자바 프로그래밍 기초

514760-1
2016년 가을학기
9/8/2016
박경신

자바 프로그램 구조

```
/*
* 소스 파일 : Hello.java
*/
public class Hello {

  // main() 메소드에서 실행 시작
  public static void main(String[] args) {
     // "Hello World!" 문자열 화면 출력
     System.out.println("Hello World!");
  }
}
```

주석문
주석문
클래스
메소드

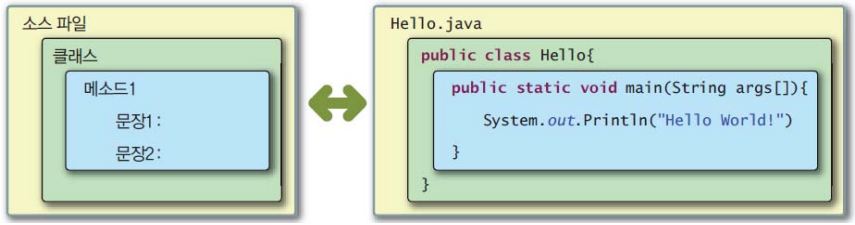
```
?
Hello World!
```

주석문 (Comments)

- 주석문
  - 프로그램에 대한 설명을 하기 위해 활용되는 코드의 일부로 컴파일 시에는 무시되고 사용되지 않음
- 세 가지 형태의 주석문
  - /* 설명 */
    - /*에서 */ 까지가 주석으로서 컴파일 시에 무시된다
    - 여러 줄에 걸쳐서 사용 가능
  - // 설명
    - //에서 줄의 끝까지 무시된다
  - /** 설명 */
    - /* 설명 */ 형태의 주석문이지만, 주로 선언문 앞에 사용되어 JDK에 포함된 Javadoc 프로그램을 이용해서 HTML문서를 만드는데 활용되는 주석문

자바 프로그램의 구조

- 클래스 (class): 객체(object)를 만드는 설계도 (템플릿)
- 자바 프로그램은 기본적으로 클래스로 구성됨



- **public** 키워드는 Hello 클래스가 다른 클래스에서도 사용 가능함을 나타냄
- 하나의 클래스 안에는 여러 개의 메소드가 포함될 수 있음
- 하나의 메소드 안에는 여러 개의 문장이 포함될 수 있음

Method

- 클래스 = 데이터 + 동작
  = 필드(변수) + 메소드(함수)
- **메소드(Method)**는 특정한 작업을 수행하는 코드의 묶음
- 메소드는 외부로부터 입력을 받아서 특정한 작업을 수행하고 작업의 결과를 반환하는 블랙 박스

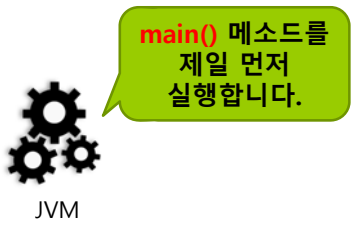
Method

```
public static void main(String[] args) {
    // "Hello World!" 문자열 화면 출력
    System.out.println("Hello World!");
}
```

- public: 누구나 이용 가능
- static : 정적 메소드
- void : 반환값 (결과값) 없음
- main : 메소드 이름
- String args[] : 외부에서 주어지는 데이터를 받는 매개변수 (입력)

main()

- 자바 프로그램은 **main()** 메소드를 가지고 있는 클래스가 반드시 하나는 있어야 함
- **main()** 메소드에서 자바 프로그램의 실행이 시작됨



Statement

- 문장 (Statement)은 사용자가 컴퓨터에게 작업을 지시하는 코드의 단위가 됨
- 문장들은 메소드 안에 들어 있거나 또는 클래스 내부에서 변수 등을 정의하는데 활용됨
- 보통 프로그램의 한 줄이 하나의 문장이 됨. 이때에는 문장의 끝은 항상 **세미콜론(;)**으로 끝남
- 때로는 { 문장 또는 문장들 }로 구성되는 블록 문장도 가능

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.

자바 프로그램 구조 – Hello2 예제

```
/*
* 소스 파일 : Hello2.java
*/
public class Hello2 {

   public static int sum(int n, int m) {
      return n + m;
   }

   // main() 메소드에서 실행 시작
   public static void main(String[] args) {
      int i = 20;
      int s;
      char a;

      s = sum(i, 10); // sum() 메소드 호출
      a = '?';
      System.out.println(a); // 문자 '?' 화면 출력
      System.out.println("Hello2"); // "Hello2" 문자열 화면 출력
      System.out.println(s); // 정수 s 값 화면 출력
   }
}
```

메소드

클래스

메소드

```
?
Hello2
30
```

Hello2 예제 설명

- 클래스 만들기
  - Hello2 이름의 클래스 선언

    ```
    public class Hello2 {
    }
    ```

  - class 키워드로 클래스 정의(4장 참고)
  - public으로 선언하면 다른 클래스에서도 접근 가능
  - 클래스 본문은 '{'으로 시작하여 '}'으로 끝남
- main() 메소드
  - public static void으로 선언되어야 함

    ```
    public static void main(String[] args) {
    }
    ```

  - 자바 프로그램은 main() 메소드부터 실행 시작
  - String[] args로 실행 인자를 전달 받음(3장 참고)
- 멤버 메소드
  - 메소드 sum() 정의

