

어서와 Java는 처음이지!

제2장 자바 프로그래밍 기초

- 자바 프로그램 구조
- 변수
 - ◉ 데이터 타입
 - ◉ 상수
- 연산자
 - ◉ 산술, 논리 연산자 외
 - ◉ 연산자 우선순위

이제 본격적으로
프로그래밍이 시작되나요?

네, 이번 장에서는 변수나
함수와 같은 프로그래밍 기초
개념들을 학습합니다. 이것들은
프로그래밍의 토대가 됩니다.





자바 프로그램의 구조

- 하나의 클래스 안에는 여러 개의 함수가 포함될 수 있다.
- 하나의 함수 안에는 여러 개의 문장이 포함될 수 있다.



가방= 클래스

필통= 함수

펜슬= 문장



클래스

- 클래스(class)는 자바와 같은 객체 지향 언어의 기본적인 빌딩 블록
- 필요한 클래스를 하나씩 만들어감으로써 전체 프로그램이 완성된다.



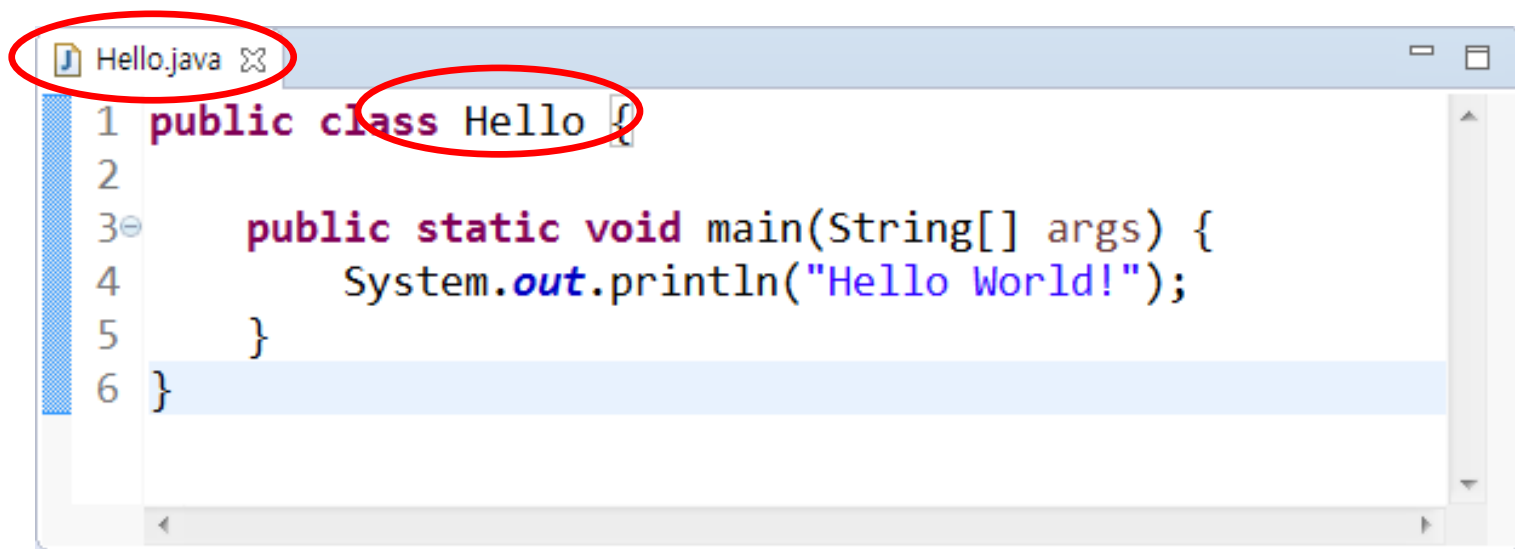
클래스는 자바 프로그램을
이루는 기본적인
빌딩블록 입니다.





소스 파일과 클래스 이름

- 소스 안에 `public` 클래스가 있다면 반드시 소스 파일의 이름은 `public` 클래스의 이름과 일치하여야 한다.
- 하나의 소스 파일 안에 `public` 클래스가 2개 이상 있으면 컴파일 오류가 발생한다.

