1장 프로그래밍 기본

남광우

목 차

Hello Java!

- // import 구문은 이 프로그램에서 사용할 클래스를 선언!
- import java.lang.*;

- /* public 클래스는 한 파일에 하나! Public클래스의 이름은 파일이름과 똑같이!!!*/
- public class Hello
- {
- // main 메소드의 선언은 public static void !
- public static void main(String arg[ ])
- {
- // java.lang 패키지 아래의 System 클래스를 사용
- System.out.println("Hellow World");
- }
- }

Hello Java!

- Compile

C:/> javac Hello.java

=> Hello.class

C:/> java Hello.class

Hello You!

– 예제 프로그램 1

```
class Sample1 {
  public  static void main( String args[] ) {
      int n;     // 정수 변수의 선언
      n=24;    // 변수 n에 24를 저장함.
      System.out.println("This is n= " + n );

      n=n*2;   // n에 48의 값이 저장됨.

      System.out.print("The value of n * 2 is ");
      System.out.println(n);   // 변수 n의 출력

      String s1 = "Hello!";     // 문자열 변수 선언 및 초기화
      String s2 = "Java World";

      System.out.println( s1+"  "+ s2 ); //문자열 변수s1,s2의 연속출력
    }
}
```

Hello You!

– 실행결과

```
C:₩java₩c2>javac  Sample1.java
C:₩java₩c2>java  Sample1


This is n = 24
The value of n * 2 is 48
Hello!  Java World
```

if

if문

프로그램에서 원하는 문장을 선택적으로 실행할 때 사용하는 문장.

자바의 if문은 C나 기타 언어의 if문과 매우 유사.

자바 if문의 형식은 다음과 같음.

형식1) if (condition) statement;

형식2) if (condition) { statement_list; }

여기에서 condition은 참과 거짓의 결과 값을 가지는 부울(boolean)식,

식이 참인 경우에만 문장이 실행됨.

if

– 예제 프로그램 2

```
class Sample2 {
    public  static void main( String args[] ) {
        int x,y;
        x=12;     y=24;
        if( x<y )  // y값이 x보다 크므로 다음 문장 실행
            System.out.println("X is less than y");

        x=x*2;    // now  x=24
        if ( x==y )   // x와 y의 값이 동일하므로 다음 문장 실행
            System.out.println("x now equal to y");

        x=x*2;   // now x=48
        if( x > y )
            System.out.println("x now greater than y");
        else
            System.out.println("x is less than y");
    }
}
```

if

– 실행 결과
C:₩java₩c2>java Sample2

```
x is less than y
x now equal to y
x now greater than y
```

for

- for문

  for문은 하나 이상의 문장을 원하는 횟수만큼 반복 실행하고자 할 때 사용하는 문장. for문의 형식은 C언어와 동일하며 다음과 같음.

  for(initialization; condition; iteration) { statement_list; }

  initialization부분 : 루프 제어 변수를 초기화하는 코드,
  condition부분 : 제어 변수를 테스트하는 부울식, 이 식의 결과 값이 참인 경우에만 실행을 계속하게 됨.
  iteration부분: 루프안의 문장을 실행한 후에 루프 제어 변수를 변경하는 부분.

for

- 다음 코드는 1부터 10까지의 정수 값을 출력하는 for문의 예.

```
int a;
for(a=1; a<=10; a++)
        System.out.println(a);
```

- 다음은 for문을 사용한 예제 프로그램, 1부터 100까지의 정수 중에서 홀수의 합을 구하여 출력하는 자바 프로그램.

```
class ForTest {
   public static void main(String arg[]) {
            int total = 0;
            for(int a=1; a<= 100; a=a+2)
                  total=total+a;
            System.out.println("total = " + total );
   }
}
```

for

- 실행 결과
  C:₩java₩c2>java ForTest
  total=2500

명령행 인자

- 명령행 인자(command-line argument) 받아들이기
  main() 매소드는 문자열 배열을 명령형 인자로 받아들임.

  이 명령행 인자를 사용하면, 사용자가 프로그램 실행 시에 필요한 정보를 전달할 수 있음.

  명령행 인자를 자바 실행 명령어 다음에 명시하면,
  예를 들어 main()메소드의 인자 이름이 args[]이고,
  다음과 같은 실행 명령어를 사용하였다면,

  C:> java java_program 10 20 30.1

명령행 인자

- args[0]에는 문자열 “10”의 주소가, args[1]에는 문자열 “20”의 주소가, args[2]에는 문자열 “30.1”의 주소가 저장됨.

  이를 숫자로 바꾸기 위해서는 다음과 같은 메소드를 사용해야 함.

  Integer.parseInt() 메소드는 문자열 인자를 정수로 전환하고, Float.parseFloat() 메소드는 문자열 인자를 실수로 전환하는 기능을 가짐.

  다음의 명령문

  ```
  int x = Integer.parseInt( args[0] );
  int y = Integer.parseInt( args[1] );
  float z = Float.parseFloat( args[2] );
  ```
  을 실행하면,

  x, y에는 정수 10, 20의 값이 각각 저장되고,
  z에는 실수 30.1의 값이 저장됨.

명령행 인자

– 다음은 명령행 인자를 받아들이는 자바 예제 프로그램.

```
class  ArgInput {
   public  static void main( String args[] ) {
            int a;
            float f;
            a= Integer.parseInt( args[0] );
            f= Float.parseFloat( args[1] );

            System.out.println("Your inputs are "+a+"  "+f);
   }
}
```

– 실행 결과

```
C:\java\c2> java ArgInput 12 24.87
Your inputs are 12 24.87
```

1. Java 프로그래밍의 기초

- 1.1 데이터 타입이란?

한창헌
2002.1.25
ch710@netian.com
http://www.jabook.org

주석

- // comment on one line
- /* comment on one or more lines */
- /** documenting comment */
  - used to generate documentation automatically

C와 동일

데이터 타입

- Java is a *strongly typed language*.
- 8 primitive types
  - number types
    - 4 integer types (`int, short, long, byte`)
    - 2 floating-point types (`float, double`)
    - The sizes are *platform independent*.
  - character type (`char`)
    - 2-byte Unicode encoding
  - boolean type (`boolean`)
    - cannot convert between numbers and Boolean variables, not even with a cast.

unsigned 형과 bit형이 없음

1.1 데이터 타입이란?

데이터 타입

- 자료에 대한 형태를 지정하는 것
- 데이터 타입의 역할
  - 데이터가 가질 자료형을 지정
  - 데이터가 가질 메모리 크기를 지정
  - 변수를 생성
  - 변수와 만나면 그 변수를 메모리에 존재 시킴

데이터 타입의 종류

- 기본 데이터 타입 : JVM이 미리 정해놓은 기본적인 데이터 타입
  - ex) int, long, float, boolean, double, byte, char..

	정 수 형		실 수 형		문 자 형	불 린 형
데이터 타입	int	long	float	double	char	boolean
데이터	0	0L	0.0f	0.0d	'a'	false
byte 수	4	8	4	8	2	1

2. 변수와 상수

- 2.1 변수와 상수

2.1 변수와 상수

변수

- 데이터를 담을 수 있는 그릇
- 일종의 작은 데이터 베이스
- 변수 생성

```
int a;
long b, c;
```

상수

- 미리 정해진 실제값
- 상수 자체가 수이며 이미 만들어져 있는 것
- literal이라 부름

변수의 사용

- Must begin with a letter ('A'-'Z', 'a'-'z', '_', or *any* Unicode characters) and be a sequence of letters or digits.
- Case sensitive
- Length is unlimited.
- Cannot be a keyword.
- Examples:

```
double salary;      int vacationDays;
char yesChar;       boolean done;
long earthPopulation;
```

상수의 사용

- The keyword '***final***' denote a constant.

  e.g., `final double CM_PER_INCH = 2.54;`

- ***Class constants***
  - constants for all the methods in the class
  - use the keywords `static final`
  - definition of a class constant appears *outside* the `main` method

  e.g.,

```
public class UseConstants2
{ public static final double G = 9.81;
  public static void main(String[]args)
  { … }
}
```

3. 할당

- 3.1 할당이란?

3.1 할당이란?

할당

- 변수에 데이터를 넣는 것
- 기본 법칙
  - 오른쪽에서 왼쪽으로만 할당
  - 변수끼리도 할당 가능
  - 변수에 상수가 존재하더라도 다른 것을 할당하면 마지막 할당한 것이 할당
  - “ = ”연산자 사용
- 할당의 예
  - 선언과 할당을 동시에 : int a = 100;
  - 선언과 할당을 분리 : int a; a= 100;
  - 여러 개를 동시에 선언과 할당 : int a = 100, b = 200;
  - 분리하여 여러 개를 동시에 선언과 할당 : int a,   b; a = 100,b=200;
  - 변수끼리의 할당 : int a = 100, b; b = a;

할당의 기본 법칙의 예

```
int sum =100;
sum = sum + 200;
System.out.println(sum);
```

```
300
```

할당의 사용

- No uninitialized variables allowed.
- Declare and initialize a variable on the same line.
- Examples

```
int vacationDays;  // this is a declaration
vacationDays = 12; // this is an assignment
char capitalA = '\u0041'; // decimal 65
```

- *Declaration can be put anywhere in your code, but a variable can be declared only once in any block in a method.*

프리미티브 타입

01. 자바의 타입 분류 체계

자바의 데이터 타입

- 자바 프로그램의 모든 **변수**와 **데이터**에는 타입이 있음

```
class SimpleArith {
    public static void main(String args[]) {
        double result;
        int a = 1 , b = 2 ;
        double c = 0.5 ;
        result = ( a + b ) * c;
        System.out.println( "result = " + result);
    }
}
```

변수에는 모두 타입이 있습니다.

상수에도 모두 타입이 있습니다.

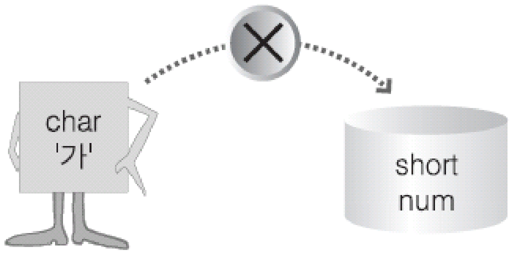
이런 계산의 중간 결과에도 내부적으로 타입이 부여됩니다.