    ```
    public static int sum(int n, int m) {
    ...
    }
    ```

  - 클래스에 속한 함수, 클래스 내에서만 선언
  - 인자들의 타입과 변수 명을 ','로 분리하여 나열
  - 메소드 코드는 '{'과 '}' 사이에 작성
- 변수 선언
  - 변수 타입과 변수 이름 선언

    ```
    int i=20;
    int s;
    char a;
    ```

  - 메소드 내에서 선언된 변수는 지역 변수
    - 지역 변수는 메소드 실행이 끝나면 저장 공간 반환
- 메소드 호출
  - sum() 메소드 호출

    ```
    s = sum(i,10); // 메소드 sum() 호출
    ```

  - sum() 메소드의 호출 시 변수 i의 값과 정수 10을 전달
  - sum() 메소드의 인자인 n, m에 각각 20, 10 값 전달
  - sum() 메소드는 n과 m 값을 더한 30 리턴
  - 변수 s는 정수 30을 전달받아 저장

sum() 메소드 호출과 리턴

```
public static int sum(int n, int m) {
   return n + m; // 30 리턴
}
```

n 20
m 10

```
int i=20;

s = sum(i, 10);
```

sum() 메소드 호출

s 30

System.out.println

- 화면 출력
  - 표준 출력 스트림에 메시지 출력

```
System.out.println(a); // 문자 ? 화면 출력
System.out.println("Hello2"); // "Hello2" 문자열 화면 출력
System.out.println(s); // 정수 s 값 화면 출력
```

  - 표준 출력 스트림 System.out의 println() 메소드 호출
  - println()은 여러 종류 데이터 타입 출력 가능
  - println()은 출력 후 다음 행으로 커서 이동

Identifier

- 식별자란?
  - 클래스, 변수, 상수, 메소드 등에 붙이는 이름
- 식별자의 원칙
  - '@', '#', '!'와 같은 특수 문자, 공백 또는 탭은 식별자로 사용할 수 없으나 '_', '$'는 사용 가능
  - 유니코드 문자 사용 가능. 한글 사용 가능
  - 자바 언어의 키워드는 식별자로 사용불가
  - 식별자의 첫 번째 문자로 숫자는 사용불가
  - ' ' 또는 '$'를 식별자 첫 번째 문자로 사용할 수 있으나 일반적으로 잘 사용하지 않는다.
  - 불린 리터럴 (true, false)과 널 리터럴(null)은 식별자로 사용불가
  - 길이 제한 없음
- 대소문자 구별
  - Test와 test는 별개의 식별자

Identifier

- 사용 가능한 예

```
int name;
char student_ID;                    // '_' 사용 가능
void $func() { }                    // '$' 사용 가능
class Monster3 { }                  // 숫자 사용 가능
int whatsyournamemynameiskitae; // 길이 제한 없음
// 대소문자 구분. barChart와 barchart는 다름
int barChart; int barchart;
int 가격;                           // 한글 이름 사용 가능
```

- 잘못된 예

```
int 3Chapter;          // 숫자로 사용하였기 때문
class if { }           // if는 자바의 예약어임
char false;            // false는 사용 불가
void null() { }        // null 사용 불가
class %calc { }        // '%'는 특수문자
```

Keywords

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

식별자 이름 붙이는 관례

- 클래스(class) 이름

```
public class HelloWorld {}
class Vehicle {}
class AutoVendingMachine {}
```

  - 첫 번째 문자는 대문자로 시작
  - 여러 단어가 복합되어 있을 때는 각 단어의 첫 번째 문자만 대문자로 표시
- 변수(variable), 메소드(method) 이름

```
int iAge;                  // iAge의 i는 int의 i를 표시
boolean bIsSingle;         // bIsSingle의 처음 b는 boolean의 b를 표시
String strName;            // strName의 str은 String의 str을 표시
public int iGetAge() {}    // iGetAge의 i는 int의 i를 표시
```

  - 첫 단어 이후 각 단어의 첫 번째 문자는 대문자로 시작

식별자 이름 붙이는 관례

- 상수(constant) 이름

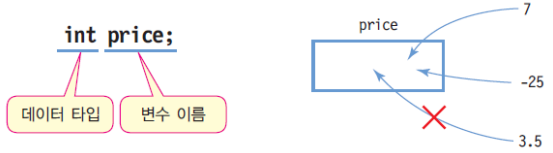
```
final static double PI = 3.141592;
```

  - 모든 문자를 대문자로 표시

종류	사용 방법	예
클래스명	각 단어의 첫글자는 대문자	StaffMember, ItemProducer
변수명, 메소드명	소문자로 시작되어 2번째 단어의 첫글자는 대문자	width, height, payRate, accountNumber, getMonthDays()
상수	상수는 모든 글자를 대문자	MAX_NUMBER, PI

Variable

- 변수(Variable)
  - 프로그램 실행 중에 값을 임시 저장하기 위한 공간
    - 변수 값은 프로그램 수행 중 변경될 수 있음
  - 데이터 타입에서 정한 크기의 메모리 할당
  - 반드시 변수 선언과 값을 초기화 후 사용
- 변수 선언
  - 변수의 타입 다음에 변수 이름을 적어 변수를 선언



