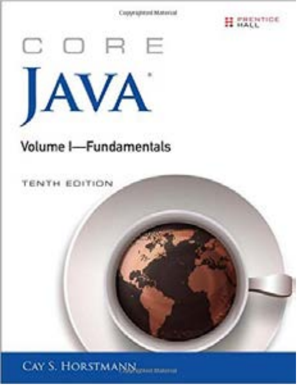


Fundamental Programming Structures in Java – Part 1

Chapter 3, Core Java, Volume I

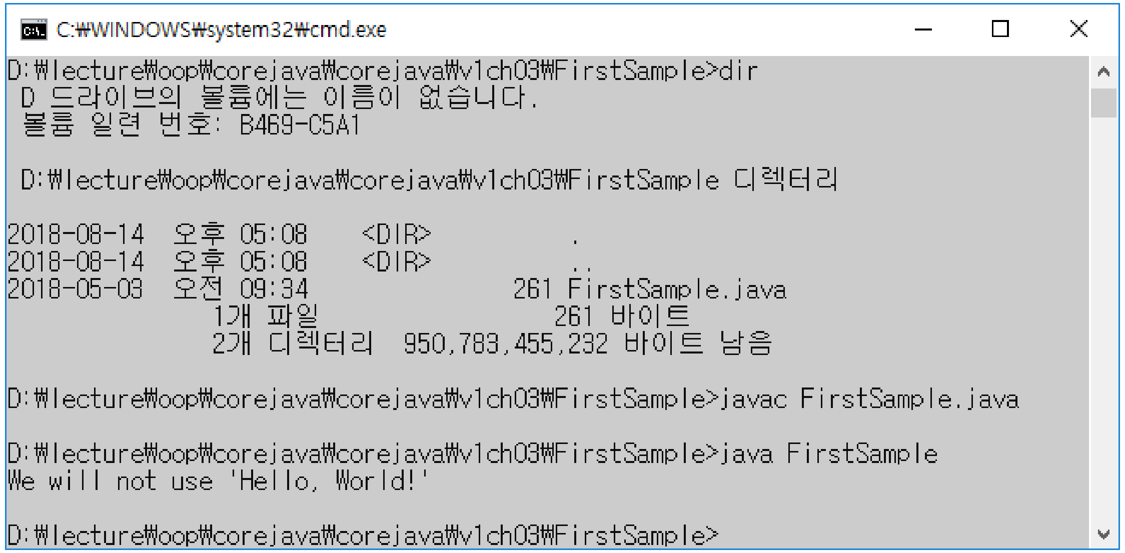
Contents

- A Sample Program in Java
- Data Types
- Variables
- Type Conversions
- Strings
- Input and Output
- Operators
- Control Flows
- Methods (Functions)
- Class Structure
- Arrays
- Command-Line Parameters

A Simple Java Program

- FirstSample.java

```
// This is the first sampe program.
public class FirstSample
{
   public static void main(String[] args)
   {
      System.out.println("We will not use 'Hello, World!'");
   }
}
```



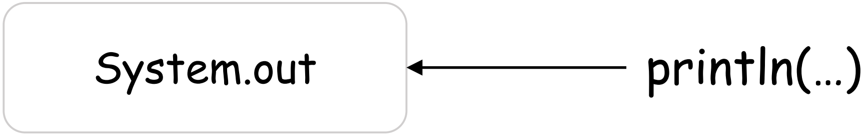
```
D:₩lecture₩oop₩corejava₩corejava₩v1ch03₩FirstSample>dir
 D 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: B469-C5A1

 D:₩lecture₩oop₩corejava₩corejava₩v1ch03₩FirstSample 디렉터리

2018-08-14  오후 05:08    <DIR>          .
2018-08-14  오후 05:08    <DIR>          ..
2018-05-03  오전 09:34               261 FirstSample.java
               1개 파일                 261 바이트
               2개 디렉터리  950,783,455,232 바이트 남음

D:₩lecture₩oop₩corejava₩corejava₩v1ch03₩FirstSample>javac FirstSample.java

D:₩lecture₩oop₩corejava₩corejava₩v1ch03₩FirstSample>java FirstSample
We will not use 'Hello, World!'

D:₩lecture₩oop₩corejava₩corejava₩v1ch03₩FirstSample>
```

- Everything is inside a class.
- The keyword public is an access modifier.
- Java is case sensitive: Main ≠ main.
- By convention, class names are CamelCase.
- Source file must be the same as the class name with the extension .java (e.g. FirstSample.java)
- JVM starts execution with the public main method.