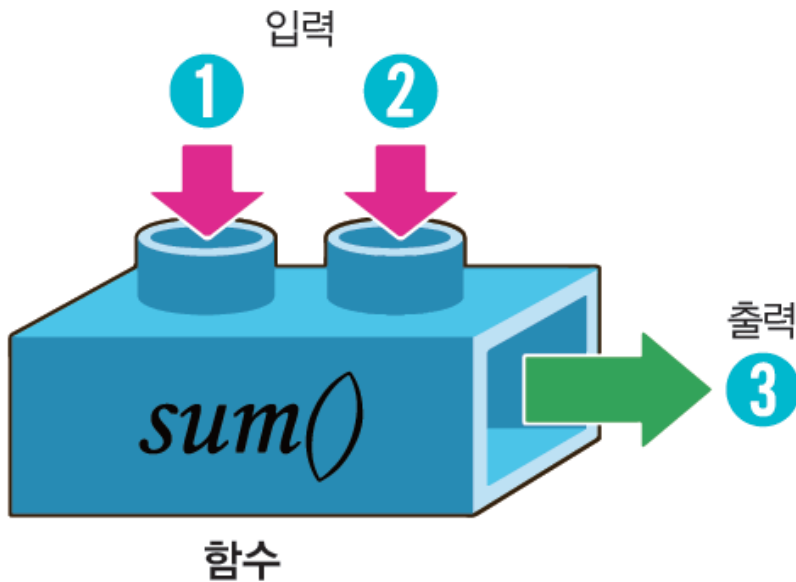


```
1 public class Hello {
2
3     public static void main(String[] args) {
4         System.out.println("Hello World!");
5     }
6 }
```



함수 = 메소드

- 메소드(method)는 특정한 작업을 수행하는 코드의 묶음
- 메소드는 외부로부터 입력을 받아서 특정한 작업을 수행하고 작업의 결과를 반환하는 블랙 박스



메소드는 입력을 받아서 어떤 처리를 하고 처리의 결과를 돌려주는 코드들의 모임입니다. 클래스 안에 정의됩니다.





main() 메소드

- main() 메소드를 가지고 있는 클래스가 반드시 하나는 있어야 한다.
- main() 메소드에서 자바 프로그램의 실행이 시작된다.



JVM

나는 main() 메소드를
제일 먼저 실행합니다.



문장

- 사용자가 컴퓨터에게 작업을 지시하는 단위

Hello.java

```
01 public class Hello {  
02  
03     public static void main(String[] args) {  
04         System.out.println("Hello World!");  
05     }  
06 }
```

이것이 작업의 내용을
기술하는 문장이다.



주석

- `/* TEXT */`

- ⊙ 주석의 시작과 끝을 `/*`와 `*/`로 표시

- `// TEXT`

- ⊙ `//`에서 줄의 끝까지가 주석이다. 한 줄짜리 주석만 가능하다.

- `/** DOCUMENTATION */`

- ⊙ `/**`에서 `*/`까지가 주석이 된



Your comment here...



주석을 붙인 소스

Hello.java

```
01  /**
02   * 첫 번째 예제 프로그램
03   * @version 1.0 2015-08-01
04   * @author 홍길동
05   */
06  public class Hello {
07
08      public static void main(String[] args) {
09          System.out.println("Hello World!"); // "Hello World!"를 출력한다.
10      }
11  }
```

주석을 붙인다.



LAB1-1: 순차적인 프로그램 작성하기

- 다음과 같은 출력을 생성하는 프로그램을 작성

```
Hello World!  
I'm a new Java programmer.
```



SOLUTION

소스코드

Hello.java

```
01 public class Hello1 {  
02     public static void main(String args[]) {  
03         System.out.println("Hello World!");  
04         System.out.println("I'm a new Java programmer.");  
05     }  
06 }
```

문장들은 순차적으로 실행된다.