Variable

- 변수 선언 사례

```
int radius;
char c1, c2, c3; // 3 개의 변수를 한 번에 선언
double weight;
```

- 변수 선언과 초기화
  - 선언과 동시에 초기값 지정

```
int radius = 10;
char c1 = 'a', c2 = 'b', c3 = 'c';
double weight = 75.56;
```

- 변수에 값 대입
  - 대입 연산자인 = 다음에 식(expression)

```
radius = 10 * 5;
c1 = 'r';
weight = weight + 5.0;
```

변수 이름 짓기

- 변수의 이름은 식별자(identifier)의 일종
- 변수 이름의 규칙
  - 식별자는 유니코드 문자와 숫자의 조합(한글 가능!)
  - 식별자의 첫 문자는 일반적으로 유니코드 문자
  - _, $로 시작 가능
  - 두 번째 문자부터는 문자, 숫자, _, $ 등이 가능함
  - 대문자와 소문자는 구별됨
  - 식별자의 이름으로 키워드(keyword)를 사용해서는 안됨

변수 이름의 예

적법한 변수 선언의 예를 들어보면 다음과 같다.

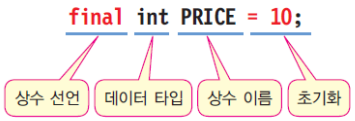
```
int    speed;
long   earthPopulation;
int    _count;          // _로 시작할 수 있다.
long   $value;          // $로 시작할 수 있다.
int    반복횟수;        // 유니코드를 지원하므로 한글 변수 이름도 가능
int    Henry8;          // 맨 처음이 아니라면 숫자도 넣을 수 있다.
```

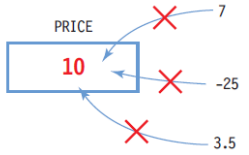
잘못된 변수 선언의 예는 다음과 같다.

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

Constant

- 상수 (Constant)
  - **final** 키워드 사용
  - 변하지 않는 문자나 숫자 값. 값 변경 불가
  - 선언 시 초기값 지정



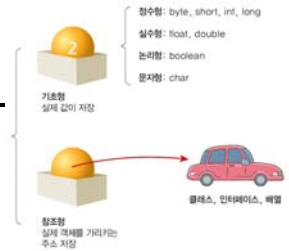


- 상수 선언 사례

```
final double PI = 3.141592;
final int LENGTH = 20;
```

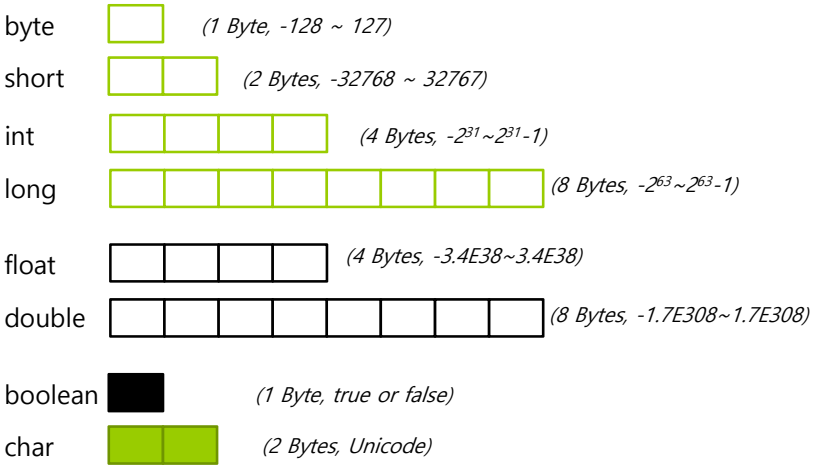
Data Type

- 자바의 자료형 (Data Type)
  - 기초형 (Primitive Type)
    - boolean
    - char
    - byte
    - short
    - int
    - long
    - float
    - double
  - 참조형 (Reference Type)
    - class
    - interface
    - array



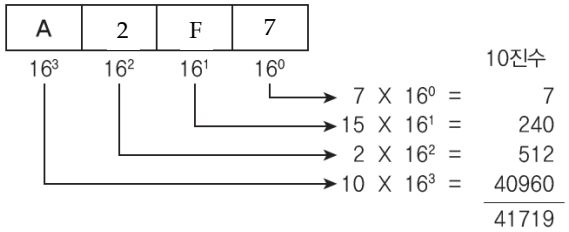
Data Type	Default Value
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'₩u0000'
String (or any object)	null
boolean	false

Primitive Data Types



Integer Literals

- 10진수(decimal): 14, 16, 17
- 8진수(octal): 016, 018, 019
- 16진수(hexadecimal): 0xe, 0x10, 0x11
- 2진수(binary): 0b1100



Integer Literals

- 정수형 리터럴 : 정수 직접 표시
  - 8진수 : 0으로 시작하는 숫자는 모두 8진수
    - int n = 015; // 10진수로 13
  - 16진수 : 0x로 시작하는 숫자는 16진수
    - int n = 0x15; // 10진수로 21
  - 10진수 : 0으로 시작하지 않는 숫자는 10진수
    - 15, 3, 20, 55, 88
  - 모든 정수타입 리터럴은 int형으로 컴파일함
  - long 타입 리터럴은 숫자 뒤에 L 또는 l을 붙임
    - ex) 24L, 3578l