프리미티브 타입

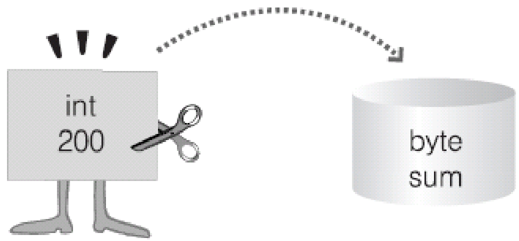
- 01. 자바의 타입 분류 체계

자바의 데이터 타입

- 데이터 타입으로 인해 발생할 수 있는 문제



변수와 다른 타입의 값을
대입하면 에러가 발생할 수 있습니다.

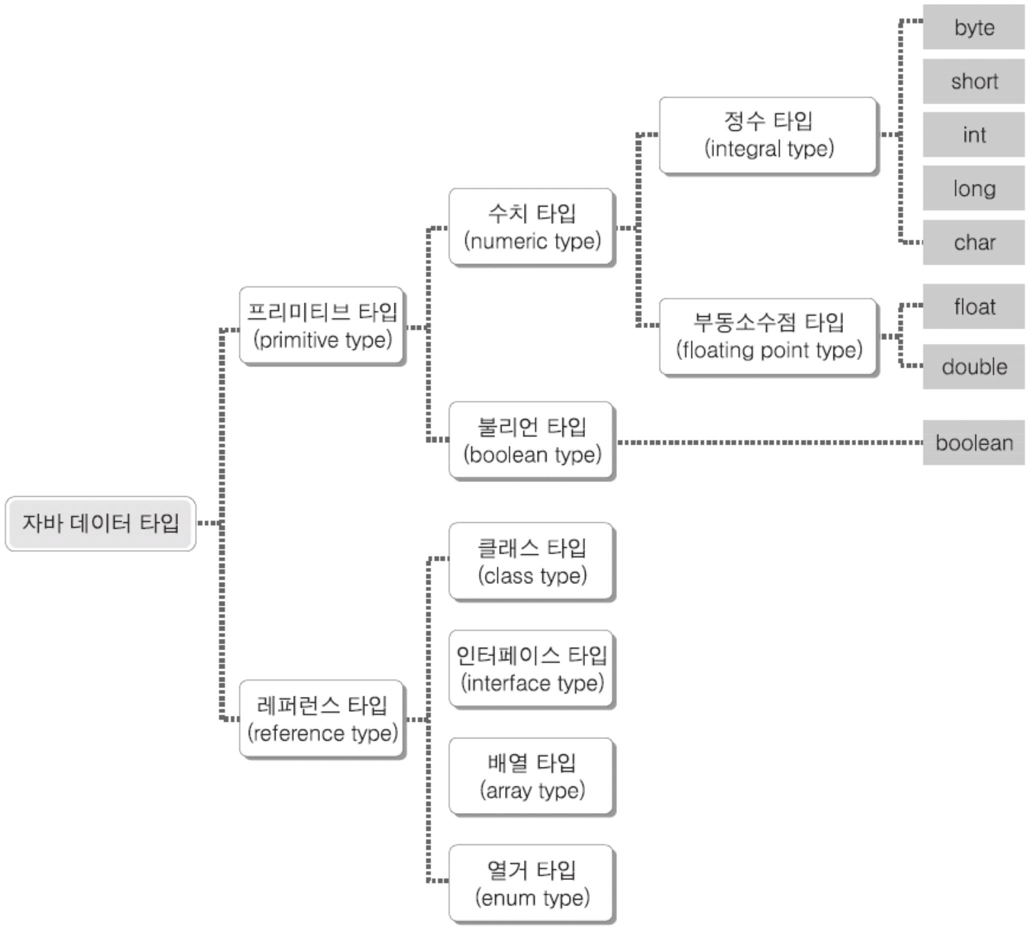


변수와 다른 타입의 값을 담는 도중에
데이터 손실이 일어날 수도 있습니다.

프리미티브 타입

- 01. 자바의 타입 분류 체계

- 자바 데이터 타입의 분류 체계

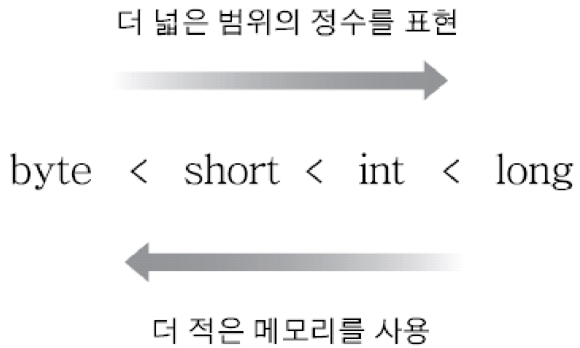


프리미티브 타입

01. 자바의 타입 분류 체계

정수 타입

- byte, short, int, long : 정수를 표현하는 타입



- char : 문자를 표현하는 타입
  - - Unicode 코드 값으로 표현 : 0 ~ 65535 범위의 정수

프리미티브 타입

01. 자바의 타입 분류 체계

부동소수점 타입

- 부동소수점이란?

고정소수점 표기 방법	부동소수점 표기 방법
12.375	0.012375×10^{3}
	0.12375×10^{2}
	1.2375×10^{1}
	12.375×10^{0}
	123.75×10^{-1}
	1237.5×10^{-2}
	$12375. \times 10^{-3}$

모두 12.375의 부동소수점 표기입니다.

프리미티브 타입

01. 자바의 타입 분류 체계

부동소수점 타입

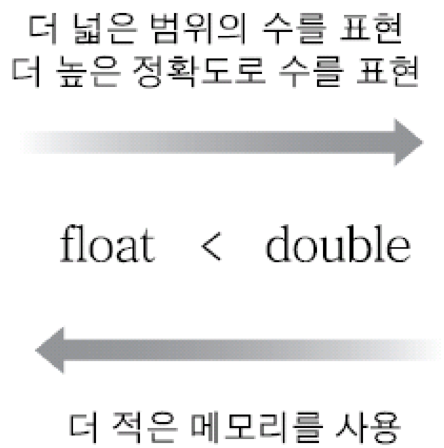
- 자바에서 말하는 부동소수점
  - : 10진법의 부동소수점이 아니라 2진법의 부동소수점
  - [예] $11_2 \times 2^2$ $1.1_2 \times 2^3$ $0.11_2 \times 2^4$

프리미티브 타입

❑ 01. 자바의 타입 분류 체계

❑ 부동소수점 타입

- float, double : 부동소수점 수를 표현하는 타입



프리미티브 타입

01. 자바의 타입 분류 체계

불리언 타입

- boolean : 참과 거짓을 표현하는 타입

```
class SimpleAdder2 {
    public static void main(String args[]) {
        int num;
        num = 10 + 20;
        if ( num > 10 )
            System.out.println("계산 결과가 10보다 큽니다.");
    }
}
```

이런 조건식의 결과는 boolean 타입의 값이 됩니다.

```
class WhileExample2 {
    public static void main(String args[]) {
        while ( true )
            System.out.println("Hello, Java");
    }
}
```

참을 의미하는 true라는 단어는 boolean 타입의 상수입니다.

프리미티브 타입

01. 자바의 타입 분류 체계

불리언 타입

- [예제 3-1] boolean 타입 변수를 선언해서 사용하는 예

```
class BooleanExample1 {
    public static void main(String args[]) {
        int num = 10 + 20;
        boolean truth;               // boolean 타입의 변수를 선언
        truth = num > 10;            // boolean 타입의 변수에 조건식의 결과 대입
        if (truth)                   // boolean 타입의 변수를 사용
            System.out.println("계산 결과가 10보다 큽니다.");
    }
}
```

명령 프롬프트

```
E:\work\chap3\3-1>java BooleanExample1
계산 결과가 10보다 큽니다.

E:\work\chap3\3-1>_
```

프리미티브 타입

- 02. 프리미티브 타입 데이터의 내부 표현

데이터의 내부 표현

- 데이터 표현 방식의 차이
  - 소스 코드에 있는 **상수**와 **입력 데이터** : 사람이 읽기에 적합한 형태
  - 컴파일 후의 데이터 : 컴퓨터가 처리하기에 적합한 형태

- 데이터 표현 방식을 누가 바꾸는가?
  - **상수** : 컴파일러가 컴파일할 때 변환
  - **입력 데이터** : JDK 라이브러리의 메소드를 이용하여 변환 가능

프리미티브 타입

❑ 02. 프리미티브 타입 데이터의 내부 표현

❑ 데이터의 내부 표현

- [예제 3-2] 데이터의 내부 표현을 잘 모르고 작성한 프로그램 (1)

```
class SSimpleAdder {
    public static void main(String args[]) {
        int num;
        num = 1000000000 + 2000000000;
        System.out.println(num);
    }
}
```

명령 프롬프트

```
E:\work\chap3\3-2>java SSimpleAdder
-1294967296

E:\work\chap3\3-2>_
```

프리미티브 타입

- 02. 프리미티브 타입 데이터의 내부 표현

데이터의 내부 표현

- 프리미티브 타입의 데이터 표현을 위해 사용되는 메모리

구분	데이터 타입	사용 메모리 크기
정수 타입	byte	1 바이트
	short	2 바이트
	int	4 바이트
	long	8 바이트
	char	2 바이트
부동소수점 타입	float	4 바이트
	double	8 바이트
불리언 타입	boolean	* 정해져 있지 않음