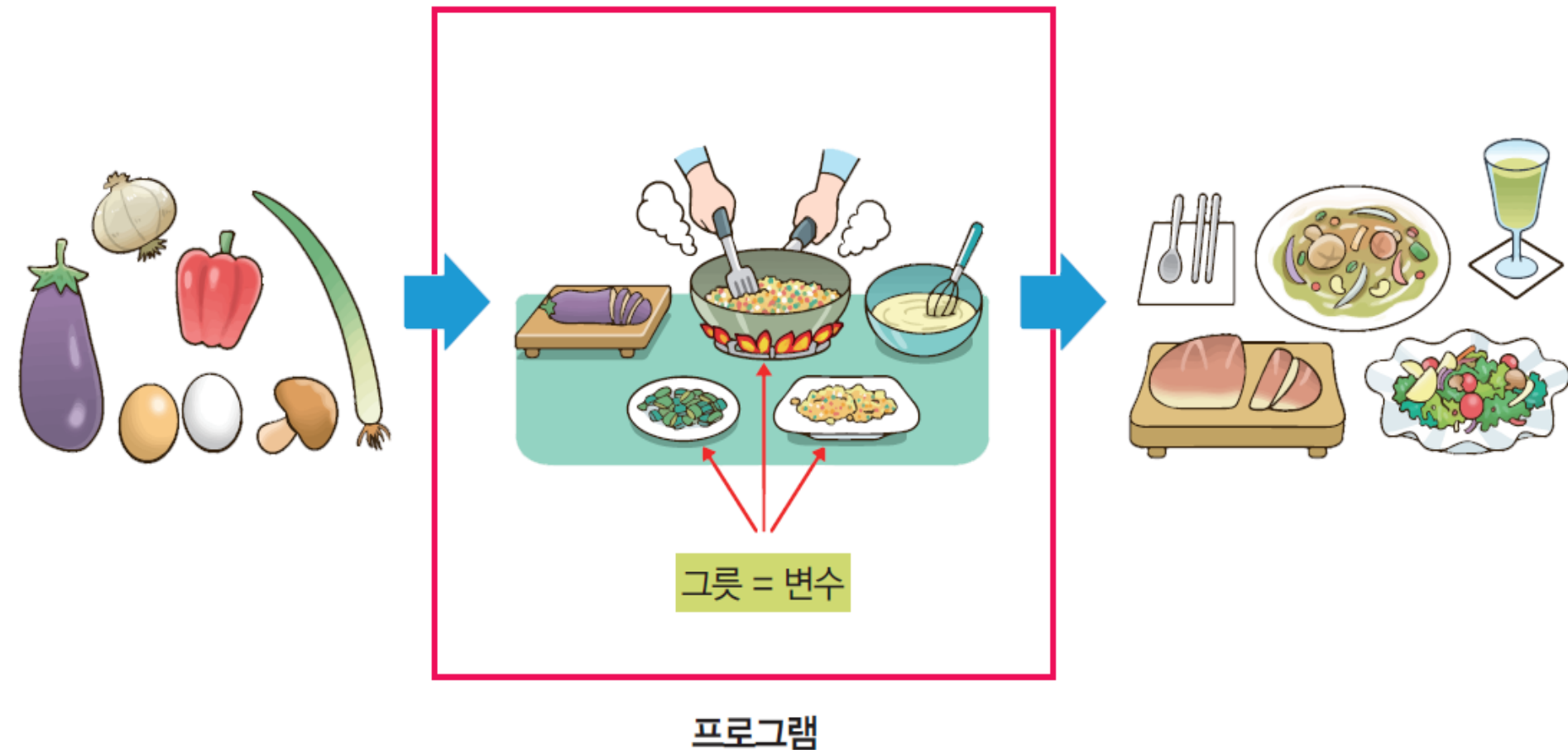
변수

- 변수(variable)는 데이터를 담아두는 상자





변수의 필요성





변수 선언

전체적인 구조



형식

`int`

`value;`

자료형

변수 이름

value

전체적인 구조



형식

`int`

`value = 9`

초기값

value

9



자료형

○ 변수에 저장되는 데이터의 타입

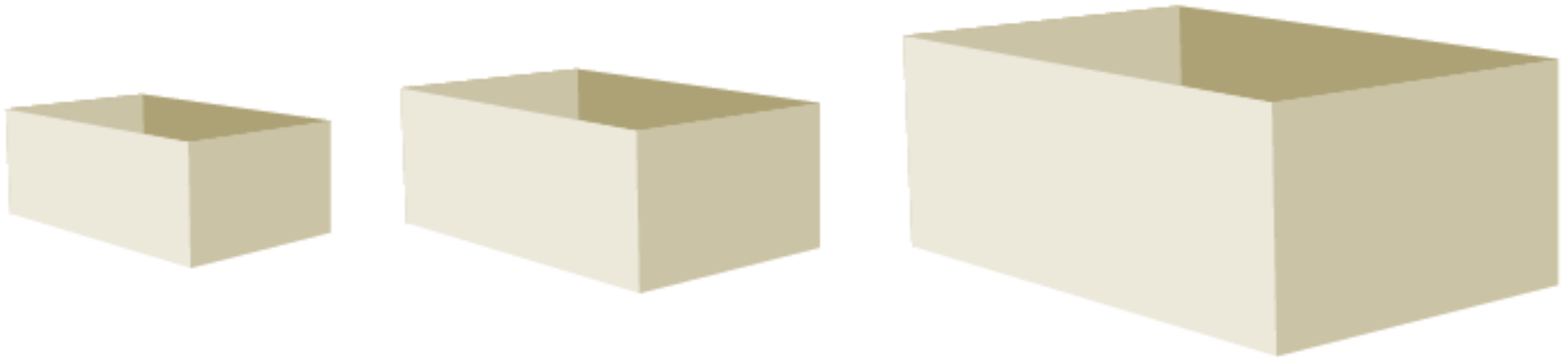


그림 2-7 • 자료형에는 여러 가지 종류가 있다.