Floating-Point Literals

- 부동 소수점을 갖는 수 직접 표시
  - 소수점을 찍은 실수, 지수(exponent)식으로 표현한 실수
    - 12. 또는 12.0
    - .1234 또는 0.1234 또는 1234E-4
  - 숫자 뒤에 f(float)나 d(double)을 명시적으로 붙이기도 함
    - 0.1234 또는 0.1234D 또는 0.1234d → double 타입
    - 0.1234f 또는 0.1234F → float 타입
    - 1234D 또는 1234d → 1234.0과 같으며 double 타입
    - 1234F 또는 1234f → 1234.0과 같으며 float 타입
  - 실수 타입 리터럴은 double 타입으로 컴파일됨

Character Literals

- 문자는 유니코드 규격 중에서 UTF-16 구격 사용
  - 단일 인용부호('')로 문자 하나 표현
    - 'a', 'W', '가', '*', '3', '7'

```
char ch1 = '가';
char ch2 = '\uac00'; // '가'
```

  - ₩다음에 숫자는 8진수로서 0 ~ 337사이의 8진수만 가능
    - ₩102 -> 문자 'B'를 나타내는 8진수
    - ₩337 -> 문자 'β'를 나타내는 8진수
  - ₩u다음에 4자리 16진수, 2 바이트의 유니코드(Unicode)
    - ₩u0041 -> 문자 'A'의 유니코드(0041)
    - ₩uae00 -> 한글문자 '글'의 유니코드(ae00)
  - 특수 기호는 ₩로 시작

'₩b'	백스페이스	'₩t'	탭
'₩n'	라인피드	'₩f'	폼피드
'₩r'	캐리지리턴	'₩"'	이중 인용 부호
'₩''	단일 인용 부호	'₩₩'	백슬래시

String Literals

- 문자열 리터럴
  - 이중 인용부호로 묶어서 표현
    - "Good", "Morning", "자바", "3.19", "26", "a"
  - 자바에서 문자열은 객체이므로 기본 타입이 아님
  - 자바에서 문자열**(String)**은 문자들의 모임이다. 예를 들어서 문자열 "Hello"는 H, e, l, l, o 등의 5개의 유니코드 문자로 구성됨
  - 문자열 리터럴은 String 객체로 생성됨
- **String 클래스**가 제공됨

```
String str1 = “Welcome”;
String str2 = null;
System.out.println(str1);
```

예제: String

+ 연산자로 문자열을 합칠 수 있다.

StringTest.java

```
01 public class StringTest {
02    public static void main(String args[]) {
03       String s1 = "Hello World!";
04       String s2 = "I'm a new Java programmer!";
05
06       System.out.println(s1 + "\n" + s2);
07    }
08 }
```


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

Boolean

- 논리 값 표시
  - true 또는 false
- 논리 타입과 정수타입 사이의 타입 변환 허용 안 됨

```
int i;
if ((boolean)i) { }// 정수 i를 논리 타입으로 변환할 수 없음
                   // 컴파일 에러
```

  - (i == 1) 또는 (i != 0)과 같은 논리 연산을 사용해야 함

예제: Boolean

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.


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

예제: 변수 초기화 오류

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.


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

예제: 상수

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.


```
60마일은 96.56064킬로미터입니다.
```

예제: 변수, 상수 사용하기

- 지구에서 가장 가까운 별까지 거리 계산하기

CalTime.java

```
01 public class CalTime {
02
03    public static void main(String[] args) {
04       final double light_speed = 30e4;
05       double distance = 40e12;
06
07       double secs;
08
09       secs = distance/light_speed;
10
11       double light_year = secs/(60.0*60.0*24.0*365.0);
12       System.out.println("걸리는 시간은 " + light_year + "광년입니다.");
13    }
14 }
```

final을 붙여서 빛의 속도를
부동소수점형 기호상수로
정의하고 있다.

예제: 변수, 상수 사용하기

- 원의 면적 계산

```
public class CircleArea {
  public static void main(String[] args) {
    final double PI = 3.141592; // 원주율을 상수로 선언
    double radius = 5.0; // 원의 반지름
    double circleArea = 0; // 원의 면적

    circleArea = radius*radius*PI; // 원의 면적 계산

    // 원의 면적을 화면에 출력한다.
    System.out.print("원의 면적 = ");
    System.out.println(circleArea);
  }
}
```

```
원의 면적 = 78.5398
```

Type Conversion

- 자동적인 형 변환(Implicit Type Conversion) 경우
  - 원래의 타입보다 큰 자료타입으로 바뀔 때

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

  - 원본 데이터 그대로 보존
- 자동적인 형 변환 사례

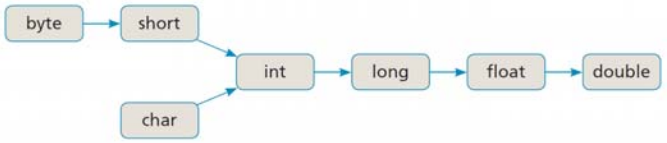
```
byte a;
int price;
price = a;
```

정수타입 변수 / 자동타입변환 / 바이트타입 변수

```
long var;
int n = 32555;
byte b = 25;
var = n; // int → long 타입으로 자동 변환.
         // var 값은 32555
var = b; // byte → long 타입으로 자동 변환
         // var 값은 25
```

