

(INTERMEDIATE) JAVA PROGRAMMING

3. Java Basics: Chapter 2

Today's Topic

- Basic Java Programming
 - Java Program Structure
 - Datatypes
 - Expressions

JAVA: THE LANGUAGE

Java Program Structure

```
// comments about the class
```

```
public class MyProgram
```

```
{
```

class header

A diagram illustrating the structure of a Java program. It shows a code snippet with three lines: a comment line, a class declaration line, and an opening curly brace. An orange arrow points from the text 'class header' to the class declaration line. A large orange curly brace spans from the opening curly brace to the closing curly brace, with the text 'class body' next to it. Below the code, the text 'Comments can be placed almost anywhere' is displayed.

class body

Comments can be placed almost anywhere

```
}
```

Java Program Structure

```
// comments about the class
```

```
public class MyProgram
```

```
{
```

```
    // comments about the method
```

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

```
    {
```

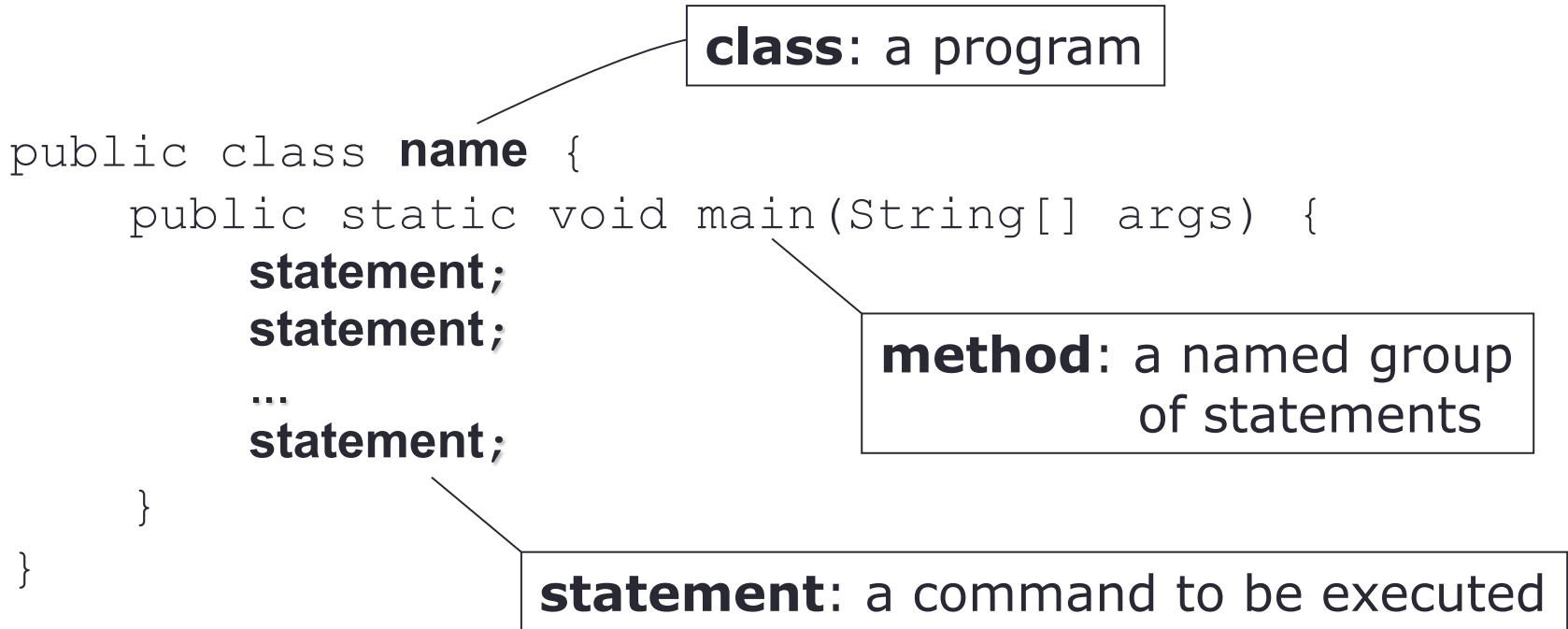
} method body

```
    }
```

```
}
```

method header

Structure of a Java program



- Every executable Java program consists of a **class**,
 - that contains a **method** named **main**,
 - that contains the **statements** (commands) to be executed.

