Tuesday, 24.04.2018 - 8:00 PM

Java Operators

- Arithmetic operators
- Relational operators
- Bitwise operators
- Logical operators
- Assignment operators
- Misc operators

Arithmetic operators

+	addition
−	subtraction
*	multiplication
/	division
%	modulus (remainder)

10 % 5 = 0
6 % 4 = 2

Relational operator

a = 8

→	==	equal to	(a == 8) → true
	!=	not equal to	(a != 8) → false
	>	greater than	
	<	less than	
	>=	greater than or equal to	
	<=	less than or equal to	

Bitwise operators

a = 11
b = 14

&	Bitwise AND	11 & 14
|	Bitwise OR	11 | 14
^	XOR (exclusive OR)	11 ^ 14
~	complement	~ 11
<<	left shift	0 0 1 0 1 1 << 2 → 0 0 1 0 1 1 0 0
>>	right shift	0 0 1 0 1 1 >> 2 → 0 0 1 0 1 1

Logical operators

&&	logical AND	(a && b) → false
||	logical OR	(a || b) → true
!	logical NOT	!(a && b) → true

a = true
b = false

* Both sides of logical operator have boolean values.

Assignment operators

=	assignment (assigns a value to LHS)	a = 10 b = 20 c = a + b
+=		c += a ↓ c = c + a
−=		c -= a ↓ c = c - a
*=		
/=		
%=		

Misc operators

Ternary operator / conditional operator

→ used to evaluate boolean expression

→ goal is also operator is as similar which reduces to be assigned to a variable.

Syntax:

variable x = (expression) ? value if true : value if false;

a = 10
b = 20
c = 30

boolean flag = (b == a && b) ? true : false;

flag = false;

Conditional statements

(A) **If**

(conditional statements)

(A is keeping subtle) THEN

if (condition) {
// code to be executed
}

→ 3 statement

Syntax: if (condition) {
// code to be executed
}

3

(B) if - else

Syntax: if (condition) {
// code if condition is true
} else {
// code if condition is false
}

3

Best: Given a year, check if it is a leap year.

(C) if - else if statement

Syntax: if (condition 1) {
// code if condition 1 is true
} else if (condition 2) {
// code if condition 2 is true
} else if (condition 3) {
// code if condition 3 is true
} else {
// code
}

3

Nested if statement

Syntax: if (condition) {
if (condition) {
}
}

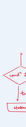
3

Switch statement

→ execute one statement from several conditions

Syntax: switch (expression) {
case value 1 :
// code if value 1 matches expression
break;
case value 2 :
// code if value 2 matches expression
break;
default :
// code to be executed if all values are false
}

3



The first matched statement is a fall through statement.

If means that it not exits all statements after the first match is found in break statement or the end of the switch.

Precedence & Associativity

→ operator precedence determines an order in which the operators in an expression are evaluated.

Let - 4 + 2 * 3 / 2

When two operators share a common operand, the operator with the higher precedence is evaluated first.

Operator precedence table:

→ values + precedence, operators are listed in decreasing order of precedence.

Operator precedence

✓ postfix increment & decrement	++, --	left to right
prefix inc, dec, unary	++, --, +, -, ~, !	right to right
multiplicative	*, /, %	left to right
additive	+ -	L to R
shift	<<, >>, >>>	L to R
relational	<, >, <=, >=	L to R
equality	== , !=	
bitwise AND	&	
^ XOR	^	
| OR	|	
logical AND	&&	
logical OR	||	
ternary	? :	→ R to L
assignment	=, +=, -=, *=,	→ R to L

Operator associativity

If an expression has two operators with the same precedence, then operator is evaluated according to associativity.

(left to right) or (right to left)