Type Conversion

- 자동적인 형변환
  - 피연산자 중 하나가 double형이면 다른 피연산자도 double형으로 변환됨
  - 피연산자 중 하나가 float형이면 다른 피연산자도 float형으로 변환됨
  - 피연산자 중 하나가 long형이면 다른 피연산자도 long형으로 변환됨
  - 그렇지 않으면 모든 피연산자는 int형으로 변환됨



Type Conversion

- 강제 형 변환 (Explicit Type Conversion)
  - 개발자의 의도적으로 타입 변환
- 강제 형 변환 방법

```
byte a;
int price;
a = (byte) price;
```

바이트타입 변수 / price 정수 값을 byte 타입으로 강제타입 변환 / 정수타입 변수

  - 강제 타입 변환 사례
    - 실수 타입이 정수 타입으로 강제 변환 시 소수점 아래가 버려짐
      - 데이터 손실

```
short var;
int n = 855638017; // n의 16진수 값은 0x33000001
var = (short) n; // int 타입에서 short 타입으로 강제 변환. var 값은 1
double d = 1.9;
int n = (int)d; // n은 1이 된다.
```

예제: 형변환

LogicalOperator.java

```
01 public class TypeConversion {
02     public static void main(String args[]) {
03         int i;
04         double f;
05
06         f = 5 / 4;
07         System.out.println(f);
08
09         f = (double) 5 / 4;
10         System.out.println(f);
11
12         i = (int) 1.3 + (int) 1.8;
13         System.out.println(i);
14     }
15 }
```

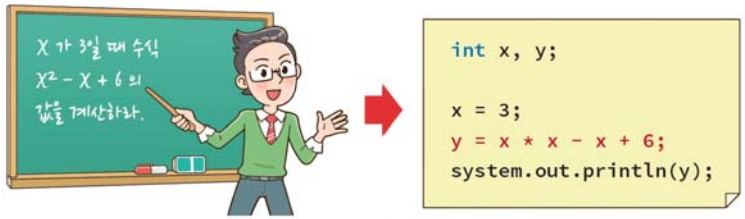
5 / 4는 피연산자가 정수이므로 정수 연산으로 계산되어서 1이 된다. 이것이 double형 변수로 대입되므로 올림 변환이 발생하여 1.0이 f에 저장된다.

(double)5 / 4에서는 먼저 형변환 연산자가 우선순위가 높기 때문에 먼저 실행되어서 정수 5가 부동소수점수 5.0으로 변환된다. 5.0 / 4는 피연산자중 하나가 double형이므로 4도 double형으로 자동 형변환되고 5.0 / 4.0 으로 계산되어서 1.25가 수식의 결과값이 된다.

수식 (int)1.3 + (int)1.8에서는 1.3과 1.8이 모두 1로 변환되므로 변수 i에는 1 + 1하여 2가 저장된다.

수식

- 수식이란 상수나 변수, 함수와 같은 피연산자들과 연산자의 조합

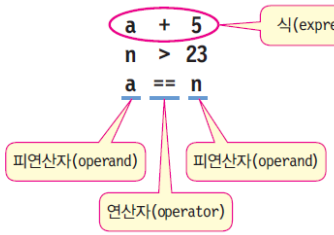



```
int x, y;

x = 3;
y = x * x - x + 6;
system.out.println(y);
```

그림 2-2 • 수식의 예

식과 연산자

- 연산 : 주어진 식을 계산하여 결과를 얻어내는 과정



연산자의 종류	연산자
증감	++ --
산술	+ - * / %
시프트	>> << >>>
비교	> < >= <= == !=
비트	& | ^ ~
논리	&& || ! ^
조건	? :
대입	= *= /= += -= &= ^= |= <<= >>= >>>=

Operator Precedence

높음 ↓ 낮음

++(postfix) -- (postfix)
+(양수 부호) -(양수, 음수 부호) ++(prefix) --(prefix) ~ !
형 변환(type casting)
* / %
+(덧셈) –(뺄셈)
<< >> >>>
< > <= >= instanceof
== !=
&(비트 AND)
^(비트 XOR)
|(비트 OR)
&&(논리 AND)
||(논리 OR)
? : (조건)
= += -= *= /= %= &= ^= |= <<= >>= >>>=

- 같은 우선순위의 연산자
  - 왼쪽에서 오른쪽으로 처리
  - 예외)오른쪽에서 왼쪽으로
    - 대입 연산자, --, ++, +,-(양수 음수 부호), !, 형 변환은 오른쪽에서 왼쪽으로 처리
- 괄호는 최우선순위
  - 괄호가 다시 괄호를 포함한 경우는 가장 안쪽의 괄호부터 먼저 처리

Assignment Operators

- 대입 연산자(=)는 왼쪽 변수에 오른쪽 수식의 값을 계산하여 저장
- 대입 연산자 == 할당 연산자 == 배정 연산자라고 함

x = 100; // 상수 100을 변수 x에 대입

그림 2-3 • 대입 연산자는 변수에 값을 저장하는 연산자이다.