Standard Output

- Standard output stream object : System.out
- Calling a standard output method:

```
System.out.println("We will not use 'Hello, World!'");
```



- Newline

```
System.out.println(“Hello\nWorld!”);
```

- Parentheses needed even if there are no parameters:

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

Comments

- Single-line comments:

```
// like this
```

- Multi-line comments:

```
/*
   like
   this
   ...
*/
```

- Documentation comments:

```
/**
 * This is the first sample program in Core Java Chapter 3
 * @version 1.01 1997-03-22
 * @author Gary Cornell
 */
```

- Caution: /* … */ comments do not nest.

Data Types

- Java is a *strongly typed* language.
  - Every variable must have a declared type
  - Types must be checked in compile or run time.
- Two kinds of types
  - Primitive types : a variable contains a value of the type in the memory
  - Reference types : a variable contains a reference to an object of the type in the memory
    - Classes, Arrays, etc
- There are eight *primitive types* in Java.
  - Numeric types
    - Integral types: int, short, long, byte
    - Floating-point types: float, double
  - Character type : char
  - Boolean type : boolean
- Java has a string type String
  - It is not a primitive type, but a class defined in the library.

Numeric Data Types

- Four integer types

int	4 bytes	–2,147,483,648 to 2,147,483, 647 (just over 2 billion)
short	2 bytes	–32,768 to 32,767
long	8 bytes	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
byte	1 byte	–128 to 127

- Under Java, the ranges of the integer types do not depend on the machine
  - The ranges for the various types are fixed.
  - Easy to write cross-platform programs.
  - In C and C++, the size of integral types such as `int` and `long` depend on the target platform.
    - int type : 16 bits for 16-bit processors, 32 bits for 32-bit processors
- Literals:
  - Long: 4000000000L
  - Hex: 0xCAFE
  - Binary: 0b1111_0100_0010_0100_0000 (since Java 7)

Numeric Data Types

- Two floating-point types:

float	4 bytes	Approximately ±3.40282347E+38F (6–7 significant decimal digits)
double	8 bytes	Approximately ±1.79769313486231570E+308 (15 significant decimal digits)

- All floating-point number representations follow the IEEE 754 specification.
- Literals
  - By default, floating-point constants are *double-precision*.
  - float literals: 3.14F or 3.14f

- Special values to denote overflows and errors (rarely used):
  - Double.POSITIVE_INFINITY (e.g. 5.0/0.0)
  - Double.NEGATIVE_INFINITY (e.g. -5.0/0.0)
  - Double.NaN. (e.g. 0.0/0.0, $\sqrt{-5.0}$)
  - 0/0 is an exception (integer division)

The char Type

- The char type describes individual characters (but, not all characters)

```
char ch = 'a';
char newline = ‘\n’;  // use escape sequence
char uniChar = '\u03A9'; // Unicode escape sequence for Greek omega character(Ω)
```

- Escape Sequences

Escape Sequence	Name	Unicode Value
\b	Backspace	\u0008
\t	Tab	\u0009
\n	Linefeed	\u000a
\r	Carriage return	\u000d
\”	Double quote	\u0022
\’	Single quote	\u0027
\\	Backslash	\u005c

Unicode and the char Type

- Unicode was invented to overcome the limitations of traditional encoding schemes such as ASCII, ISO 8859-1, etc.
- Unicode was originally designed as a fixed-width 16-bit characters.
- Unfortunatly, the a 16-bit encoding are not sufficient to represent all characters especially including CJK Unified Ideographs (87,887 한자).
- The Unicode standard therefore has been extended to allow up to 1,112,064 characters.
- Those characters that go beyond the original 16-bit limit are called *supplementary characters.*
- A code point is a unique value assigned to each Unicode character
  - The valid code points for Unicode are U+0000 to U+10FFFF
  - U+0000 ~ U+FFFF : Basic Multilingual Plane (Classic Unicode characters)
  - U+10000 ~ 10FFFF : 16 Supplmentary Planes (supplementary characters)