Names and identifiers

- You must give your program a name.

```
public class GangstaRap {
```

- Naming convention: capitalize each word (e.g. `MyClassName`)
- Your program's file must match exactly (`GangstaRap.java`)
 - includes capitalization (Java is "case-sensitive")
- **identifier**: A name given to an item in your program.
 - must start with a letter or `_` or `$`
 - subsequent characters can be any of those or a number
 - **legal:** `_myName` `TheCure` `ANSWER_IS_42` `$bling$`
 - **illegal:** `me+u` `49ers` `side-swipe` `Ph.D's`

Keywords

- **keyword:** An identifier that you cannot use because it already has a reserved meaning in Java.

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

- i.e., You may not use `char` or `while` for the name of a class.
- `true`, `false` and `null` are not reserved keywords, but still cannot be used as the name of an identifier.

좋은 이름 붙이는 언어 관습

- 기본 : 가독성 높은 이름
 - 목적을 나타내는 이름 붙이기 : s 보다 sum
 - 충분히 긴 이름으로 붙이기 : AVM보다 AutoVendingMachine
- 자바 언어의 이름 붙이는 관습 : **Hungarian Notation**
 - 클래스 이름
 - 첫 번째 문자는 대문자로 시작
 - 각 단어의 첫 번째 문자만 대문자
 - 변수, 메소드 이름
 - 첫 단어 이후 각 단어의 첫 번째 문자는 대문자로 시작
 - 상수 이름
 - 모든 문자를 대문자로 표시

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

```
int myAge;  
boolean isSingle;  
public int getAge() {}
```

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

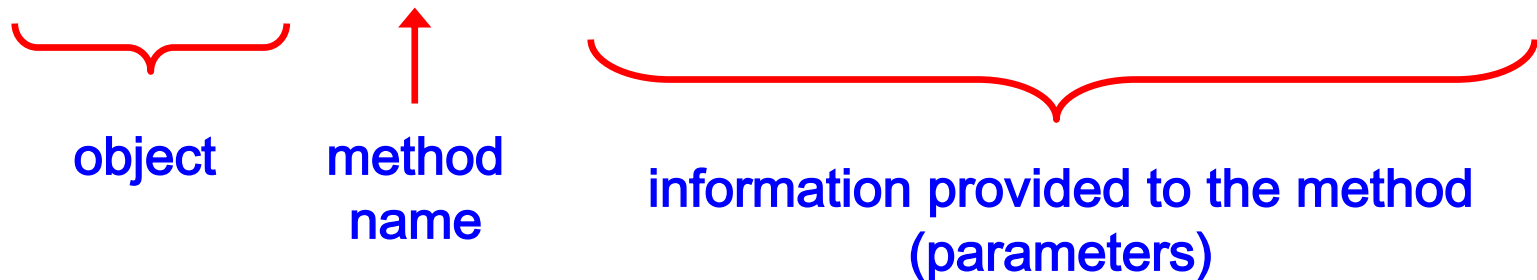
System.out.println

- A statement that prints a line of output on the console.
 - pronounced "print-linn"
 - sometimes called a "println statement" for short
- Two ways to use `System.out.println` :
 - `System.out.println("text") ;`
Prints the given message as output.
 - `System.out.println() ;`
Prints a blank line of output.

System.out object

- The `System.out` object represents a destination (the monitor screen) to which we can send output

```
System.out.println ("Whatever you are, be a good one.");
```



Try these:

- `System.out.println(123);`
- `int age = 20;`
`System.out.println(age);`
- `System.out.println("I am "+age+"years old");`
- `System.out.printf("I am %d years old", age);`
- `System.out.println("24 + 45 = "+24+45);`


JAVA:

VARIABLES AND DATA TYPES

Variables

- A *variable* is a name for a location in memory
- A variable must be *declared* by specifying the variable's name and the type of information that it will hold

