Topics:

- Blank filling questions

- Questions to ask the concept/fundamentals of important themes of Java - 서술형

- Code reading questions (to predict the output result of given codes) - 결과 예측

- Coding questions (Hand coding) - 1 or 2 문제, OOP와 OOD는 문제 X

- OOD, OOP -> 개념문제에 더 초점

Tips:

- For the questions on the basics/concepts of Java, you need to study all chapters (Introduction to Java ~ Method Overriding).

- Coding-reading questions mainly focus on types/operators/loop/OOP +@

- Hand coding questions mainly focus on operator/conditional/loop +@

Machine Language, low-level

Programming Language, high-level

Source file

Compile

Java

* WORA (Write Once Run Anywhere): the slogan of JAVA, platform independent
* Bytecode & JVM (Java Virtual Machine) -> Machine Language
* JVM -> platform dependent
* OOP
  + Encapsulation
  + Inheritance
  + Polymorphism

Variables

* type var\_name = value;
* need to be initialized before being used
* accessed by its name

Variavle Naming

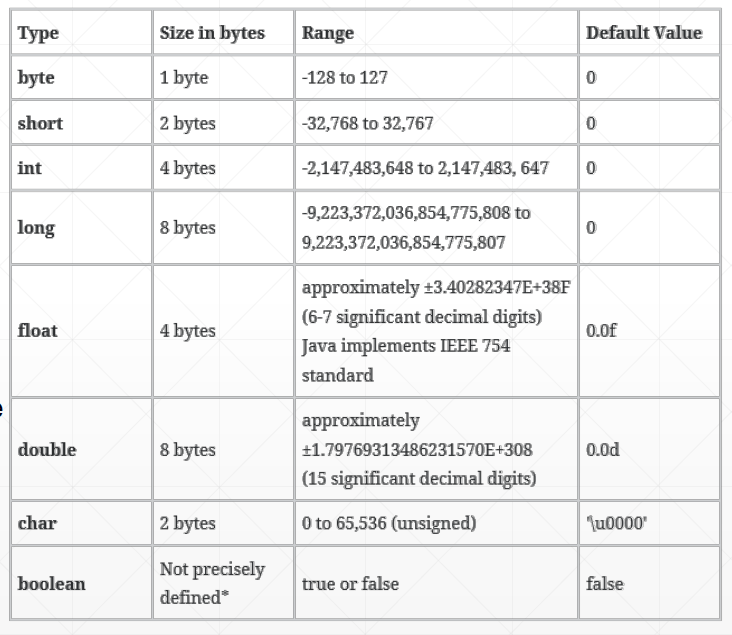
* case sensitive
* unlimited length of name
* letter, $, \_
* No java keyword
* No true, false, null
* No whitespace, special character

Type

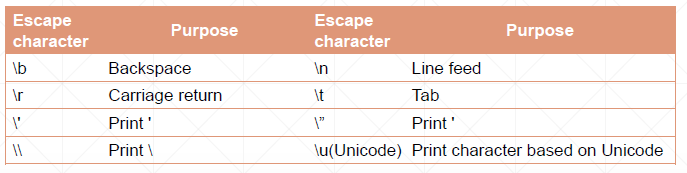
* Primitive Type
  + types of number, character, boolean
* Non-primitive Type
  + String, array, class

8bit = 1 byte

* Literal



* integer literal
  + default type: int
  + long type: end with suffix ‘L’ or ‘l’
  + binary: 0b~ (0 or 1)
  + octal: 0~ (0~7)
  + decimal:
  + heximal: 0x~ (0~F)
* float literal
  + double: default, suffix D or d
  + float: suffix F or f
  + e & E: scientific number
* String
  + “” double quotes
  + Escape Character: \~



Constant

* using ‘final’
* Naming: UPPERCASE\_NAME
* final double PI = 3.141592;

Promotion

* Automatic, small to big
* Passing a smaller size type to a larger size type
* Performing an arithmetic operation with integer-type values
  + Byte, short, char type values are automatically converted to int type values
* Performing an arithmetic operation with different types of values

Casting

* manual, big to small
* small\_type variable = (small\_type) big\_type;\
* can lose value

unicode

\u0041, 65 : A

\u0061, 97 : a

Scanner

* Scanner scanner = new Scanner(System.in);
* scanner.next(); …

printf

* % [argument\_index$] [flags] [width] [.precision] conversion
* printf(“%1$0+5.2f”, -3.141592) → result: -3.14