자료형의 종류



기초형
실제 값이 저장

정수형: byte, short, int, long

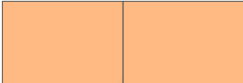
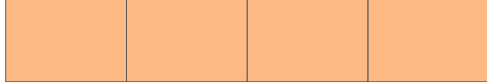
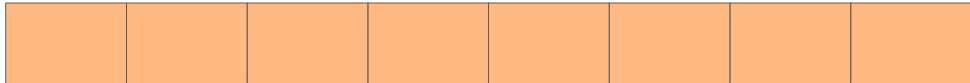
실수형: float, double

논리형: boolean

문자형: char



기초 자료형

byte		-128부터 127
short		-32768부터 32767
int		약 -21억부터 21억까지
long		
float		
double		
boolean		true, false
char		유니코드



LAB1-2: 원 면적 계산, 작성하기

- 반지름이 5.0인 원의 면적을 계산하는 프로그램을 작성
 - ⊙ 곱하기는 *로 표시
- 다음과 같은 출력을 생성하는 프로그램을 작성



원의 면적은 78.5398



SOLUTION

AreaTest.java

```
01 public class AreaTest {  
02     public static void main(String args[]) {  
03         double radius, area;  
04  
05         radius = 5.0;  
06         area = 3.141592 * radius * radius;  
07         System.out.println("원의 면적은 " + area);  
08     }  
09 }
```



논리형

직접 입력
하여 확인



BooleanTest.java

```
01 public class BooleanTest {  
02     public static void main(String args[]) {  
03         boolean b;  
04  
05         b = true;  
06         System.out.println("b : " + b);  
07         b = ( 1 > 2 );  
08         System.out.println("b : " + b);  
09  
10     }  
11 }
```

boolean형은 true
또는 false만을 가질
수 있습니다.



실행결과



```
b : true  
b : false
```



변수 초기화 오류

직접 입력
하여 확인

VarInitTest.java

```
01 public class VarInitTest {  
02     public static void main(String args[]) {  
03         int index;  
04  
05         index = index + 1;  
06         System.out.println("index : " + index);  
07     }  
08 }
```

초기화 하지 않고
변수를 사용하였다.

실행결과

컴파일 오류: The local variable index may not have been initialized



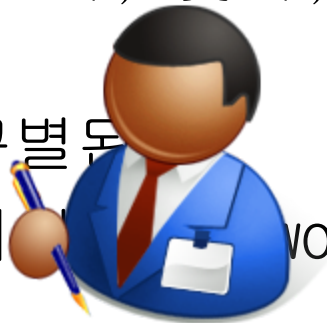
변수 이름 짓기

- 식별자는 유니코드 문자와 숫자의 조합으로 만들어진다.
- 식별자의 첫 문자는 일반적으로 유니코드 문자이어야 한다. _, \$로 시작 가능
- 두 번째 문자부터는 문자, 숫자, _, \$ 등이 가능하다.
- 대문자와 소문자는 구별된다.
- 식별자의 이름으로 키워드(keyword)를 사용해서는 안 된다.



변수의 이름

- 변수의 이름은 식별자(identifier)의 일종
- 변수 이름의 규칙
 - ⊙ 식별자는 유니코드 문자와 숫자의 조합(한글 가능!)
 - ⊙ 식별자의 첫 문자는 일반적으로 유니코드 문자
 - ⊙ 두 번째 문자부터는 문자, 숫자, _, \$ 등이 가능하다.
 - ⊙ 대문자와 소문자는 구별된다.
 - ⊙ 식별자의 이름으로 키워드(keyword)를 사용해서는 안 된다.