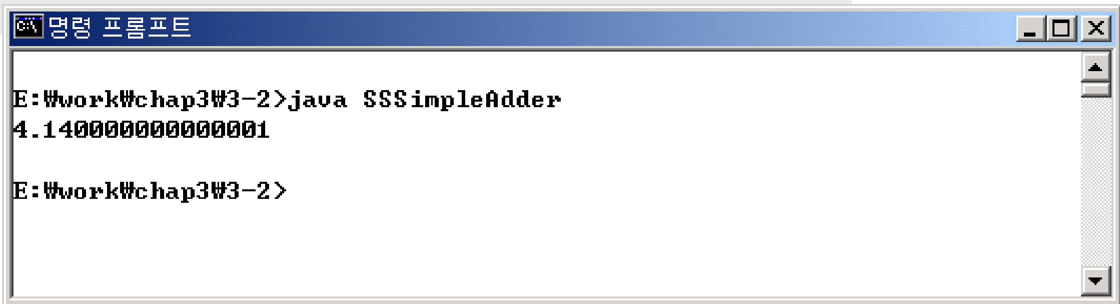
프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

데이터의 내부 표현

- [예제 3-3] 데이터의 내부 표현을 잘 모르고 작성한 프로그램 (2)

```
class SSSimpleAdder {
    public static void main(String args[]) {
        double num;
        num = 3.14 + 1;
        System.out.println(num);
    }
}
```



프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

정수의 내부 표현

- 정수 타입의 표현 가능 범위

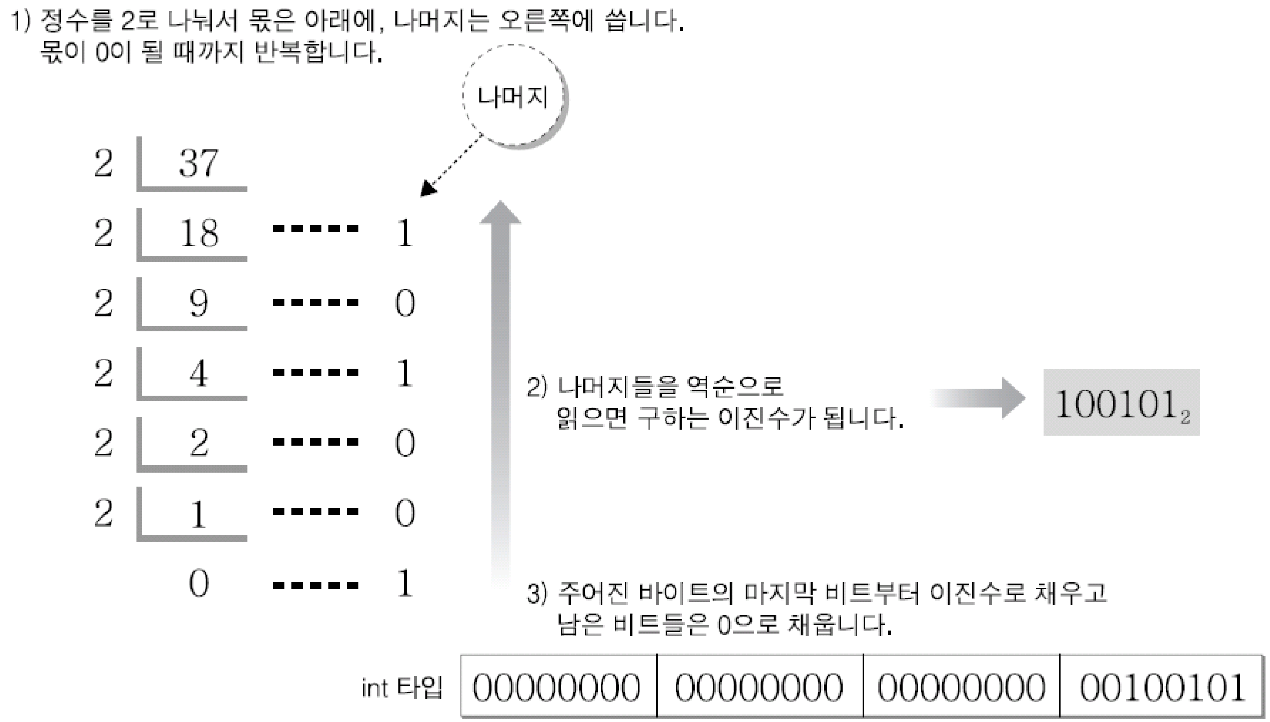
데이터 타입	크기	표현 범위
byte	1 바이트	-128 ~ 127
short	2 바이트	-32768 ~ 32767
int	4 바이트	-2147483648 ~ 2147483647
long	8 바이트	-9223372036854775808 ~ 9223372036854775807

프리미티브 타입

❑ 02. 프리미티브 타입 데이터의 내부 표현

❑ 정수의 내부 표현

- • 영(零, 0)과 양의 정수는 2진수를 그대로 표현



프리미티브 타입

- 02. 프리미티브 타입 데이터의 내부 표현

- 정수의 내부 표현
  - 부호
    - – 첫번째 비트(MSB: Most Significant Bit)를 가지고 표현
    - – +는 0, –는 1
  - 음의 정수
    - 해당 양의 정수의 2의 보수로 표현
    - [예] –37은 37의 2의 보수로 표현

프리미티브 타입

- 02. 프리미티브 타입 데이터의 내부 표현

- 정수의 내부 표현
  - 2의 보수를 만드는 방법
    - 1) 2진수의 각 자리의 숫자를 1은 0으로, 0은 1로 바꾸어서 새로운 2진수를 만듭니다.
    - 2) 1)의 결과에 1을 더합니다.

프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

정수의 내부 표현

- • 음의 정수를 2의 보수로 표현하는 방법 : −37의 예

1) 각 자리의 숫자를 1은 0으로, 0은 1로 바꿉니다.

37의 int 타입 표현 →

00000000	00000000	00000000	00100101

↓

11111111	11111111	11111111	11011010

2) 1)의 결과에 1을 더합니다

```
  11111111111111111111111111011010
+                                1
----------------------------------
  11111111111111111111111111011011
```

−37의 int 타입 표현 ←

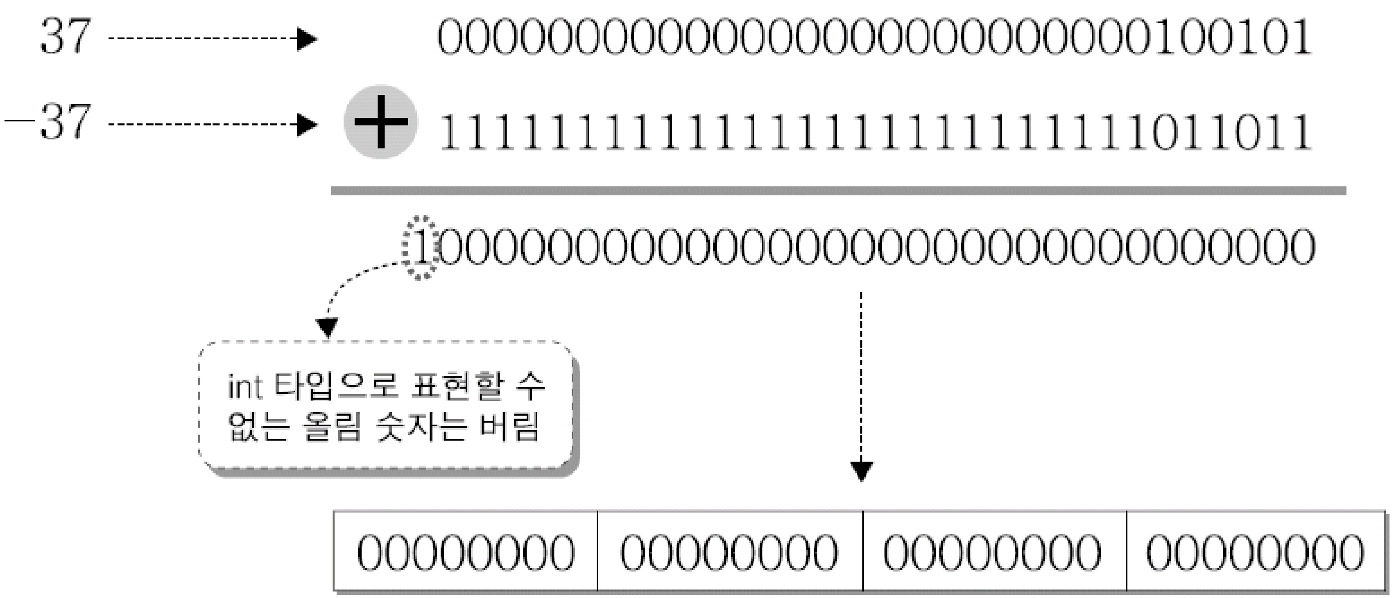
11111111	11111111	11111111	11011011

프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

정수의 내부 표현

- • 37과 2의 보수로 표현된 –37의 덧셈

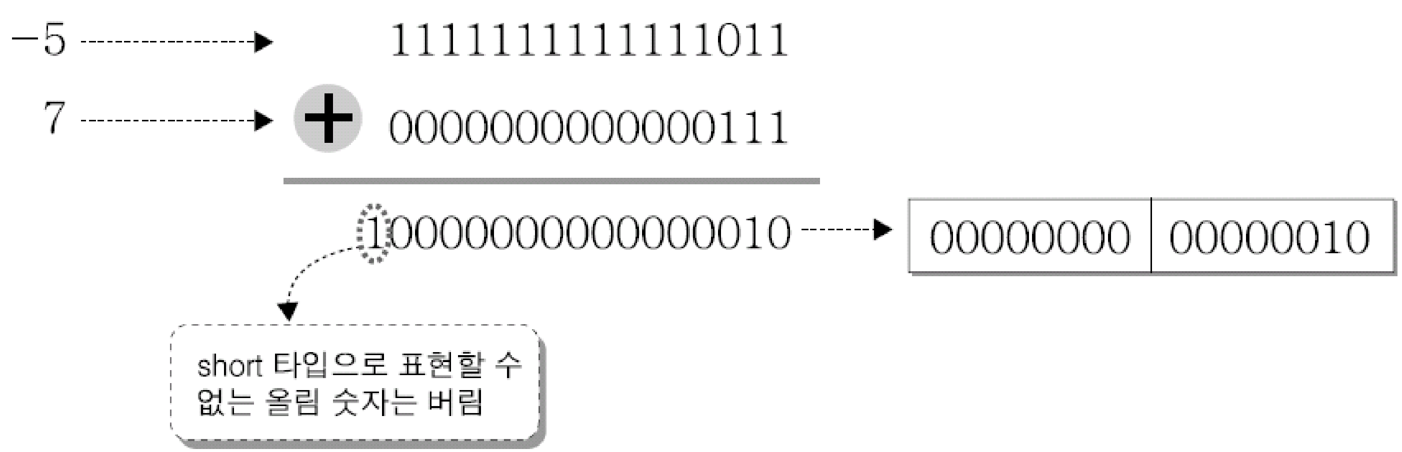


프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

정수의 내부 표현

- 2의 보수로 표현된 –5와 7의 덧셈



프리미티브 타입

❑ 02. 프리미티브 타입 데이터의 내부 표현

❑ 정수의 내부 표현

- 1바이트를 가지고 표현할 수 있는 정수의 범위

2진수로 표현되는 0과 양의 정수		2의 보수로 표현되는 음의 정수	
0	00000000		
1	00000001	11111111	−1
2	00000010	11111110	−2
⋮		⋮	
125	01111101	10000011	−125
126	01111110	10000010	−126
127	01111111	10000001	−127
		10000000	?

