.8.0\_121\bin\javaw.exe (2017. 5. 30. 오후 4:49:04)

# 디버깅 과정(6)

## ▶ 디버그 퍼스펙티브와 변수 추적

workspace - Debug - chap05/src/sec05/Debug2Demo.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Debug

- sec05.Debug2Demo at localhost:52300
  - Thread [main] (Suspended)
    - Debug2Demo.main(String[]) line: 7
- C:\Program Files\Java\jdk1.8.0\_121\bin\javaw.exe (2017. 5. 30. 오후 4)
- Debug2Demo [Java Application]

Variables

Name	Value
args	String[0] (id=16)
year	2019

year 값이 변경된다.

Debug2Demo.java

```
1 package sec05;
2
3 public class Debug2Demo {
4     public static void main(String[] args) {
5         int year = 2018;
6         ++year;
7         System.out.println(year++ + "년 새해 복 많이 받으세요!");
8     }
9 }
10
```

[F5] 혹은 [F6]을 누르면 한 행씩 실행문을 수행하고 멈춘다.

Outline

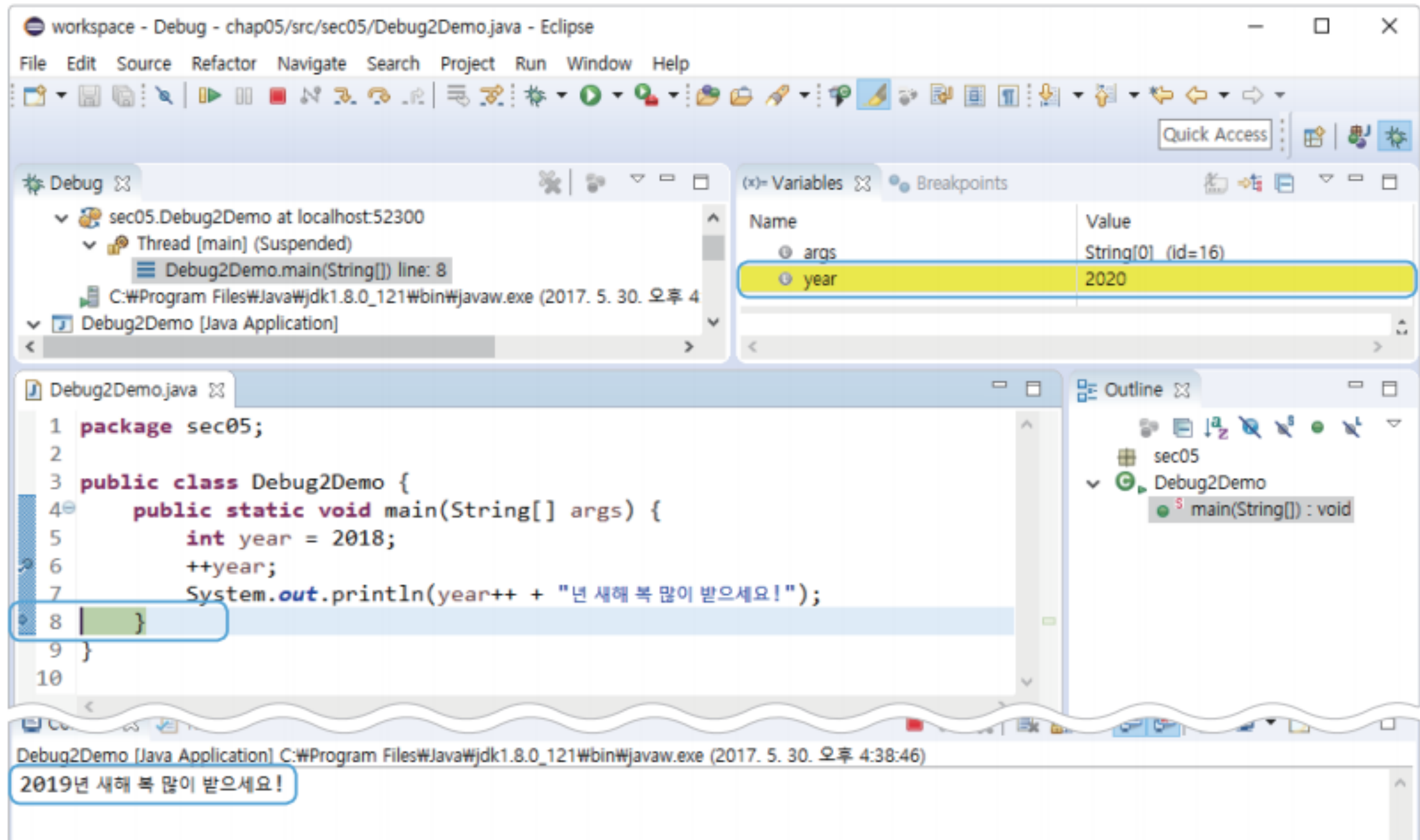
- sec05
  - Debug2Demo
    - main(String[]) : void

Writable Smart Insert 7 : 1



# 디버깅 과정(7)

## ▶ 디버그 퍼스펙티브와 실행 결과



# 실습 1

---

- ▶ 다음 소스 코드를 작성하고 발견된 문제를 수정하시오.

```
import java.util.Scanner;

public class prac01 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int maxDay = 30;
        boolean validInput = false;
        System.out.println("년과 월을 입력하면 그 달의 말일을 알려드립니다.(ex: 2003 2)");
        System.out.print("입력 > ");
        int year = scan.nextInt();
        int month = scan.nextInt();

        switch(month) {
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
            case 12:
                maxDay = 31;
                validInput = true;
        }
    }
}
```



# 실습 1

---

```
case 4:
case 6:
case 9:
case 11:
    maxDay = 30;
    validInput = true;
case 2:
    if((year%4 == 0) && (year%100 != 0) || (year%400 == 0))
        maxDay = 29;
    else
        maxDay = 28;
    validInput = true;
default:
    validInput = false;
}

if(validInput)
    System.out.println(year + "년 " + month + "월의 말일은 " + maxDay + "입니다.");
else
    System.out.println("입력이 잘못 되었습니다.");
}
}
```

# 실습 1

---

## ▶ 정상적 출력 결과

년과 월을 입력하면 그 달의 말일을 알려드립니다.(ex: 2003 2)  
입력 > 2013 2  
2013년 2월의 말일은 28입니다.

년과 월을 입력하면 그 달의 말일을 알려드립니다.(ex: 2003 2)  
입력 > 2013 1  
2013년 1월의 말일은 31입니다.

## 실습 2

---

- ▶ 다음의 모양을 화면에 출력하시오.
- ▶ `System.out.println()` - 줄바꿈(Enter)이 있는 출력
- ▶ `System.out.print()` - 줄바꿈 없이 출력
- ▶ for문만 사용한다. (if문은 사용하지 않음)

4.

1.	2.	3.	
*****	*****	*	1
*****	*****	**	1 2
*****	*****	***	1 2 3
*****	*****	****	1 2 3 4
*****	*****	*****	1 2 3
*****	*****	*****	1 2
*****	*****	*****	1

# 실습 2

---

## ▶ 이중 반복문 기본형

```
public class prac02 {  
    public static void main(String[] args) {  
        for(int row = 0; row < 5; row++) {  
            for(int colum = 0; colum < 5; colum++)  
                System.out.print("*");  
            System.out.println();  
        }  
    }  
}
```

### [출력결과]

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

# FOR문 중첩

---

- ▶ Pascal's triangle

- ▶ 입력 : 변수 rows를 변경하면서 줄 수를 변경

- ▶ If 문 사용하여야 함

- ▶ 출력

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