data type variable name



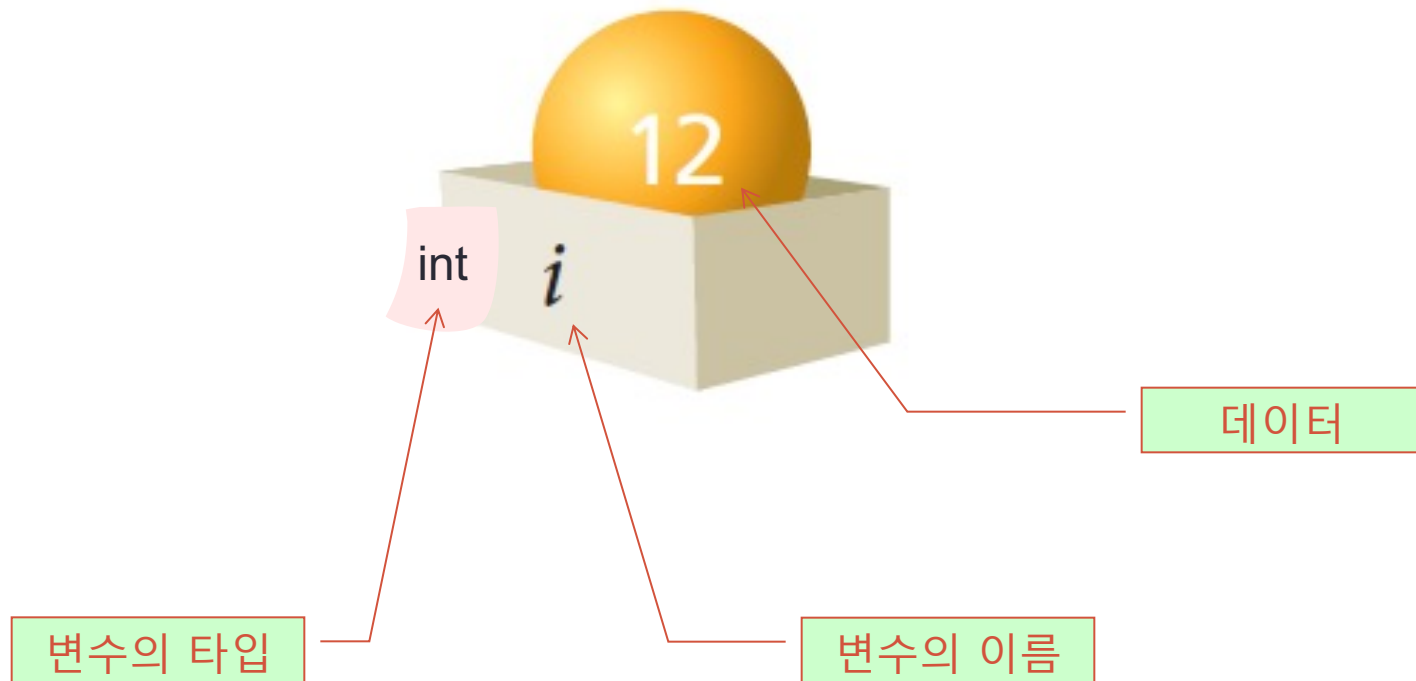
```
int total;
```

```
int count, temp, result;
```