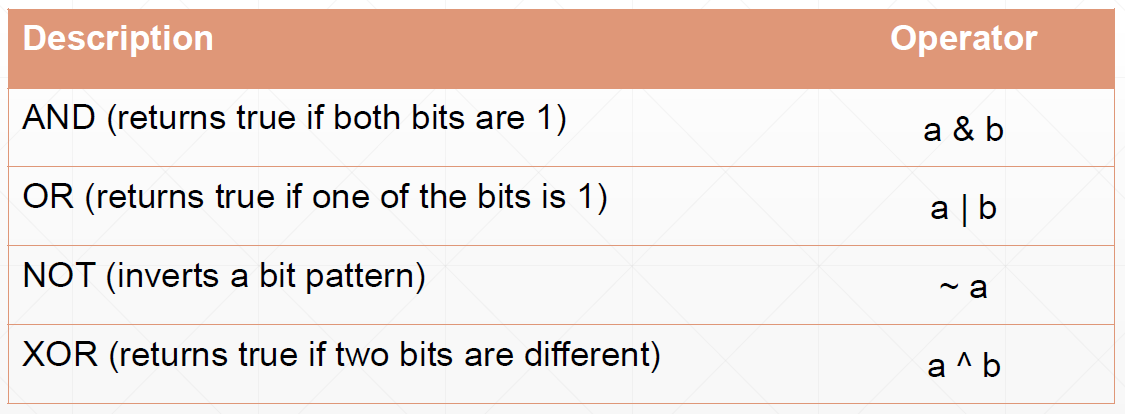
Basic Operator

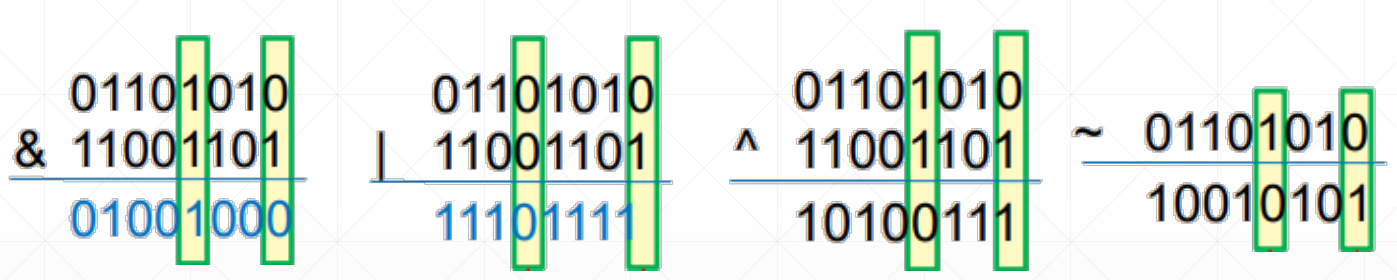
* → String + number is available
* when a = 0
* prefix unary: ++a → add 1 and return 2
* suffix unary: a++ → return 1 and add 1

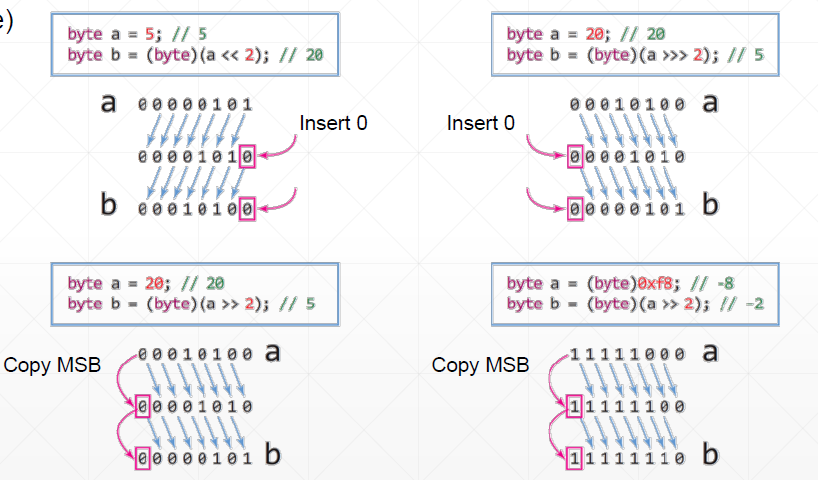
Relational operator : > < == != >= <= → return boolean

Conditional operator : && || ! → return boolean

bit & shift







IF

* if (condition){ }

else if (condition) { }

else { }

* Condition ? op2:op3;

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

* int s;

if (x>y) {s = 1;}

else {s = -1;}

* multiple if-else

mutually exclusive

SWITCH

* swtitch (expression){

case x:

code block

break;

case y:

code block

break;

default:

code block

}

* Case value → char, integer, String only!

Loop

* For
* While
* Do-While

For

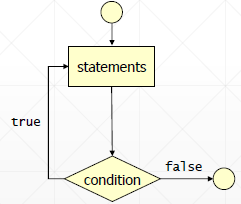
* for (initialization; condition; increment) { }
* condition = true or empty → infinite loop
* initialization & increment → can have multiple statement

While

* while (condition) { }
* condition → return true or false

Do-While

* Do { } while (condition);
* Statements in Do block should be executed at least once!



Break;

break in the nested loop

* only for current loop

Continue;

* skip remaining statements
* move to condition of the loop
* Only for current “Loop”
* Only used in Loop

Primitive Type

* integer
* floating point
* boolean

Reference Type

* array
* enumeration
* class (+String, Wrapper)
* interface
* null

Array, for-each

* for (type variable : array) { }