프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

정수의 내부 표현

- 비트 패턴 10000000에 1을 더하면 나오는 값

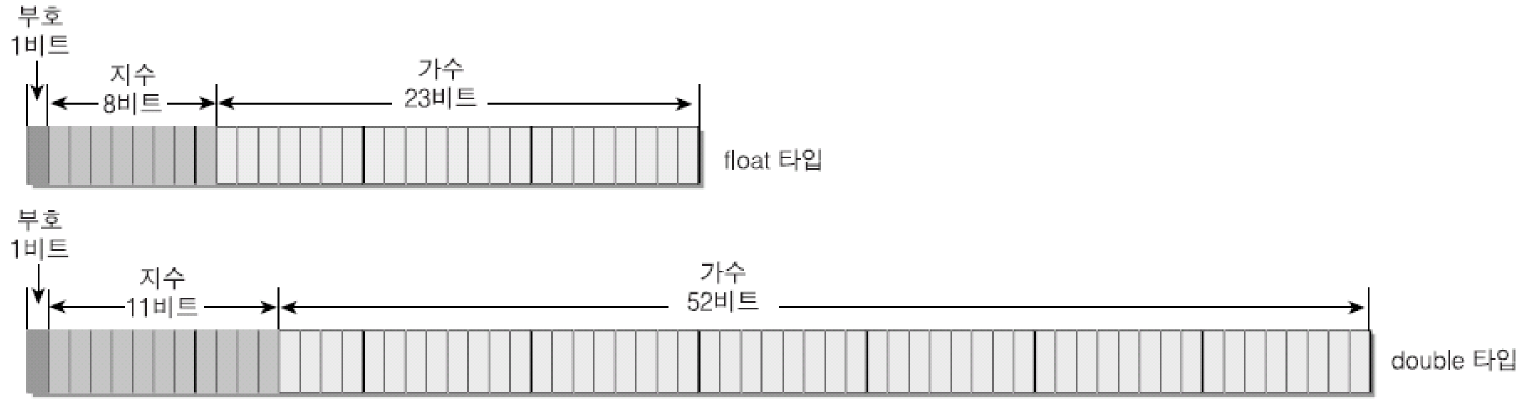
```
  10000000
+        1
----------
  10000001 -------------> -127에 해당
```

프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

부동소수점수의 내부 표현

- IEEE 754 표준 규약에 따른 표현

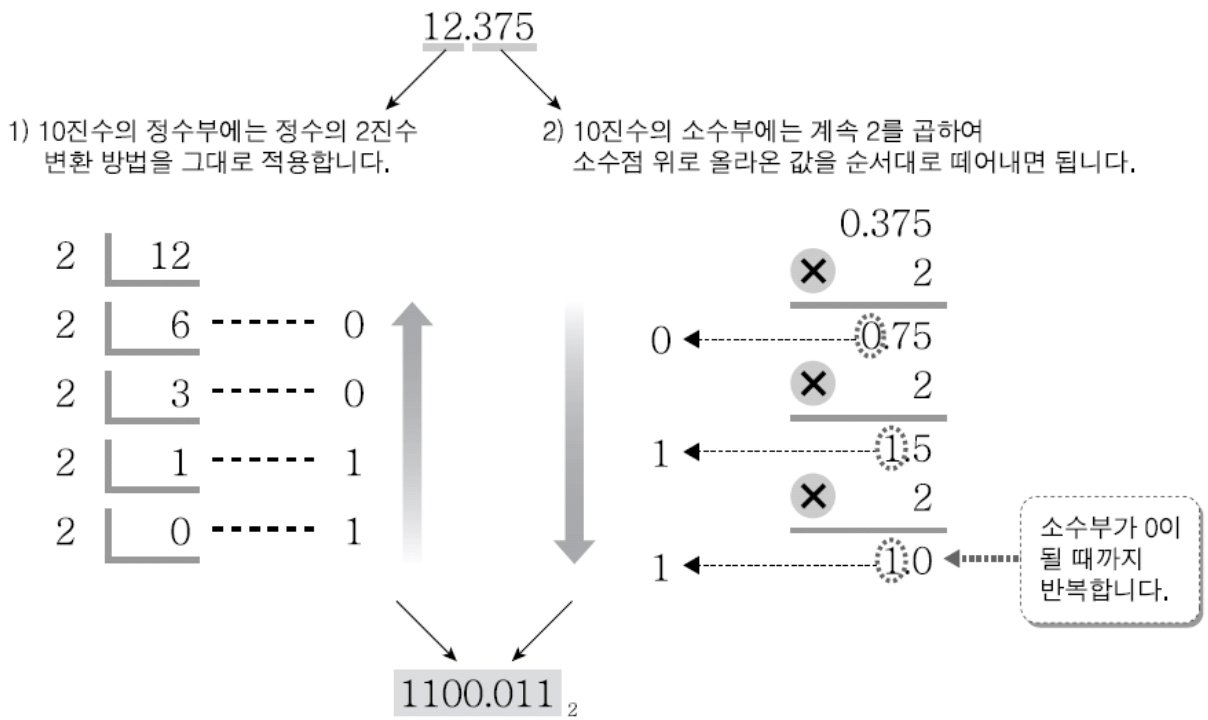


프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

부동소수점수의 내부 표현

- 10진 소수를 2진 소수로 바꾸는 방법



프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

부동소수점수의 내부 표현

- 2진수의 고정소수점과 부동소수점

고정소수점 표기 방법

1100.011_2

선택되는 표기

부동소수점 표기 방법

$0.1100011_2 \times 2^4$

$1.100011_2 \times 2^3$

$11.00011_2 \times 2^2$

$110.0011_2 \times 2^1$

$1100.011_2 \times 2^0$

$11000.11_2 \times 2^{-1}$

$110001.1_2 \times 2^{-2}$

$1100011_2 \times 2^{-3}$

$11000110_2 \times 2^{-4}$

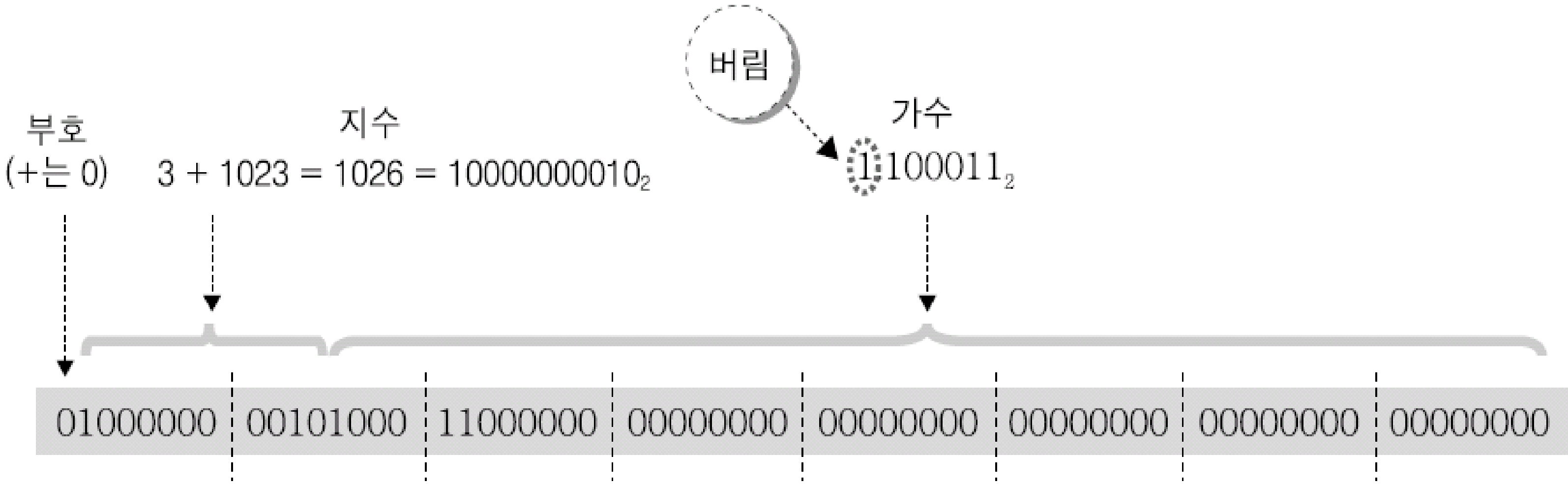
모두 1100.011_2의
부동소수점 표기입니다.

이 중 어떤 표기를 택해야할까요?