Multiple variables can be created in one declaration

변수

- 변수(**variable**) : 데이터 값들이 저장되는 메모리 공간



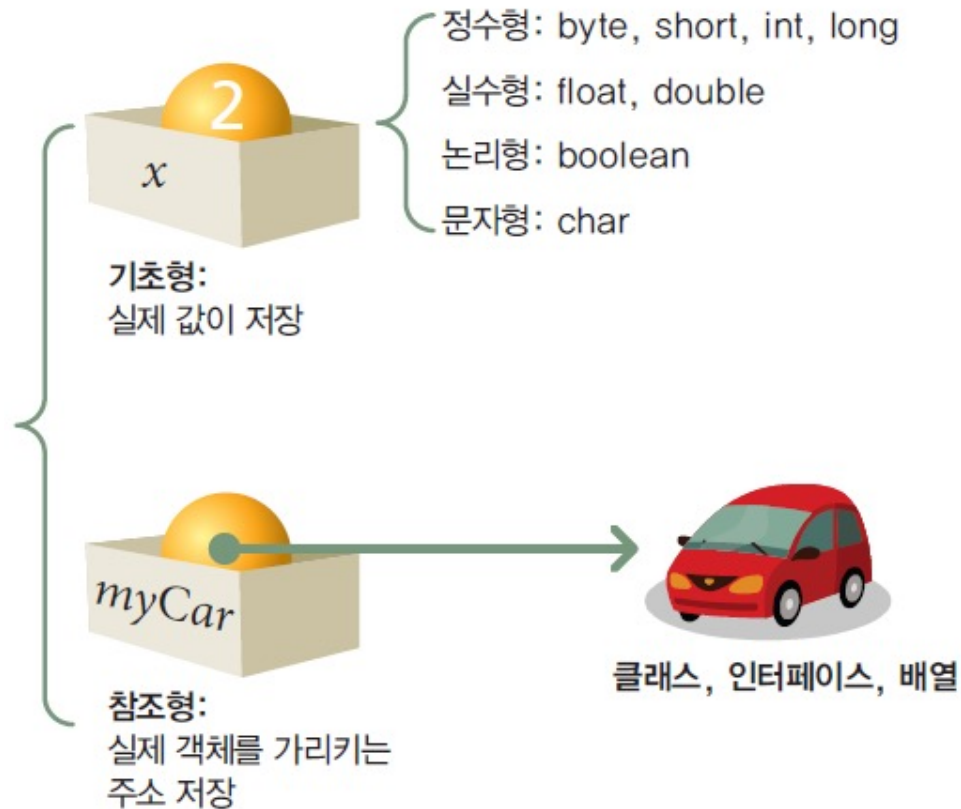
자료형

- **자료형(data type):** 자료의 타입
- 물건을 정리하는 상자도 다양한 타입이 있듯이 자료를 저장하는 변수도 다양한 종류가 있다.

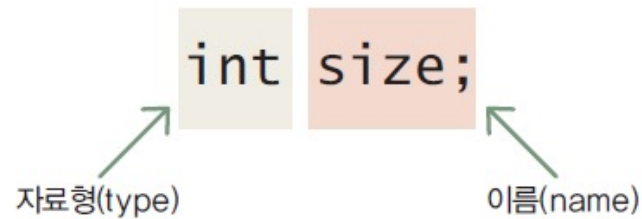


자료형의 분류

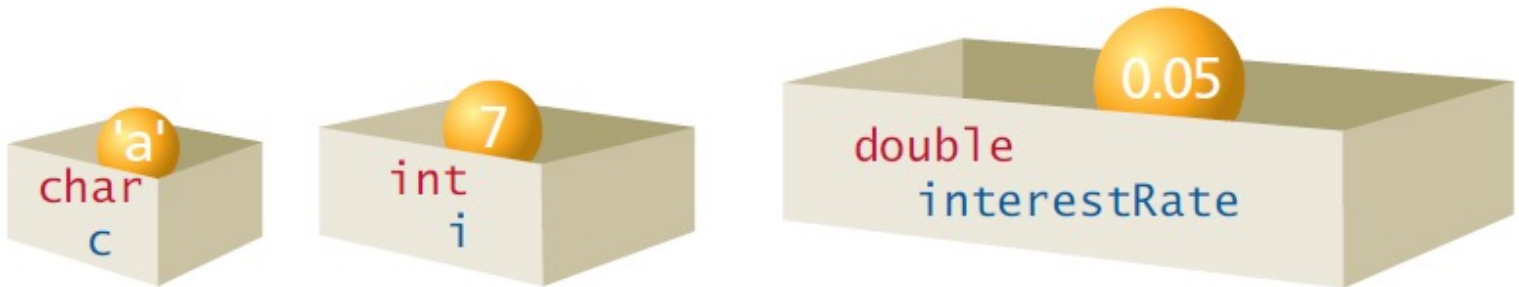
- 기본 타입과 레퍼런스(참조) 타입으로 나뉘어진다.



변수의 선언과 초기화



```
char c = 'a';  
int i = 7;  
double interestRate = 0.05;
```



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.

식별자의 관례

| 종류 | 사용 방법 | 예 |
|-----------|---------------------------------|---|
| 클래스명 | 각 단어의 첫글자는 대문자로 한다. | StaffMember, ItemProducer |
| 변수명, 메소드명 | 소문자로 시작되어 2번째 단어의 첫글자는 대문자로 한다. | width, payRate, acctNumber, getMonthDays(), fillRect(), |
| 상수 | 상수는 모든 글자를 대문자로 한다. | MAX_NUMBER |