변수 이름의 예

적법한 변수 선언의 예를 들어보면 다음과 같다.

```
int    speed;
long   earthPopulation;
int    _count;           // _로 시작할 수 있다.
long   $value;           // $로 시작할 수 있다.
int    반복횟수;         // 유니코드를 지원하므로 한글 변수 이름도 가능
int    Henry8;           // 맨 처음이 아니라면 숫자도 넣을 수 있다.
```

잘못된 변수 선언의 예는 다음과 같다.

```
int    1stPrizeMoney;    // 첫글자가 숫자
double super;            // 키워드
int    #ofComputer;      // 첫글자가 허용되지 않는 기호
```



변수 이름 관례

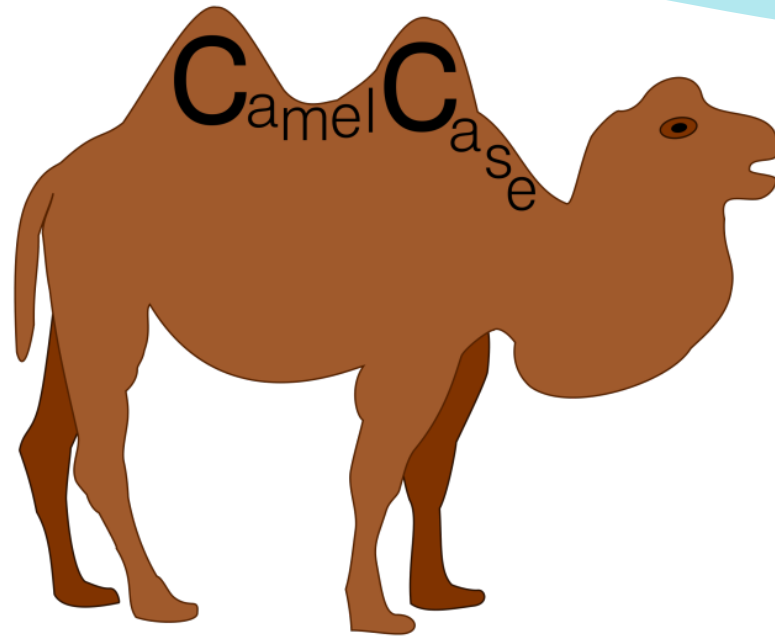
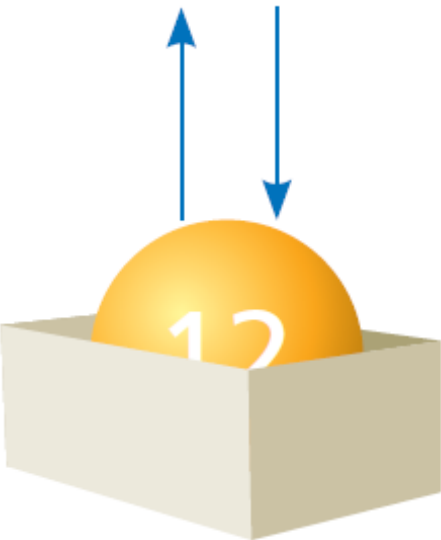


표 2-1 • 식별자의 관례

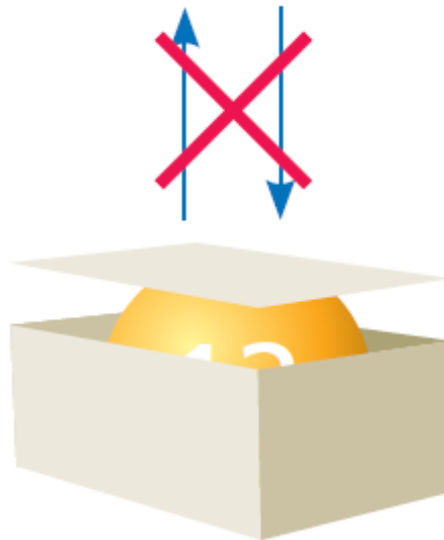
종 류	사용 방법	예
클래스명	각 단어의 첫글자는 대문자로 한다.	StaffMember, ItemProducer
변수명, 메소드명	첫번째 단어는 소문자로 시작되어 2번째 단어부터 첫글자는 대문자로 한다.	width, payRate, acctNumber, getMonthDays(), fillRect(),
상수	상수는 모든 글자를 대문자로 한다.	MAX_NUMBER



상수



변수



상수

변수는 실행도중에
값을 변경할수 있으나
상수는 한번 값이
정해지면 변경이
불가능 합니다.