프리미티브 타입

- 02. 프리미티브 타입 데이터의 내부 표현

- 부동소수점수의 내부 표현
  - • $1.100011_2 \times 2^3$을 double 타입으로 표현한 예



프리미티브 타입

- 02. 프리미티브 타입 데이터의 내부 표현

부동소수점수의 내부 표현

- 부동소수점 타입이 표현할 수 있는 값의 범위

a) float 타입으로 표현할 수 있는 가장 큰 수와 가장 작은 수

01111111	01111111	11111111	11111111	→ $+\ 3.4028235 \times 10^{38}$
11111111	01111111	11111111	11111111	→ $-\ 3.4028235 \times 10^{38}$

b) double 타입으로 표현할 수 있는 가장 큰 수와 가장 작은 수

01111111	11101111	11111111	11111111	11111111	11111111	11111111	11111111

→ $+\ 1.7976931348623157 \times 10^{308}$

11111111	11101111	11111111	11111111	11111111	11111111	11111111	11111111

→ $-\ 1.7976931348623157 \times 10^{308}$

프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

부동소수점수의 내부 표현

- 부동소수점 타입의 표현 범위를 넘어선다는 의미의 비트 패턴

a) float 타입의 범위를 넘어선 값의 표현

01111111	10000000	00000000	00000000	→ float 타입의 범위보다 큰 수라는 의미
11111111	10000000	00000000	00000000	→ float 타입의 범위보다 작은 수라는 의미

b) double 타입의 범위를 넘어섰다는 표현

01111111	11110000	00000000	00000000	00000000	00000000	00000000	00000000

→ double 타입의 범위보다 큰 수라는 의미

11111111	11110000	00000000	00000000	00000000	00000000	00000000	00000000

→ double 타입의 범위보다 작은 수라는 의미

프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

부동소수점수의 내부 표현

- 부동소수점 타입으로 표현할 수 있는 가장 미세한 수

a) float 타입으로 표현할 수 있는 가장 미세한 값

00000000	00000000	00000000	00000001	→ $+1.40 \times 10^{-45}$
10000000	00000000	00000000	00000001	→ -1.40×10^{-45}

b) double 타입으로 표현할 수 있는 가장 미세한 값

00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000001

→ $+4.9 \times 10^{-324}$

10000000	00000000	00000000	00000000	00000000	00000000	00000000	00000001

→ -4.9×10^{-324}

❑ 02. 프리미티브 타입 데이터의 내부 표현

❑ 부동소수점수의 내부 표현

- [예제 3-4] double 타입의 가장 미세한 값을 2로 나누는 프로그램

```
class TypeExample1 {
    public static void main(String args[]) {
        double smallest1 = 4.9e-324;          // double 타입의 가장 미세한 +값
        double smallest2 = -4.9e-324;         // double 타입의 가장 미세한 -값
        double result1, result2;
        result1 = smallest1 / 2.0;            // 가장 미세한 +값을 2로 나누는 식
        result2 = smallest2 / 2.0;            // 가장 미세한 -값을 2로 나누는 식
        System.out.println("result1 = " + result1);
        System.out.println("result2 = " + result2);
    }
}
```

명령 프롬프트

```
E:₩work₩chap3₩3-2-2>java TypeExample1
result1 = 0.0
result2 = -0.0

E:₩work₩chap3₩3-2-2>
```

프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

부동소수점수의 내부 표현

- 부동소수점 타입의 +0과 –0의 표현

a) float 타입의 0을 표현하는 값

00000000	00000000	00000000	00000000	→ + 0.0
10000000	00000000	00000000	00000000	→ − 0.0

b) double 타입의 0을 표현하는 값

00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

→ + 0.0

10000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

→ − 0.0

02. 프리미티브 타입 데이터의 내부 표현

부동소수점수의 내부 표현

- [예제 3-5] 부동소수점 수를 0으로 나누는 프로그램

```
class TypeExample2 {
    public static void main(String args[]) {
        double result1, result2;
        result1 = 2.0 / 0.0;
        result2 = 2.0 / -0.0;
        System.out.println("result1 = " + result1);
        System.out.println("result2 = " + result2);
    }
}
```

명령 프롬프트

```
E:\work\chap3\3-2-2>java TypeExample2
result1 = Infinity
result2 = -Infinity

E:\work\chap3\3-2-2>
```

프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

부동소수점수의 내부 표현

- 부동소수점 타입의 NaN(Not a Number) 표현

a) float 타입의 NaN 표현

01111111	11000000	00000000	00000000

b) double 타입의 NaN 표현

01111111	11111000	00000000	00000000	00000000	00000000	00000000	00000000

프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

부동소수점수의 내부 표현

- JDK 라이브러리에 있는 부동소수점 타입 관련 상수들

의미	상수 이름	
	float 타입 상수	double 타입 상수
표현 가능한 최대값, 최소값의 절대치	Float.MAX_VALUE	Double.MAX_VALUE
표현 가능한 가장 미세한 값의 절대치	Float.MIN_VALUE	Double.MIN_VALUE
표현 범위를 넘어서는 양의 값	Float.POSITIVE_INFINITY	Double.POSITIVE_INFINITY
표현 범위를 넘어서는 음의 값	Float.NEGATIVE_INFINITY	Double.NEGATIVE_INFINITY
NaN (Not a Number)	Float.NaN	Double.NaN

프리미티브 타입

02. 프리미티브 타입 데이터의 내부 표현

문자의 내부 표현 : Unicode

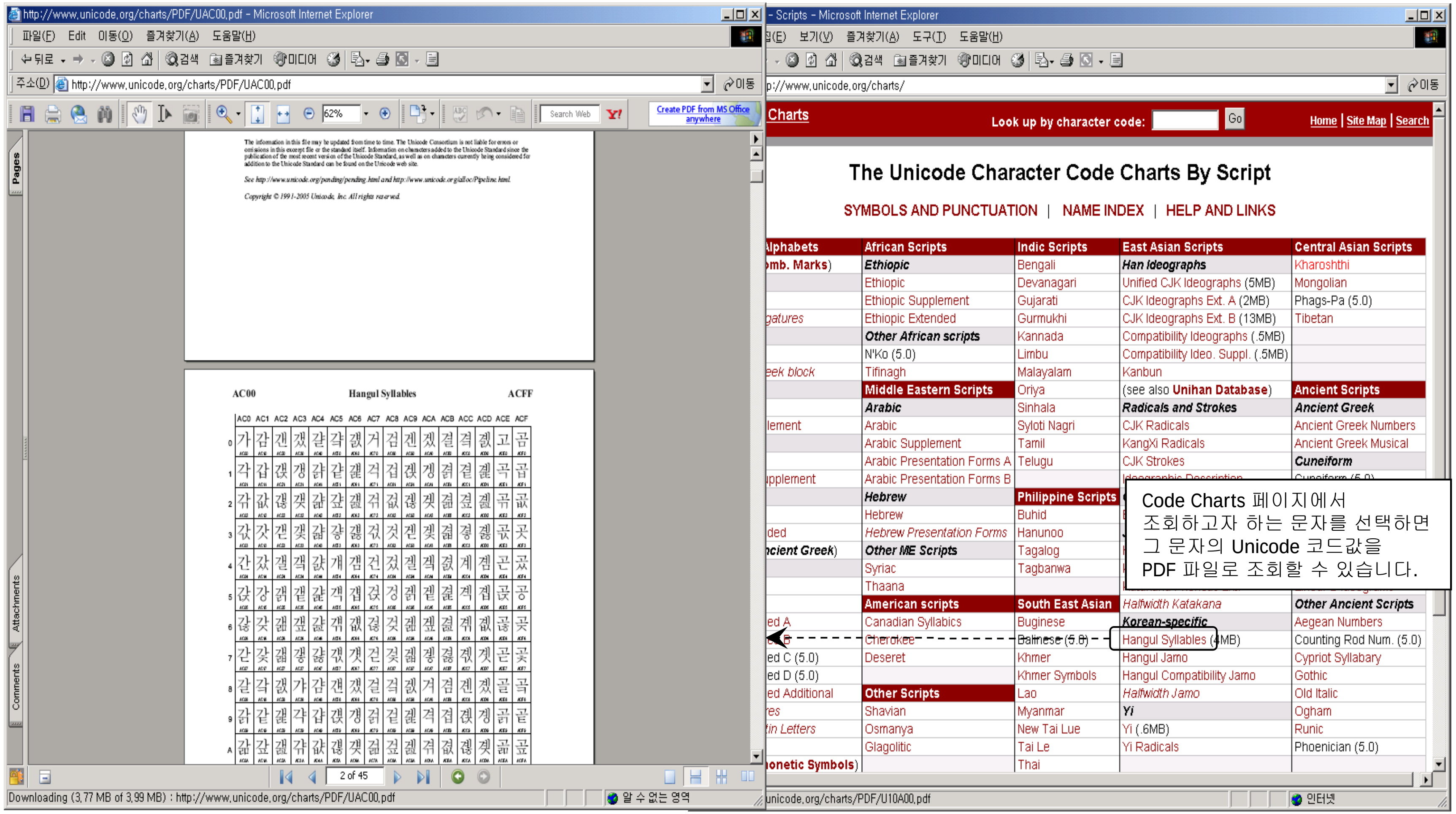
- char 타입의 내부 표현

Unicode	char 타입 내부 표현	Unicode	char 타입 내부 표현
뇌 B1CC	10110001 11001100	J 004A	00000000 01001010
를 B97C	10111001 01111100	a 0061	00000000 01100001
자 C790	11000111 10010000	v 0076	00000000 01110110
극 ADF9	10101101 11111001	5 0035	00000000 00110101
하 D558	11010101 01011000	. 002E	00000000 00101110
는 B294	10110010 10010100	0 0030	00000000 00110000

프리미티브 타입

- 02. 프리미티브 타입 데이터의 내부 표현

- 문자의 내부 표현: Unicode



프리미티브 타입

03. 리터럴의 표기 방법

리터럴

- 리터럴(literal) : 소스 코드 내에 데이터 값 그대로 쓴 상수
  - 리터럴에도 타입이 있습니다.

```
class AnotherSimpleAdder {
    public static void main(String args[]) {
        int num = 1 ;
        double sum = num + 0.5 ;
        System.out.println( "sum = " + sum);
        System.out.println('끝');
    }
}
```

소수점이 없는 수치 리터럴은기본적으로 int 타입입니다.

소수점이 있는 수치 리터럴은 기본적으로 double 타입입니다.

작은따옴표로 묶은 하나의 문자는 char 타입입니다.