Arithmetic Operators

산술 연산자	의미	예	결과값
+	더하기	25.5 + 3.6	29.1
-	빼기	3 - 5	-2
*	곱하기	2.5 * 4	10.0
/	나누기	5/2	2
%	나머지	5%2	1

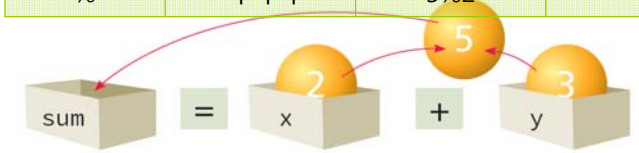


그림 2-4 • 산술 연산의 과정: 먼저 x와 y에서 값을 가져와서 덧셈연산이 수행되고 그 결과값이 sum에 저장된다.

예제: 윤년 검사 프로그램

LeapYear.java

```
01 public class LeapYear {
02    public static void main(String[] args) {
03
04       int year = 2012;
05       boolean isLeapYear;
06
07       isLeapYear = (year % 4 == 0);
08       System.out.println(isLeapYear);
09    }
10 }
```

4로 나누어지는 연도는 윤년 후보이다.

Unary Operators: Postfix & Prefix

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

예제: 증감 연산자

UnaryOperator.java

```
01 public class UnaryOperator {
02    public static void main(String[] args) {
03       int x = 1;
04       int y = 1;
05
06       int nextx = ++x; // x의 값이 사용되기 전에 증가된다. nextx는 2가 된다.
07       int nexty = y++; // y의 값이 사용된 후에 증가된다. nexty는 1이 된다.
08       System.out.println(nextx);
09       System.out.println(nexty);
10    }
11 }
```


```
2
1
```

Equality & Relational Operators

비교 연산자	내용	예제	결과
a < b	a가 b보다 작으면 true 아니면 false	3 < 5	true
a > b	a가 b보다 크면 true 아니면 false	3 > 5	false
a <= b	a가 b보다 작거나 같으면 true 아니면 false	1 <= 0	false
a >= b	a가 b보다 크거나 같으면 true 아니면 false	10 >= 10	true
a == b	a가 b와 같으면 true 아니면 false	1 == 3	false
a != b	a가 b와 같지 않으면 true 아니면 false	1 != 3	true

예제: 관계 연산자

ComparisonOperator.java

```
01 public class ComparisonOperator {
02
03     public static void main(String[] args){
04         int x = 3;
05         int y = 4;
06         System.out.print((x == y) +" ");
07         System.out.print((x != y) +" ");
08         System.out.print((x > y) +" ");
09         System.out.print((x < y) +" ");
10         System.out.print((x <= y) +" ");
11     }
12 }
```


```
false true false true true
```

Logical Operators

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

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

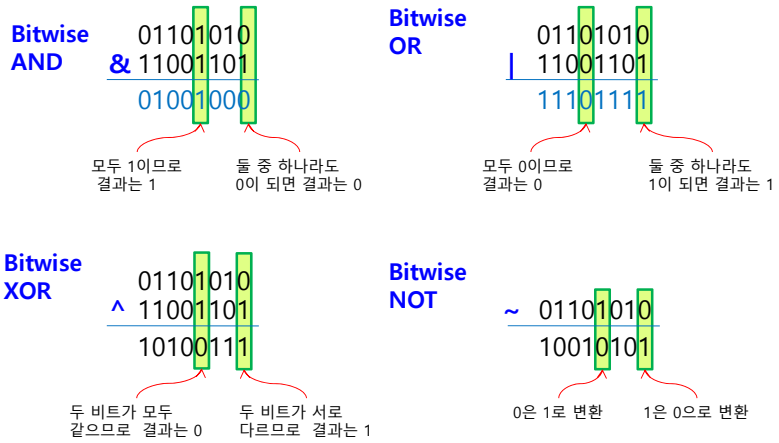

```
false
true
```

Bitwise Operators

- 피 연산자의 각 비트들을 대상으로 하는 연산

비트 연산자	내용
a & b	a와 b의 각 비트들의 AND 연산. 두 비트 모두 1일 때만 1이 되며 나머지는 0
a | b	a와 b의 각 비트들의 OR 연산. 두 비트 모두 0일 때만 0이 되며 나머지는 1
a ^ b	a와 b의 각 비트들의 XOR 연산. 두 비트가 서로 다르면 1, 같으면 0
~ a	단항 연산자로서, a의 각 비트들에 NOT 연산. 1을 0으로, 0을 1로 변환