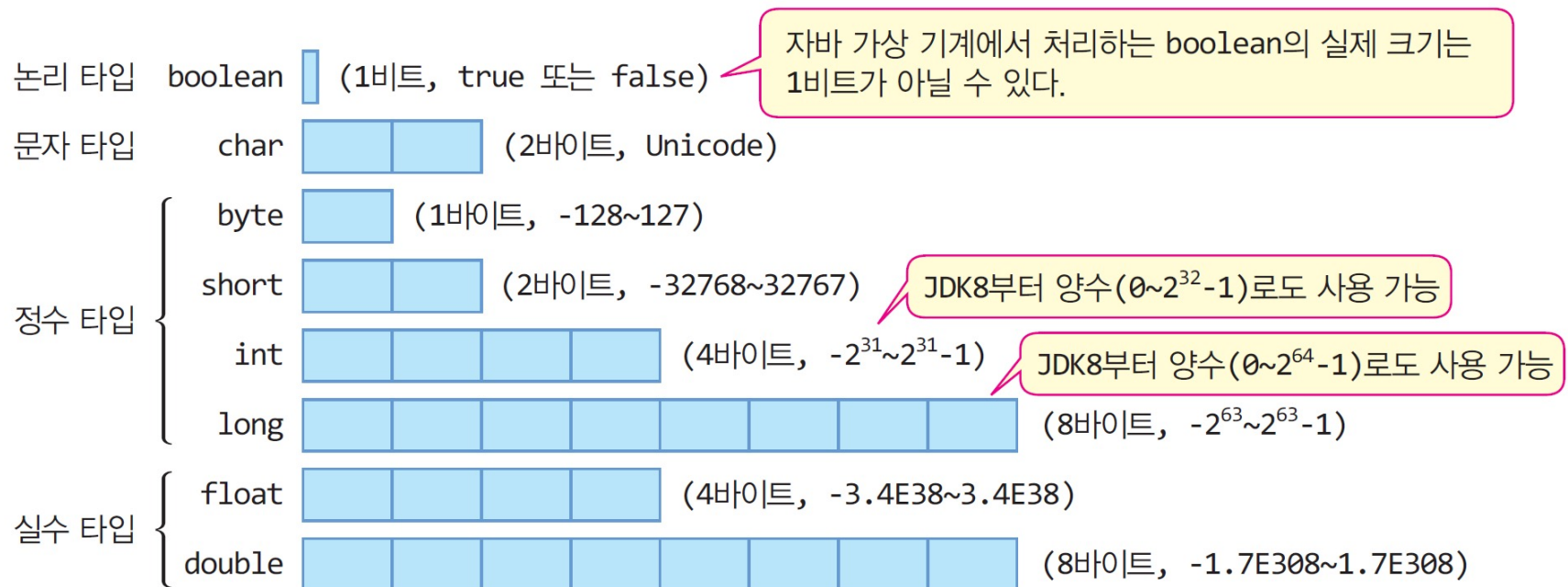
Primitive Data Types

- There are eight primitive data types in Java
- Four of them represent integers:
 - `byte`, `short`, `int`, `long`
- Two of them represent floating point numbers:
 - `float`, `double`
- One of them represents characters:
 - `char`
- And one of them represents boolean values:
 - `boolean`

참고: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

자바의 기본 타입

- 특징
 - 기본 타입의 크기가 정해져 있음
 - CPU나 운영체제에 따라 변하지 않음



Primitive Data Types

- The difference between the various numeric primitive types is their size, and therefore the values they can store:

| 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 | 32 | 0.0f | 약 $\pm 3.4 \times 10^{-38}$ (유효숫자 7개) | 약 $\pm 3.4 \times 10^{+38}$ (유효숫자 7개) |
| double | 실수 | 64 | 0.0d | 약 $\pm 1.7 \times 10^{-308}$ (유효숫자 15개) | 약 $\pm 1.7 \times 10^{+308}$ (유효숫자 15개) |
| char | 문자(유니코드) | 16 | null | '\u0000' (0) | '\uFFFF' (65535) |
| boolean | true 또는 false | 8 | false | 해당 없음 | 해당 없음 |

Data Types

Storage affects the possible values for a piece of data.

- Java uses 4 bytes = 32 bits to store the value of an integer
 - $2^{31} = 2147483648$
 - For tracking the sign, it uses one bit.
 - a Java integer has possible values from -2147483648 to +2147483647
 - $2147483647 + 1 \rightarrow$ garbage
 - **Memorize this:** “the minimum value for a Java integer is approximately negative 2 billion. The maximum value is approximately positive 2 billion.”

정수형

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

(Q) 만약 다음과 같이 정수형의 변수에 범위를 벗어나는 값을 대입하면 어떻게 될까?

byte number = 300;// 오류!!

(A) 컴파일 오류가 발생한다.