큰따옴표로 묶은 문자열은 String 타입입니다.

프리미티브 타입

03. 리터럴의 표기 방법

리터럴의 타입

- [예제 3-6] 리터럴에도 타입이 있다는 사실을 잘 모르고 작성한 프로그램

```java
class LiteralExample1 {
    public static void main(String args[]) {
        float num;
        num = 12.34;
        System.out.println(num);
    }
}
```

명령 프롬프트

```
E:₩work₩chap3₩3-3>javac LiteralExample1.java
LiteralExample1.java:4: possible loss of precision
found    : double
required: float
        num = 12.34;
              ^
1 error

E:₩work₩chap3₩3-3>_
```

프리미티브 타입

03. 리터럴의 표기 방법

정수 리터럴의 표기 방법

- int 타입 리터럴의 예 (1)

120

아라비아 숫자로만 구성된
정수 리터럴은 int 타입입니다.

프리미티브 타입

03. 리터럴의 표기 방법

정수 리터럴의 표기 방법

- int 타입 리터럴의 예 (2)

024

0으로 시작하는 정수 리터럴은
8진수로 취급됩니다.

프리미티브 타입

03. 리터럴의 표기 방법

정수 리터럴의 표기 방법

- int 타입 리터럴의 예 (3)

0x30A1

0x 또는 0X로 시작하는
정수 리터럴은
16진수로 취급됩니다.

10부터 15까지의 숫자는
A,B,C,D,E 또는 a,b,c,d,e로
표시해야 합니다.

프리미티브 타입

03. 리터럴의 표기 방법

정수 리터럴의 표기 방법

- [예제 3-7] int 타입 리터럴의 사용 예

```
class LiteralExample2 {
    public static void main(String args[]) {
        System.out.println(120);
        System.out.println(024);
        System.out.println(0x30A1);
        System.out.println(0x0030a1);
    }
}
```

명령 프롬프트

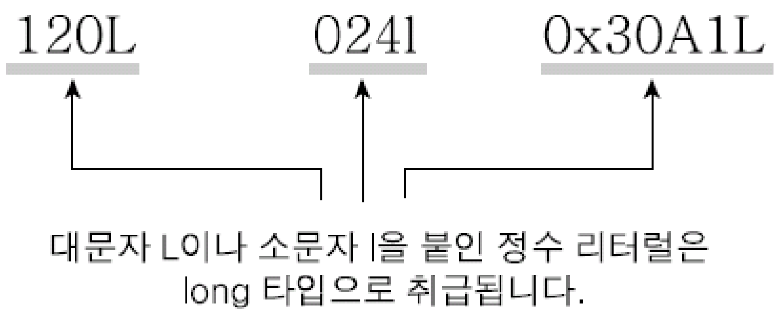
```
E:₩work₩chap3₩3-3-1>java LiteralExample2
120
20
12449
12449

E:₩work₩chap3₩3-3-1>_
```

프리미티브 타입

03. 리터럴의 표기 방법

정수 리터럴의 표기 방법

- long 타입 리터럴의 예 (1)



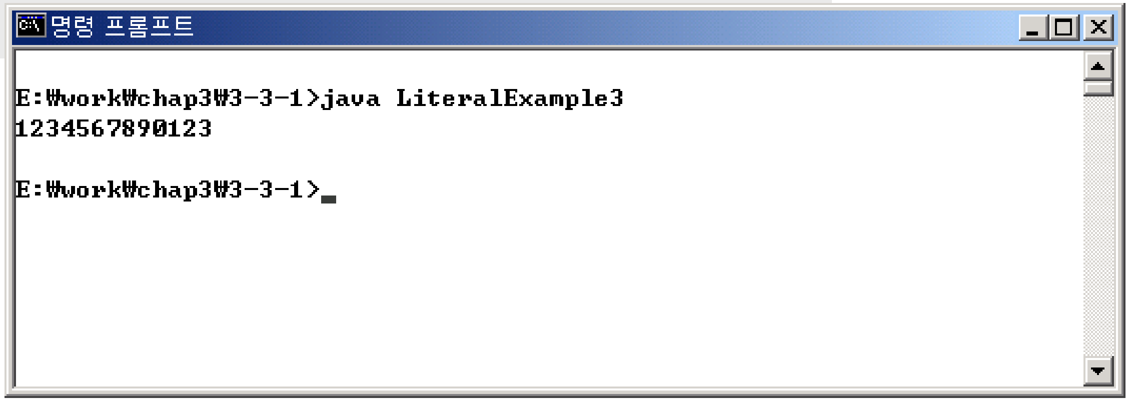
프리미티브 타입

03. 리터럴의 표기 방법

정수 리터럴의 표기 방법

- [예제 3-8] long 타입 리터럴의 사용 예

```
class LiteralExample3 {
    public static void main(String args[]) {
        long num = 1234567890123L;
        System.out.println(num);
    }
}
```



프리미티브 타입

- 03. 리터럴의 표기 방법

- 부동소수점 리터럴의 표기 방법
  - double 타입 리터럴의 예 (1)

12.025

소수점과 아라비아 숫자로 구성된 리터럴은
double 타입으로 취급됩니다.

프리미티브 타입

03. 리터럴의 표기 방법

부동소수점 리터럴의 표기 방법

- double 타입 리터럴의 예 (2)

12.

이렇게 쓴 리터럴은
12.0과 동일합니다.

.025

이렇게 쓴 리터럴은
0.025와 동일합니다.

프리미티브 타입

03. 리터럴의 표기 방법

부동소수점 리터럴의 표기 방법

- double 타입 리터럴의 예 (3)

12e100

12×10^{100}을 표현하는 부동소수점 리터럴입니다.

0.25E−20

0.25×10^{-20}을 표현하는 부동소수점 리터럴입니다.

프리미티브 타입

- 03. 리터럴의 표기 방법

부동소수점 리터럴의 표기 방법

- double 타입 리터럴의 예 (4)

0xA1.27p5

0xA1.27 x 16^5 값을 표현하는
부동소수점 리터럴입니다.

프리미티브 타입

❑ 03. 리터럴의 표기 방법

❑ 부동소수점 리터럴의 표기 방법

- [예제 3-9] double 타입 리터럴의 사용 예

```
class LiteralExample4 {
    public static void main(String args[]) {
        System.out.println(12.025);
        System.out.println(12e3);
        System.out.println(12e-3);
        System.out.println(0xA1.27p5);
    }
}
```

명령 프롬프트

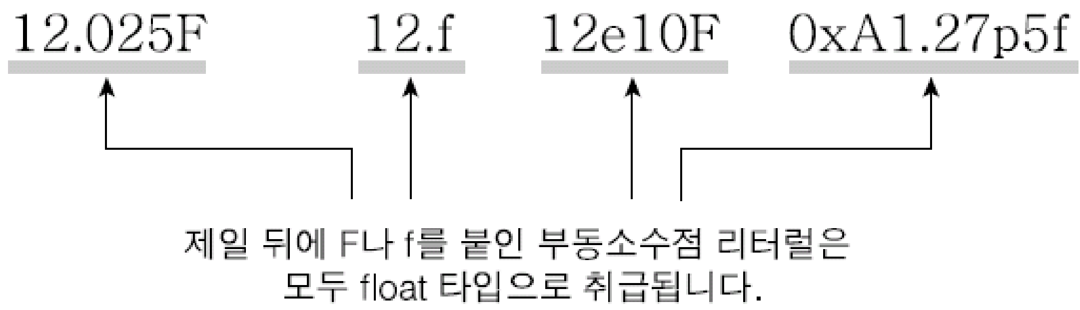
```
E:₩work₩chap3₩3-3-2>java LiteralExample4
12.025
12000.0
0.012
5156.875

E:₩work₩chap3₩3-3-2>
```

프리미티브 타입

03. 리터럴의 표기 방법

부동소수점 리터럴의 표기 방법

- float 타입 리터럴의 예 (1)



프리미티브 타입

03. 리터럴의 표기 방법

부동소수점 리터럴의 표기 방법

- float 타입 리터럴의 예 (2)

12F

뒤에 F나 f가 붙은 10진수는
소수점이 없어도 float 타입입니다.

프리미티브 타입

03. 리터럴의 표기 방법

부동소수점 리터럴의 표기 방법

- [예제 3-10] float 타입 리터럴의 사용 예

```java
class LiteralExample5 {
    public static void main(String args[]) {
        float num = 12.34f;
        System.out.println(num);
    }
}
```

명령 프롬프트

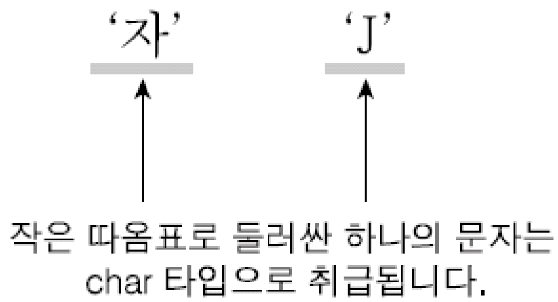
```
E:\work\chap3\3-3-2>java LiteralExample5
12.34

E:\work\chap3\3-3-2>
```

프리미티브 타입

03. 리터럴의 표기 방법

문자 리터럴의 표기 방법

- char 타입 리터럴의 예 (1)

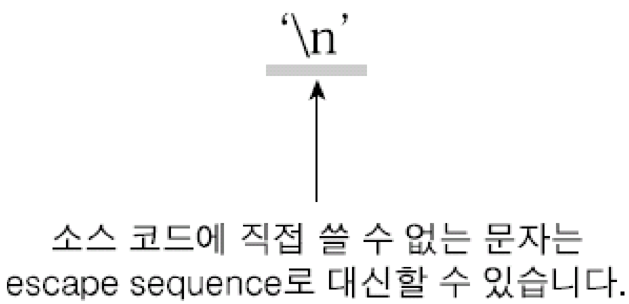


프리미티브 타입

03. 리터럴의 표기 방법

문자 리터럴의 표기 방법

- char 타입 리터럴의 예 (2)