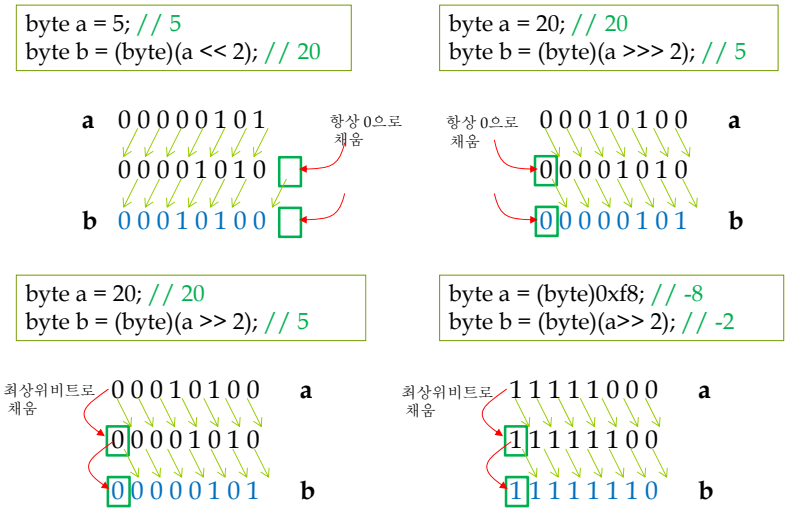
비트 연산자의 사례



Bit Shift Operators

시프트 연산자	내용
a >> b	a의 각 비트를 오른쪽으로 b 번 시프트한다. 최상위 비트의 빈자리는 시프트 전의 최상위 비트로 다시 채운다. 산술적 오른쪽 시프트.
a >>> b	a의 각 비트를 오른쪽으로 b 번 시프트한다. 그리고 최상위 비트의 빈자리는 0으로 채운다. 논리적 오른쪽 시프트.
a << b	a의 각 비트를 왼쪽으로 b 번 시프트한다. 그리고 최하위 비트의 빈자리는 0으로 채운다. 산술적 왼쪽 시프트.

시프트 연산자의 사례



Ternary Operator

- opr1?opr2:opr3
  - 세 개의 피연산자로 구성된 삼항(ternary) 연산자
    - opr1이 true이면, 연산식의 결과는 opr2, false이면 opr3
  - if-else을 간결하게 표현할 수 있음

```
int x = 5;
int y = 3;

int s;
if(x>y)
  s = 1;
else
  s = -1;
```

→

```
int s = (x>y) ? 1 : -1;
```

예제 : 조건 연산자 사용하기

다음 소스의 실행 결과는 무엇인가?

```
public class TernaryOperator {
  public static void main (String[] args) {
    int a = 3, b = 5;

    System.out.println("두 수의 차는 " + ((a>b) ? (a-b) : (b-a)));
  }
}
```

```
두 수의 차는 2
```

LAB: 2차 방정식의 근을 계산

- x*x+b*x+c 식의 근을 계산

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 }
```


```
근은 2.0
근은 1.0
```

사용자로부터 값을 입력 받으려면?

- Scanner 클래스를 사용

```
import java.util.*;                    // Scanner 클래스 포함

Scanner input = new Scanner(System.in);

System.out.print("문장을 입력하시오: ");
String line = input.nextLine();        // 한 줄을 읽는다.
```

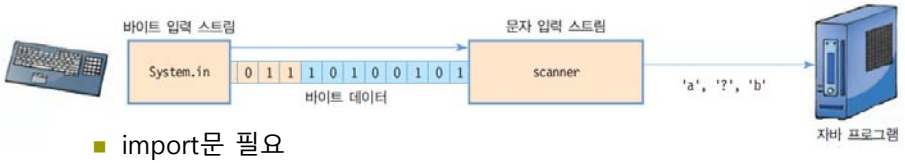
import 문장

- import java.util.Scanner; // Scanner 클래스 포함

- Scanner 클래스를 포함시키는 문장
- Scanner는 자바 클래스 라이브러리(Java Class Library)의 일종
- Scanner는 입력을 받을 때 사용

Scanner

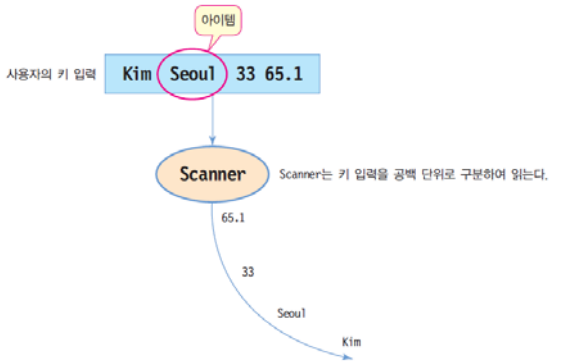
- Scanner 클래스를 이용한 키 입력
  - java.util.Scanner 클래스
  - Scanner 객체 생성 `Scanner a = new Scanner(System.in);`



  - import문 필요
    - 소스 맨 앞줄에 사용 `import java.util.Scanner;`
  - Scanner에서 키 입력 받기
    - Scanner는 입력되는 키 값을 공백으로 구분되는 아이템 단위로 읽음
    - 공백 문자 :'₩t',' ₩f',' ₩r',' ',' ₩n'

예제: Scanner