상수를 만드는 방법

- 일반 변수 선언에 `final`을 붙이면 된다.

```
final double PI = 3.141592
```



LAB1-3: 마일을 킬로미터로 변환

- 60마일을 킬로미터로 변환하는 프로그램 작성
 - ◎ KM-PER-MILE의 변환수치는 1.609344
- 다음과 같은 출력을 생성하는 프로그램을 작성



Solution

직접 입력
하여 확인



Constant.java

```
01 public class Constant {  
02     public static void main(String[] args) {  
03         final double KM_PER_MILE = 1.609344;  
04         double km;  
05         double mile = 60.0;  
06         km = KM_PER_MILE * mile;  
07  
08         System.out.println("60마일은 " + km + "킬로미터입니다.");  
09     }  
10 }
```

final을 붙여서 부동소수점형
기호상수를 정의하고 있다.

실행결과

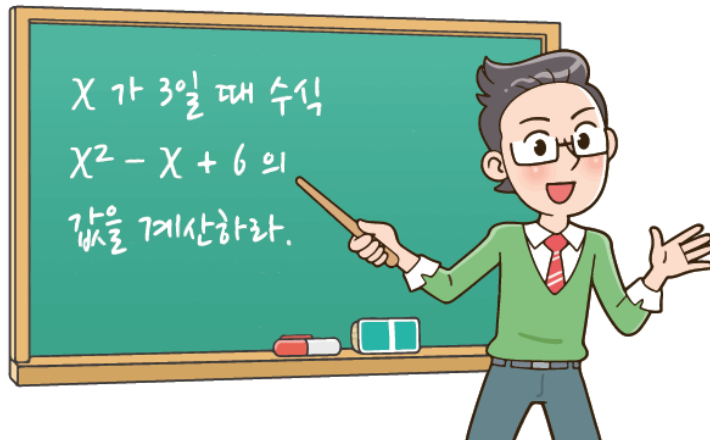


60마일은 96.56064킬로미터입니다.



수식

- 수식이란 상수나 변수, 함수와 같은 피연산자들과 연산자의 조합



```
int x, y;
```

```
x = 3;
```

```
y = x * x - x + 6;
```

```
system.out.println(y);
```

그림 2-2 • 수식의 예



대입 연산자

- 대입 연산자(=)는 왼쪽 변수에 오른쪽 수식의 값을 계산하여 저장
- 대입 연산자 == 할당 연산자 == 배정 연산자라고도 한다.

$x = 100;$ // 상수 100을 변수 x에 대입한다.

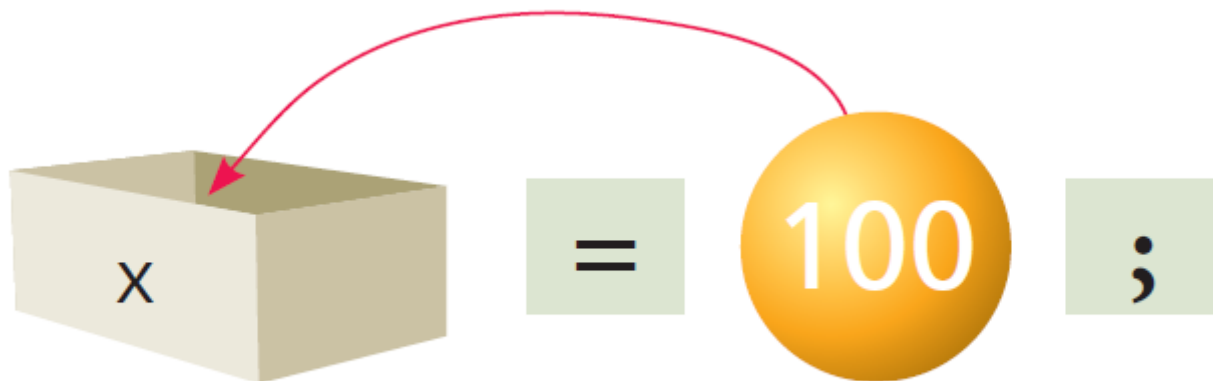


그림 2-3 • 대입 연산자는 변수에 값을 저장하는 연산자이다.