03. 리터럴의 표기 방법

문자 리터럴의 표기 방법

- 자바 프로그램에서 사용할 수 있는 escape sequence

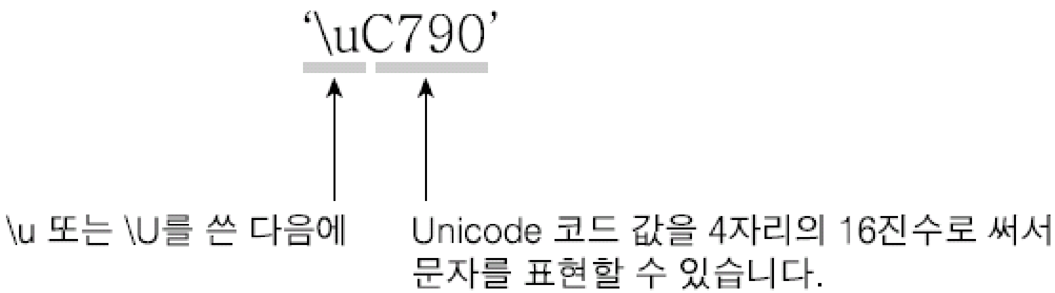
Escape Sequence	의미	Unicode
₩b	백스페이스 (backspace BS)	0x0008
₩t	수평 탭 (horizontal tab HT)	0x0009
₩n	줄 바꿈 문자 (line feed LF)	0x000a
₩f	새 페이지 문자 (form feed FF)	0x000c
₩r	리턴 문자 (carriage return CR)	0x000d
₩”	큰따옴표 (double quote “)	0x0022
₩’	작은따옴표 (single quote ‘)	0x0027
₩₩	백슬래쉬 (backslash ₩)	0x005c
₩8진수	8진수에 해당하는 Unicode 문자. 예) ₩8, ₩42, ₩377	0x0000 ~ 0x00ff

프리미티브 타입

03. 리터럴의 표기 방법

문자 리터럴의 표기 방법

- char 타입 리터럴의 예 (3)



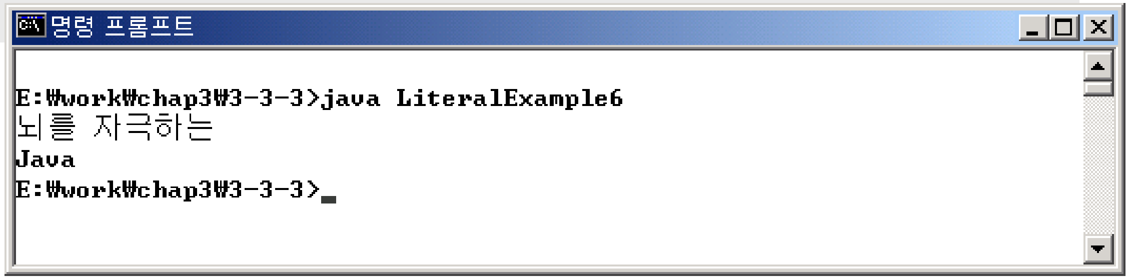
프리미티브 타입

03. 리터럴의 표기 방법

문자 리터럴의 표기 방법

- [예제 3-11] char 타입 리터럴의 사용 예

```
class LiteralExample6 {
    public static void main(String args[]) {
      char arr[] = { '뇌', '를', ' ', '\uC790', '\uADF9', '하',
                     '는', '\n', 'J', 'a', '\166', '\141' };
      for (char ch : arr)
            System.out.print(ch);
    }
}
```



프리미티브 타입

03. 리터럴의 표기 방법

문자열 리터럴의 표기 방법

- String 타입 리터럴의 예 (1)

"Hello, Java"

큰 따옴표로 둘러싼 텍스트는
String 타입으로 취급됩니다.

프리미티브 타입

03. 리터럴의 표기 방법

문자열 리터럴의 표기 방법

- String 타입 리터럴의 예 (2)

“ ”

문자를 하나도 포함하지 않는
문자열 리터럴도 있습니다.

프리미티브 타입

03. 리터럴의 표기 방법

문자열 리터럴의 표기 방법

- [예제 3-12] escape sequence를 포함하는 문자열 리터럴의 예

```
class LiteralExample7 {
    public static void main(String args[]) {
        String str1 = "Hello, Java.₩nNice to meet you.";
        String str2 = "나는 ₩"비틀즈는 1970년에 해체됐어요₩"라고 말했다";
        System.out.println(str1);
        System.out.println(str2);
    }
}
```

명령 프롬프트

```
E:₩work₩chap3₩3-3-4>java LiteralExample7
Hello, Java.
Nice to meet you.
나는 "비틀즈는 1970년에 해체됐어요"라고 말했다

E:₩work₩chap3₩3-3-4>
```

프리미티브 타입

03. 리터럴의 표기 방법

문자열 리터럴의 표기 방법

- [예제 3-13] 아주 긴 문자열 리터럴을 +로 연결해서 표시한 예

```
class LiteralExample8 {
    public static void main(String args[]) {
        String str = "살어리 살어리랏다 청산에 살어리랏다 " +
                     "머루랑 다래랑 먹고 청산에 살어리랏다 " +
                     "얄리얄리 얄라셩 얄라리 얄라";
        System.out.println(str);
    }
}
```

명령 프롬프트

```
E:\work\chap3\3-3-4>java LiteralExample8
살어리 살어리랏다 청산에 살어리랏다 머루랑 다래랑 먹고 청산에 살어리랏
다 얄리얄리 얄라셩 얄라리 얄라

E:\work\chap3\3-3-4>
```

프리미티브 타입

- 03. 리터럴의 표기 방법

불리언 리터럴의 표기 방법

- boolean 타입 리터럴

true **false**

이 두 가지가 전부임

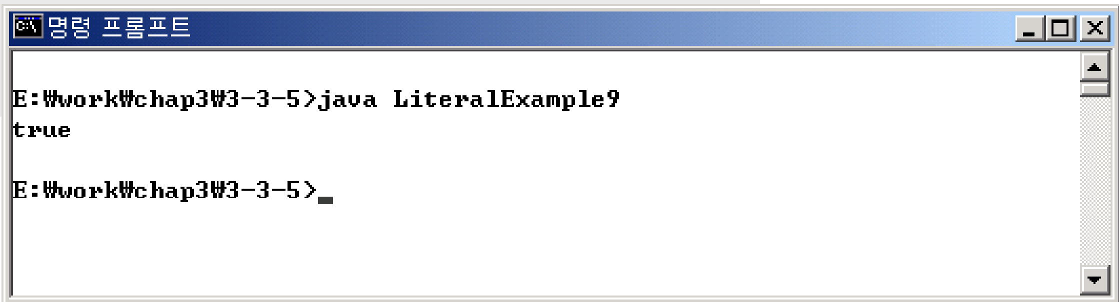
프리미티브 타입

03. 리터럴의 표기 방법

불리언 리터럴의 표기 방법

- [예제 3-14] 불리언 리터럴의 사용 예

```
class LiteralExample9 {
    public static void main(String args[]) {
        int num = 10000;
        boolean isBig;
        if (num > 100)
            isBig = true;
        else
            isBig = false;
        System.out.println(isBig);
    }
}
```



4. 연산자

- 3.1 할당이란?

연산자

- Arithmetic operators
  - can be used in variable initializations:

```
int n = 5;
int a = 2 * n; // a is 10
```

- Exponentiation
  - no operator for raising a quantity to a power
  - use `pow` function (e.g., `y = Math.pow(x, a);`)
- Increment and Decrement Operators
  - same as in C and C++

관계 및 블린 연산자

- Uses different symbols for assignment (=) and equality (==).
- Bitwise operators (&, |, ^, ~, >>, <<, >>>)
  - ^ is the “xor” operator and not the power operator.
  - >>> (logical shift) vs >> (arithmetic shift)
- Logical operators (&&, ||)

비트 연산자

- You can work directly with the bits that make up the integers.
  - &("and") |("or")   ^("xor")   ~("not")

    e.g., int fourthBitFromRight = (n & 8) / 8;

    (if n=10, then fourthBitFromRight = 8)
  - >>("right shift") <<("left shift")

    e.g., int fourthBitFromRight = (n & (1 << 3)) >> 3;

    (if n=10, then fourthBitFromRight = 1)
  - >>> operator fills the top bits with zero.

수학 함수

- The `Math` class contains an assortment of mathematical functions that you may occasionally need.
  - `Math` class supplies `sqrt`, `pow` method
  - `Math` class supplies the usual trigonometric functions(`Math.sin`, `Math.cos`, …) and the exponential function and its inverse, the natural log(`Math.exp`, `Math.log`)
  - Math class supplies two constants:

    `Math.PI`, `Math.E`

수치 데이터 타입간의 변환

- Any binary operations on numeric values of different types will be acceptable (*without loss of information*).
- Conversions where loss of information is possible are done by means of *casts*

  e.g., `double x = 9.997; int nx = (int) x;`
- Certain assignment conversions are allowed by assigning the value of a variable of one type to another without an explicit cast

  `byte → short → int → long → float → double`

수치변환

- 클래스를 이용한 방법
  - Integer
    - http://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html
  - Long
    - http://docs.oracle.com/javase/7/docs/api/java/lang/Long.html
  - Float
    - http://docs.oracle.com/javase/7/docs/api/java/lang/Float.html
  - Double
    - http://docs.oracle.com/javase/7/docs/api/java/lang/Double.html
  - BigDecimal
    - http://docs.oracle.com/javase/7/docs/api/java/math/BigDecimal.html

String 연산자

- 02. 여러 가지 연산자들

- 문자열 연결 연산자
  - 두 번째 +는 덧셈 연산자일까요? 문자열 연결 연산자일까요?

“num=” + 3 + 4

String(1)

- No built-in string type, but
- Predefined class called "*String*"
- Concatenation ("+")
  - e.g., `String message = "Expletive" + "deleted";`
    `String rating = "PG" + 13;`
  - Every Java object can be converted to a string
- Substrings
  - e.g., `String s = greeting.substring(0, 4);`
- String editing
  - e.g., `int n = greeting.length()`
    `s.charAt(n), where 0 ≤ n ≤ s.length()-1`

연산자

- 02. 여러 가지 연산자들

- 문자열 연결 연산자
  - 다음 연산식의 결과는 무엇일까요?