리터럴과 정수 리터럴

- 리터럴(literal)
 - 프로그램에서 직접 표현한 값
 - 정수, 실수, 문자, 논리, 문자열 리터럴 있음
 - 사례) 34, 42.195, ' % ', true, " hello "
- 정수 리터럴
 - 10진수, 8진수, 16진수, 2진수 리터럴

| | |
|--------|---------------------------|
| 15 | -> 10진수 리터럴 15 |
| 015 | -> 0으로 시작하면 8진수. 십진수로 13 |
| 0x15 | -> 0x로 시작하면 16진수. 십진수로 21 |
| 0b0101 | -> 0b로 시작하면 2진수. 십진수로 5 |



```
int n = 15;  
int m = 015;  
int k = 0x15;  
int b = 0b0101;
```

- 정수 리터럴은 int 형으로 컴파일
- long 타입 리터럴은 숫자 뒤에 L 또는 l을 붙여 표시
 - ex) long g = 24L;

예제

```
01 class Light {  
02     public static void main(String args[]) {  
03         long lightspeed;  
04         long distance; ←----- 64비트로 정수를 저장한다.  
05  
06         lightspeed = 300000;  
07         distance = lightspeed * 365L * 24 * 60 * 60;  
08  
09         System.out.println("빛이 1년 동안 가는 거리 : " + distance + " km.");  
10     }  
11 }
```

실행결과

빛이 1년 동안 가는 거리 : 9460800000000 km.

예제

```
public class IntegerLiterals {  
    public static void main(String args[]) {  
        int i = 26;           // 26을 10진수로 표현  
        int oi = 032;         // 26을 8진수로 표현  
        int xi = 0x1a;        // 26을 16진수로 표현  
  
        System.out.println(i);  
        System.out.println(oi);  
        System.out.println(xi);  
    } // end method main  
  
} // end class IntegerLiterals
```

실행결과

26
26
26

실수형

- float는 32비트를 이용하여 실수를 표현
- double은 64비트를 이용하여 실수를 표현
- float는 약 7개 정도의 유효 숫자
- double은 약 15개 정도의 유효 숫자
- 대부분의 경우에는 double을 사용하는 것이 바람직

실수형 상수

| 일반 표기법 | 과학적 표기법 | 지수 표기법 |
|---------|----------------------|-----------|
| 146.91 | 1.4691×10^2 | 1.4691E+2 |
| 0.00081 | 8.1×10^{-4} | 8.1E-4 |
| 1800000 | 1.8×10^6 | 1.8E+6 |

- 다음 문장이 오류가 나는 이유는?

```
float temperature = 25.6;           // 25.6은 double형이므로 오류!
```

```
float temperature = 25.6F;          // OK!
```

- JDK 7부터 실수형 상수에도 밑줄 기호를 사용할 수 있다.

```
double number = 123_456_789.0;      // 밑줄 기호 사용 가능
```

Characters

- A **char** variable stores a single character
- Character literals are delimited by single quotes:

`'a' 'X' '7' '$' ',' '\n'`

- Example declarations:

```
char topGrade = 'A';
```

```
char terminator = ';', separator = ' ';
```

- Note the distinction between a primitive character variable, which holds only one character, and a **String** object, which can hold multiple characters

Character Sets

byte ~~=~~ char

- A *character set* is an ordered list of characters, with each character corresponding to a unique number
- A `char` variable in Java can store any character from the *Unicode character set*
- The Unicode character set uses **sixteen bits per character**, allowing for 65,536 unique characters
- It is an international character set, containing symbols and characters from many world languages



참고사항 유니코드

유니코드(unicode)는 전세계의 모든 문자를 컴퓨터에서 일관되게 표현하고 다룰 수 있도록 설계된 산업 표준이다. 유니코드 협회(unicode consortium)가 제정하며, 현재 최신판은 유니코드 5.0이다. 이 표준에는 문자 집합, 문자 인코딩, 문자 정보 데이터베이스, 문자들을 다루기 위한 알고리즘 등이 포함된다.

문자 리터럴

- 단일 인용부호(' ')로 문자 표현
 - 사례) 'w', 'A', '가', '*', '3', '글', \u0041
 - \u다음에 4자리 16진수(2바이트의 유니코드)
 - '\u0041' -> 문자 'A'의 유니코드(0041)
 - '\uae00' -> 한글문자 '글'의 유니코드(ae00)

```
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) |

String

- 자바에서는 문자열은 **String** 클래스로 제공된다.
- 문자열 변수를 선언하려면 **String** 타입을 사용한다.