산술 연산자

연산자	기호	의미	예
덧셈	+	x와 y를 더한다	$x+y$
뺄셈	-	x에서 y를 뺀다.	$x-y$
곱셈	*	x와 y를 곱한다.	$x*y$
나눗셈	/	x를 y로 나눈다.	x/y
나머지	%	x를 y로 나눌 때의 나머지값	$x\%y$

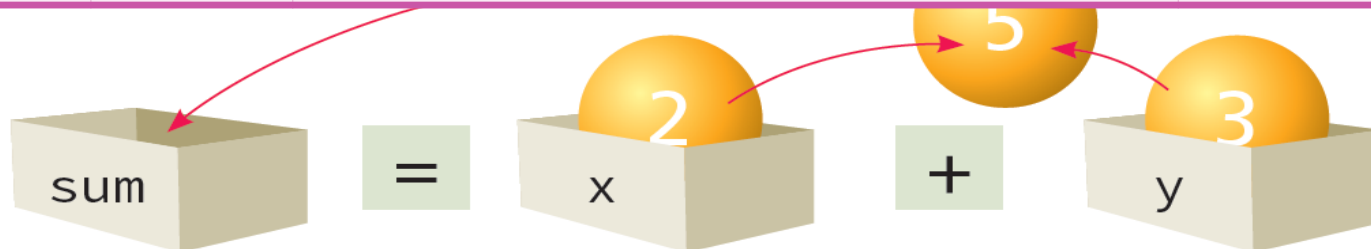


그림 2-4 • 산술 연산의 과정: 먼저 x와 y에서 값을 가져와서 덧셈연산이 수행되고 그 결과값이 sum에 저장된다.



중감 연산자



연산자	의미
$++x$	x 값을 먼저 증가한 후에 다른 연산에 사용한다. 이 수식의 값은 증가된 x 값이다.
$x++$	x 값을 먼저 사용한 후에, 증가한다. 이 수식의 값은 증가되지 않은 원래의 x 값이다.
$--x$	x 값을 먼저 감소한 후에 다른 연산에 사용한다. 이 수식의 값은 감소된 x 값이다.
$x--$	x 값을 먼저 사용한 후에, 감소한다. 이 수식의 값은 감소되지 않은 원래의 x 값이다.



예제: 증감 연산자

```
int x = 1;  
int y = 1;  
y = x++;
```

```
System.out.println(x);  
System.out.println(y);
```

○ 결과는?



관계 연산자

표 2-3 • 관계 연산자

연산자 기호	의미	사용예
==	x와 y가 같은가?	$x == y$
!=	x와 y가 다른가?	$x != y$
>	x가 y보다 큰가?	$x > y$
<	x가 y보다 작은가?	$x < y$
>=	x가 y보다 크거나 같은가?	$x >= y$
<=	x가 y보다 작거나 같은가?	$x <= y$



논리 연산자



연산자 기호	사용예	의미
&&	x && y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
	x y	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓
!	!x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참



예제: 논리 연산자

○ 결과는?

LogicalOperator.java

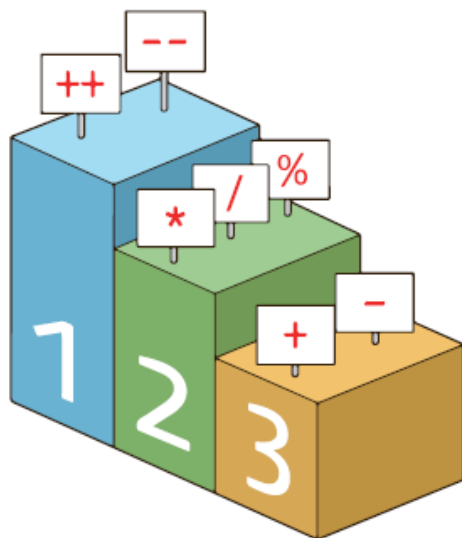
```
01 public class LogicalOperator {  
02  
03     public static void main(String[] args){  
04         int x = 3;  
05         int y = 4;  
06         System.out.println((x == 3) && (y == 7)) ;  
07         System.out.println((x == 3 || y == 4)) ;  
08     }  
09 }
```

AND 연산

OR 연산



연산자의 우선순위



우선 순위는 어떤 연산자를
먼저 계산할지를 나타내는
순위입니다.