```
Scanner scanner = new Scanner(System.in);
String name = scanner.next();              // "Kim"
String addr = scanner.next();              // "Seoul"
int age = scanner.nextInt();               // 33
double weight = scanner.nextDouble();      // 65.1
```

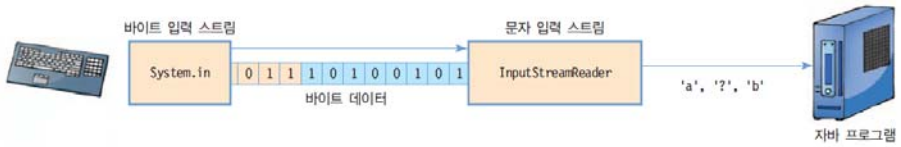


Scanner 주요 메소드

생성자/메소드	설명
String next()	다음 아이템을 찾아 문자열로 반환
boolean nextBoolean()	다음 아이템을 찾아 boolean으로 변환하여 반환
byte nextByte()	다음 아이템을 찾아 byte로 변환하여 반환
double nextDouble()	다음 아이템을 찾아 double로 변환하여 반환
float nextFloat()	다음 아이템을 찾아 float로 변환하여 반환
int nextInt()	다음 아이템을 찾아 int로 변환하여 반환
long nextLong()	다음 아이템을 찾아 long으로 변환하여 반환
short nextShort()	다음 아이템을 찾아 short로 변환하여 반환
String nextLine()	한 라인 전체('₩n' 포함)를 문자열 타입으로 반환

System.in

- 자바에서 키 입력: System.in
  - 자바의 표준 입력 스트림
  - java.io 패키지의 InputStream 클래스
    - System.in은 바이트 스트림으로서 키 값을 바이트로 리턴
    - 문자로 변환하려면 InputStreamReader 클래스를 이용



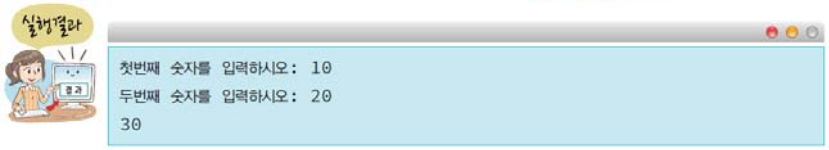
  - 입력 동안 문제가 발생하면 IOException 발생
    - try-catch를 이용한 예외 처리 필요

예제: 키보드로부터 문자 입력, 화면 출력

다음 소스의 실행 결과는 무엇인가?
System.in을 InputStreamReader에 연결하여 사용자로부터 키 입력.
입력 받은 문자를 화면에 출력하고, ctrl-z 키를 누르면 읽기 종료

```
import java.io.*;
public class InputExample {
  public static void main (String args[]) {
    InputStreamReader rd = new
                InputStreamReader(System.in);
    try {
        int a = rd.read();
        if (a == -1)// ctrl-z가 입력되면 read()는 -1을 리턴
          break;
        System.out.println((char)a); // 입력된 문자 출력
    }
    catch (IOException e) {
      System.out.println("입력 에러 발생");
    }
  }
}
```

예제: 사용자로 부터 입력을 받아 더하기

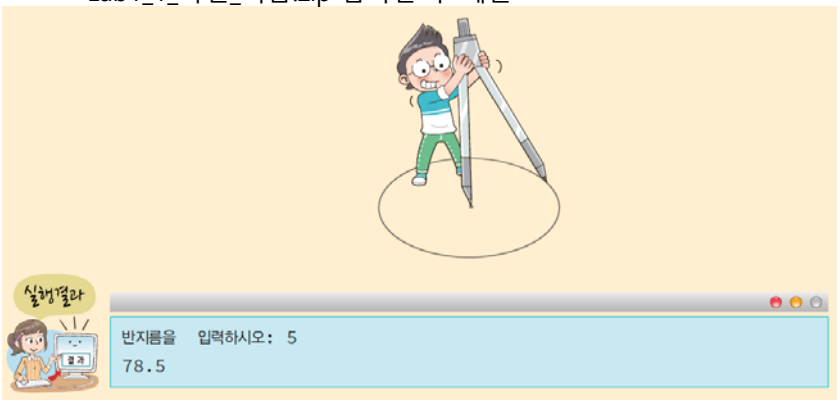


Add2.java

```
// 사용자가 입력한 두 개의 숫자를 더해서 출력한다.
import java.util.Scanner; // Scanner 클래스 포함

public class Add2 {
   // 메인 메소드에서부터 실행이 시작된다.
   public static void main(String args[]) {

      Scanner input = new Scanner(System.in);
      int x; // 첫 번째 숫자 저장
      int y; // 두 번째 숫자 저장
      int sum; // 합을 저장

      System.out.print("첫번째 숫자를 입력하시오: "); // 입력 안내 출력
      x = input.nextInt(); // 사용자로부터 첫 번째 숫자를 읽는다.

      System.out.print("두번째 숫자를 입력하시오: "); // 입력 안내 출력
      y = input.nextInt(); // 사용자로부터 두 번째 숫자를 읽는다.

      sum = x + y; // 두 개의 숫자를 더한다.

      System.out.println(sum); // 합을 출력한다.

   }
}
```

LAB1 원의 면적 계산하기

- 원의 면적 계산 프로그램을 작성한다.
  - Method 사용
  - 프로젝트 디렉토리 안에 보고서 (1~2장)를 넣고 Lab1_1_학번_이름.zip 압축한 후 제출



LAB1 직사각형의 둘레와 면적 계산하기

- 직사각형의 둘레와 면적 계산 프로그램을 작성한다.
  - 프로젝트 디렉토리 안에 보고서 (1~2장)를 넣고 Lab1_2_학번_이름.zip 압축한 후 제출