```
String s = "Hello World!";  
System.out.println(s);
```

예제

```
public class CharTest {
```

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

```
        char c;
```

문자형 변수

```
        c = 'a';
```

```
        System.out.println(c);
```

```
        c = '가';
```

```
        System.out.println(c);
```

문자열

```
        String s = "Hello World!";
```

```
        System.out.println(s);
```

```
    }
```

```
}
```



Boolean

- A `boolean` value represents a true or false condition
- The reserved words `true` and `false` are the only valid values for a boolean type

```
boolean done = false;
```

- A `boolean` variable can also be used to represent any two states, such as a light bulb being on or off




오류주의

C나 C++ 언어에서는 정수값이 논리형으로 사용된다. 0은 `false`에 해당되고 나머지값은 `true`에 해당된다. 그러나 자바에서는 그렇지 않다. 따라서 정수값을 논리형으로 형변환할 수 없다.

논리 리터럴

- 논리 리터럴은 2개 뿐
 - true, false
 - boolean 타입 변수에 치환하거나 조건문에 이용

 boolean a = true;
boolean b = 10 > 0; // 10>0가 참이므로 b 값은 true
boolean **c** = **1**; // 타입 불일치 오류. C/C++와 달리 자바에서 1,0을 참, 거짓으로 사용 불가

while(**true**) { // 무한 루프. while(1)로 사용하면 안 됨
...
}

예제

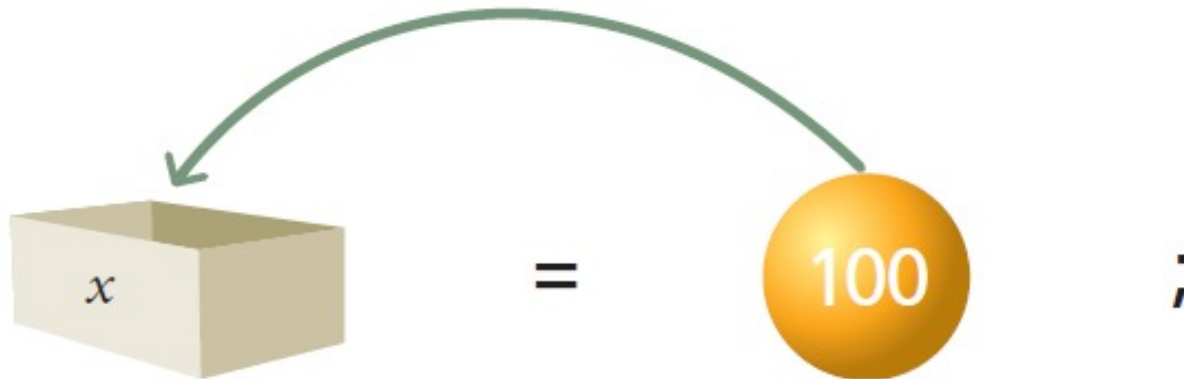
```
01 class BooleanTest {
02     public static void main(String args[]) {
03         boolean b; ←----- true 또는 false의 값만 가진다.
04
05         b = true;
06         System.out.println("b : " + b);
07         b = (1 > 2);
08         System.out.println("b : " + b);
09
10     }
11 }
```

실행결과

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

Assignment (대입 연산자)

- 대입 연산자 == 할당 연산자 == 배정 연산자
- 변수에 값을 저장하는 연산자
- (예) $x = 100;$




Tip: JDK7부터 숫자에 '_' 허용, 가독성 높임

- 숫자 리터럴의 아무 위치에나 언더스코어('_') 허용
 - 컴파일러는 리터럴에서 '_'를 빼고 처리
- 사용 예

```
int price = 20_100; // 20100과 동일
long cardNumber = 1234_5678_1357_9998L; // 1234567813579998L와 같음
long controlBits = 0b10110100_01011011_10110011_111110000;
long maxLong = 0x7fff_ffff_ffff_ffffL;
int age = 2____5; // 25와 동일
```

- 허용되지 않는 4가지 경우

 오류

```
int x = 15_; // 오류. 리터럴 끝에 사용할 수 없다.
double pi = 3_.14; // 오류. 소수점(.) 앞뒤에 사용할 수 없다.
long idNum = 981231_1234567_L; // 오류. _L(_F) 앞에 사용할 수 없다.
int y = 0_x15; // 오류. 0x 중간이나 끝에 사용할 수 없다. 0x_15(오류)
```


Tip: 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(4장 참조)
var v = new Vector<Integer>(); // v는 Vector<integer> 타입으로 결정(7장 참조)
```