3 + 4 + “=num”

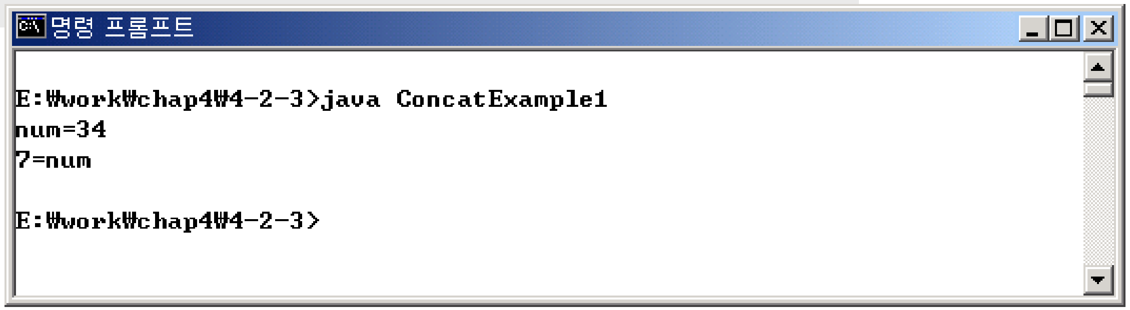
연산자

02. 여러 가지 연산자들

문자열 연결 연산자

- [예제 4-4] +의 처리 순서를 테스트하는 프로그램

```java
class ConcatExample1 {
    public static void main(String args[]) {
        String str1 = "num=" + 3 + 4;
        String str2 = 3 + 4 + "=num";
        System.out.println(str1);
        System.out.println(str2);
    }
}
```



String(2)

- Cannot change a character in the string.
  - i.e., the objects of the `String` class is *immutable*.
  - Why? Strings can be *shared* !
- To modify the string,
  - take the substring that you want to keep and concatenate it with the characters that you want to replace. (Or, use `StringBuffer` class.)
  - e.g.,

```
String greeting = "Hello";
greeting = greeting.substring(0,3) + "p!";
```

Java.lang.String

- String substring(int beginIndex, int endIndex)
  e.g., String s = greeting.substring(0, 4);
- int length()
  e.g., int n = greeting.length();
- char chatAt(int index)
  e.g., s.charAt(n), where $0 \leq n \leq$ s.length()-1
- String toLowerCase()
- String toUpperCase()
- boolean equals(Object other)
- int indexOf(String str)
- 기타 50개 이상의 메소드를 포함하고 있음.

String의 사용

- To test whether or not two strings are equal, use the `equals` method

  e.g., `s.equals(t);`

  `“Hello”.equalsIgnoreCase(“hello”);`

- Do not use the == operator to compare strings.
  - it only determines whether or not the strings are stored in the same location.

String 사용의 예

```
public class STRToStringTest {
  public static void main(String args[]) {
    String s1 = "lower string";
    String s2 =s1;
    String s3= new String("lower string");
    if(s1 == s2) System.out.println("true");
    else System.out.println("false");
    if(s1.equals(s2))
          System.out.println("true");
    else System.out.println("false");
    if(s1 == s3)System.out.println("true");
    else System.out.println("false");
    if(s1.equals(s3))
          System.out.println("true");
    else System.out.println("false");
  }
}
```

* Output *

true
true
false
true

입력과 출력

표준출력 스트림

- System.out.println
  - System.out.println(“Hello You!”);
- System.out.printf
  - double x = 10000.0/3.0;
  - System.out.printf(“%i.2f”, x); =>3333.33

표준입력 스트림

- Scanner in = new Scanner(System.in);
- String name = in.nextLine(); => 한줄씩 입력
- in.next() => 한 단어 입력
- in.nextInt() => 정수 입력
- in.nextDouble() => 부동소수 입력

입력과 출력

- InputTest.java

```
import java.util.*;

public class InputTest
{
	public static void main(String[] args)
	{
		Scanner in = new Scanner(System.in);
		System.out.print("이름을 입력하세요");
		String name = in.nextLine();

		System.out.print("나이를 입력하세요");
		int age = in.nextInt();

		System.out.println("안녕,"+name+", 내년에 "+(age+1)+"살이군요");
	}
}
```

Formatting!

- Use the `System.out.println()` Method
- `NumberFormat` class in the `java.text` package

  e.g., `NumberFormat.getNumberInstance();`
  `NumberFormat.getCurrencyInstance();`
  `NumberFormat.getPercentInstance();`
- Use the CoreJava `Format` class
  - only have one formatted value at a time
  - if you need to print two values, use two calls

  e.g.,
```
Format.printf("With rate %6.3f", 100*y);
Format.printf("%%, your monthly payment is
                        %10.2f\n, payment);
```

제어 흐름(Control Flow)

- 조건 문 (if)
- 반복문 (for, while,do while)
- 다중선택 (switch)
- Goto 문 없음, but Labeled Breaks
  - 중첩 반복문을 빠져나오기 위해 사용
  - to break out of multiple nested loops
  - Label은 Break하길 원하는 곳의 외부에 Label이 있어야 함
  - 레이블 : (:)

Control Flow – Block

- 블록(block)
  - 중괄호({}) 안에 포함된 일련의 자바 문장
  - 블록은 변수의 범위를 결정

- Main 메소드의 블록내에 중첩된 블록
  - 동일 메소드의 다른 블록에서 같은 이름 변수 불가

```
Public static void main( String
args[])
{ int n;
   …
        { int k;
           …
        } // k는 블록 내에서만 효력
      을 가짐
```

```
Public static void main( String
args[])
{ int n;
   …
        { int k;
           int n;   // Error!!
           …
        } // k는 블록 내에서만 효력
      을 가짐
```

조건문

- 조건문의 형태
  - If (condition) statement;
  - If (condition) { block }

- 조건문의 예 : If (condition) { block }

```
If (yourSales >= target )
{
      Performance = “Satisfactory”;
      Bonus = 100;
}
```

조건문

- 조건문의 형태(else)
  - If (condition) statement else statement
  - If (condition) {block1} else {block2}

- 조건문의 예

```
If (yourSales >= target )
{
        performance = “Satisfactory”;
        bonus = 100+0.01*(yourSales-target);
}
else
{     performance = “UnSatisfactory”;
        bonus = 0;
}
```

조건문

- 조건문의 예 : 다중 else

```
If (yourSales >= 2*target)
{
    performance = “Excellent”;
    bonus = 1000;
}
else if (yourSale >= 1.5*target)
{
    performance = “Fine”;
    bonus=500;
}
else if (yourSale >= target)
{
    performance = “Satisfactory”;
    bonus = 100;
}
else {
    System.out.println(“Your’re fired”);
}
```

반복문 – while

- while 문의 형태
  - While (condition) { block };
  - Condition이 true인 동안 block을 반복 실행
  - 처음 시작할 때 조건이 false이면 실행되지 않음

- while문의 예

```
// balance : 예금액
// payment : 한달월급
// goal : 목표 금액
while ( balance < goal )
{
    balance = (balance+payment)*(1+interest)
    year++;
}
```

반복문 – do

- do 문의 형태
  - do {block} while (condition);
  - 블록을 실행한 후 조건을 검사

- do문의 예 :

```
// balance : 예금액
// payment : 한달월급
// goal : 목표 금액

do {
    balance = (balance+payment)*(1+interest)
    year++;
} while ( balance < goal ) ;
```

반복문 – for

- for 문의 형태
  - for (statement1; expression1; expression2) { block };
  - 다음의 while문과 완전히 동일

```
statement1;
while (expression1)
{
    block;
    expression2;
}
```

  - statement1 : 초기화
  - expression1 : 조건판단
  - expression2 : 사후실행

• for문의 예

```
int sum = 0;

for( int i=0; i<100; i++)
{
    sum=sum+i;
}
```

다중선택 – Switch

- switch 문의 형태
  - choice에는 long형을 제외한 모든 정수형과 char형이 가능

```
switch (choice)
{
    case 1:
        ....
        break;
    case 2:
        ...
        break;
    default :
        //bad input
        break;
}
```

- case에 break문이 없다면?
- default문이 없다면?

Labeled Breaks

- 레이블을 갖는 break
  - goto문 없이 비슷한 역할을 수행
  - 레이블을 갖는 중첩 반복문의 바깥부분으로 빠져나가기 위해사용

- 레이블 없는 break의 예

```
while (years <= 100)
{
    balance = (balance+payment)*(1+interest);
    if (balance > goal) break;
    years++;
}
```

Labeled Breaks

- 레이블 있는 break의 예

```
int n;
   read_data :
while(...)
{
   ...
   for(...)
   {
       if (n<0)
      break read_data;
    }
}
```

클래스 메소드

- Class Methods
  - Method : 클래스 내부의 함수
  - Class Method : 클래스 외부에서 instance생성없이 invoke가능
    - public static {long, int, 클래스명, ...} methodName()

- 다른 메소드내에서 Invoke 방법
  - class 메소드는 다음과 같이 같이 호출 가능
    - Integer.ParseInt(stringVar)
    - NumberFormat.GetCurrencyInstance()
  - 동일 클래스에서 사용될 경우는 클래스명 없이 사용
    - long oddsAre = lotteryOdds(topNumber, numbers);

클래스 변수(Class Variables)

- Class Variable
  - 클래스의 모든 메소드에서 접근 가능한 변수

```
class Employee
{
    private static double socialSecurityRate = 7.62;
    public static void main( String[] args )
    {
       ....
     }
}
```

```
class Employee
{
    public static double socialSecurityRate = 7.62;
    public static void main( String[] args )
    {
       ....
     }
}
```

실습문제

- 입력 : 이름, 월입금액, 월이자율
- 출력 : 6개월 마다의 예금액, 은행고객등급

```
/* 은행고객 등급
    A : 월 은행잔고 > 1000만원
    B :             > 800만원
    C :             > 500만원
    X : 보통
*/
```