그림 2-5 • 산술 연산자의 우선 순위

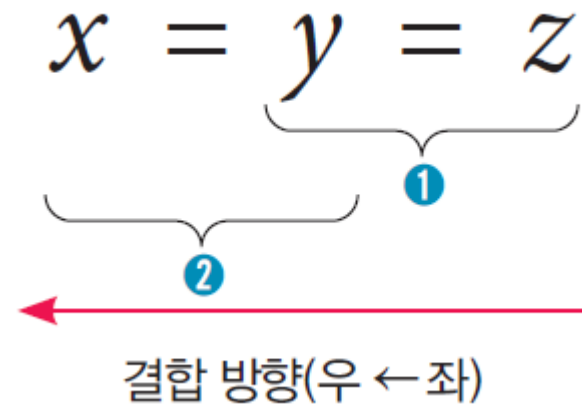
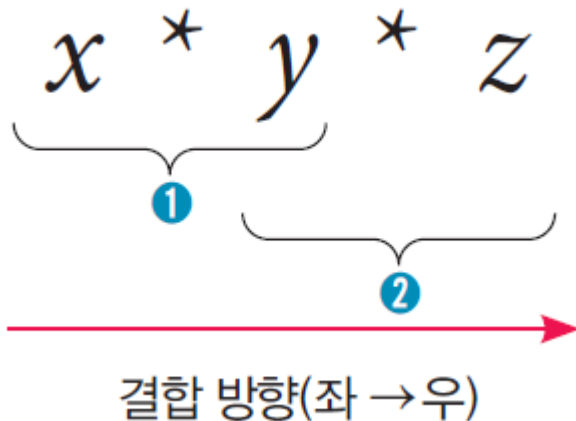
$$x + y * z$$

The diagram shows the expression $x + y * z$ with brackets indicating the order of operations. A bracket labeled '1' is under the $y * z$ part, and a larger bracket labeled '2' is under the entire expression $x + y * z$.



결합 규칙

- 동일한 우선 순위의 연산이 있는 경우에 무엇을 먼저 수행하느냐에 대한 규칙

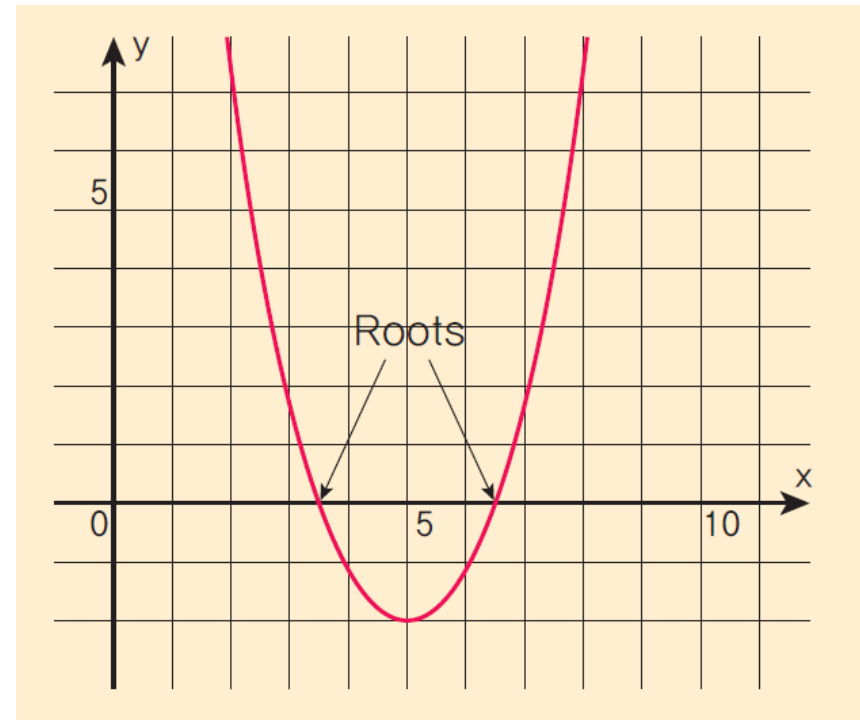




LAB1-4: 2차 방정식의 근을 계산

- 2차 방정식 $x^2 + b \cdot x + c$ 에서 b 를 -3.0 으로 c 를 2.0 으로 한다.
- 근을 계산하는 프로그램을 작성하자 (공식)

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



⊙ 루트 함수: `Math.sqrt` (값)



Solution

QuadraticEq.java

```
01 public class QuadraticEq {  
02  
03     public static void main(String[] args) {  
04         double b = -3.0;  
05         double c = 2.0;  
06  
07         double disc = b * b - 4.0 * c;  
08         double sqr = Math.sqrt(disc);  
09  
10         double r1 = (-b + sqr) / 2.0;  
11         double r2 = (-b - sqr) / 2.0;  
12  
13         System.out.println("근은 " + r1);  
14         System.out.println("근은 " + r2);  
15     }  
16 }
```

실행결과



```
근은 2.0  
근은 1.0
```



Q & A