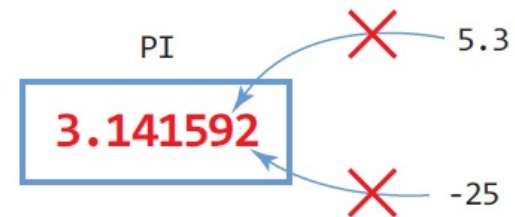
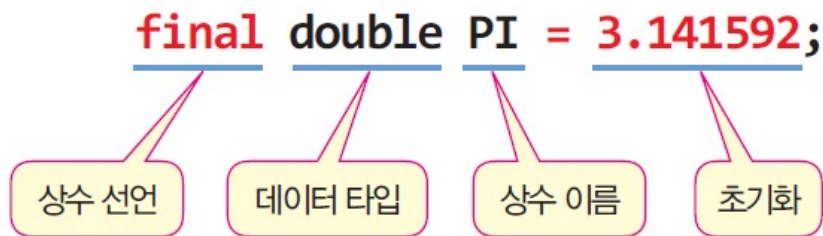
- 변수 선언문에 반드시 초기값 지정



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

상수

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



예제 2-2 : 변수, 리터럴, 상수 활용

원의 면적을 구하는 프로그램을 작성해보자.

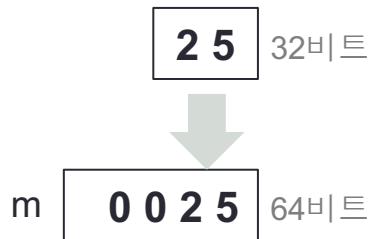
```
public class CircleArea {  
  
    public static void main(String[] args) {  
        final double PI = 3.14; // 원주율을 상수로 선언  
  
        double radius = 10.0; // 원의 반지름  
        double circleArea = radius*radius*PI; // 원의 면적 계산  
  
        // 원의 면적을 화면에 출력한다.  
        System.out.println("원의 면적 = " + circleArea);  
    }  
}
```

원의 면적 = 314.0

자동 타입 변환

- 자동 타입 변환
 - 작은 타입은 큰 타입으로 자동 변환
 - 컴파일러에 의해 이루어짐
 - 치환문(=)이나 수식 내에서 타입이 일치하지 않을 때

`long m = 25;` // 25는 int 타입. 25가 long 타입으로 자동 변환되는 사례



`double d = 3.14 * 10;` // 실수 연산을 하기 위해 10이 10.0으로 자동 변환
// 다른 피연산자 3.14가 실수이기 때문

강제 타입 변환

- 자동 타입 변환이 안 되는 경우 : 큰 타입이 작은 타입으로 변환할 때

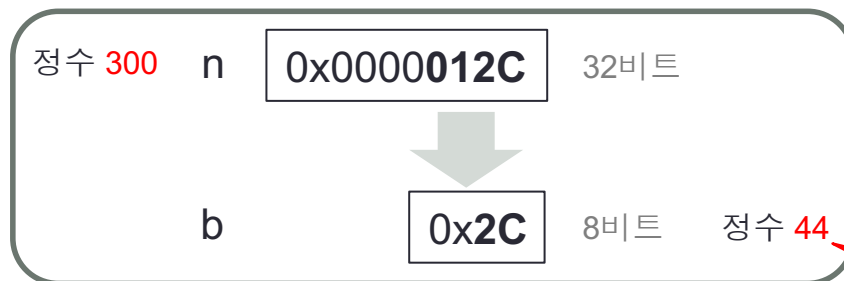
오류

```
int n = 300;
byte b = n; // 컴파일 오류. int 타입이 byte로 자동 변환 안 됨
```

강제 타입 변환하려면, byte b = (byte)n; 로 수정

- 강제 타입 변환
 - 개발자가 필요하여 강제로 타입 변환을 지시
 - () 안에 변환할 타입 지정
 - 강제 변환은 값 손실 우려

byte b = (byte)n; 에 따른 손실



$$44 = 300 \% 256$$

```
double d = 1.9;
int n = (int)d; // n = 1
```

강제 타입 변환으로
소숫점 이하 0.9 손실

예제 2-3 : 타입 변환

자동 타입 변환과 강제 타입 변환의 이해를 위한 예제이다.
다음 소스의 실행 결과는 무엇인가?

```
public class TypeConversion {
    public static void main(String[] args) {
        byte b = 127;
        int i = 100;

        System.out.println(b+i);
        System.out.println(10/4);
        System.out.println(10.0/4);
        System.out.println((char)0x12340041);
        System.out.println((byte)(b+i));
        System.out.println((int)2.9 + 1.8);
        System.out.println((int)(2.9 + 1.8));
        System.out.println((int)2.9 + (int)1.8);
    }
}
```

강제 타입 변환 결과 \u0041이며, 문자 A의 코드임

227
2
2.5
A
-29
3.8
4
3