Unicode and the char Type

- Unicode Encoding Schemes
  - UTF-32 : same as the code points
  - UTF-16 : maps each code point to one or two unsigned 16-bit values, *code units*
    - BMP : one code units (2048 values, U+D800~DFFF, are reserved for supplementary characters)
    - Supplementary Planes : 2 code units
    - 'A' has "code point" U+0041 and is encoded by a single code units (hex 0041 or decimal 65).
    - '𝕆' has "code point" U+1D546 and is encoded by two code units (hex D835 and DD46).
  - UTF-8 : maps each code point to one to four byes
- A char value in Java describes a code unit in the UTF-16 encoding.
  - One char value cannot represents supplementary characters
  - A supplementary character can be represented by two-char value array or a String.
  - The Character class provides various methods that let you map between various char and code point-based representations

The boolean Type

- Two values: false, true

```
boolean b = a > 0 && a < 10;
```

- No conversion between int and boolean
  - In C & C++, numbers and even pointers can be used in place of boolean values.

```
if ( x = 0 )           // always false in C
   (a)                 // compile error in Java
else
   (b)
```

Variables

- Every variable must be declared with a type, which comes before the name.
- Local and Non-local variables
    - Local variables
        - Defined inside a method or inside a block
        - Parameter variables
        - Scope : a method or block
    - Non-local variables
        - Defined outside methods
        - Scope : a class

```
pubic class A
{
    static int x;  // non-local variable
    public static void main(String[] args)
    {
        double a;  // local variable
        …x…          // non-local x
        f(a);         // local a
        …
    }
    public static int f( double x )
    {
         …x…  // local x
    }
}
```

Constants

- Constant declared with final:

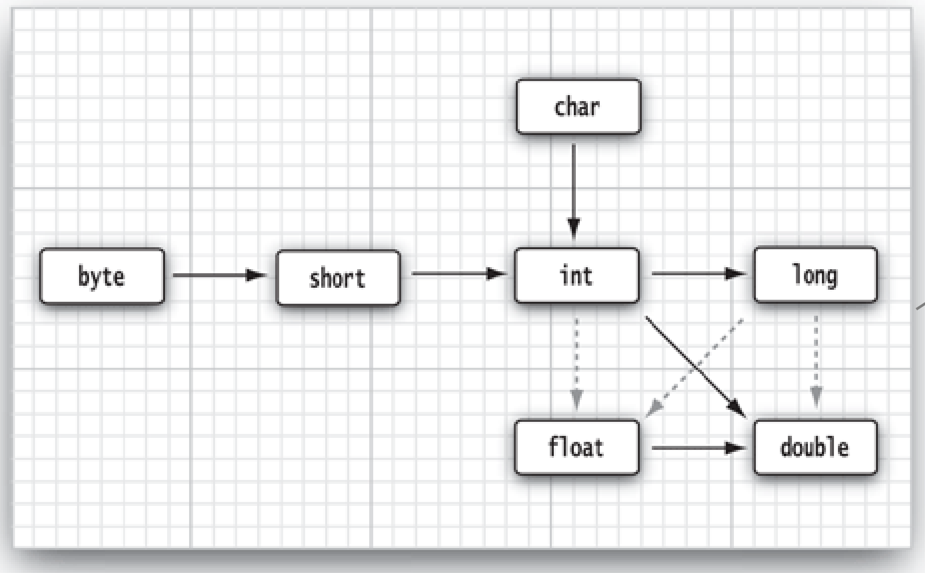
```
pubic class Constants
{
    public static void main(String[] args)
    {
        final double CM_PER_INCH = 2.54;
        …
    }
}
```

- Class-scope constants : static final

```
pubic class Constants
{
    public static final double CM_PER_INCH = 2.54;
    public static void main(String[] args)
    {
        …
    }
}
```

Type Conversions

- Automatic Type Conversion (Widening)



Dotted arrows indicate possible precision loss.

- Explicit Type Conversions (using cast operator)

```
double x = 9.997;
int nx = (int) x;
int rx = (int) Math.round(x);
```