



បណ្ណបណ្ណាលក្នុង សម្បត្តិ អេច អ ឌី Korea Software HRD Center

Syntax និង Data Type ប្រព័ន្ធគណិត Java Java Syntax and Data Type

ណែនាំដោយ : Dr. Kim Tae Kyung



<http://www.kshrd.com.kh>

មាតិកា

១. ការប្រកាសអថេរក្នុង Java
២. ការស្វែងយល់អំពី Java Naming Rule
៣. ការប្រើប្រាស់ Comments នៅក្នុងភាសា Java
៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class
៥. ការប្រើប្រាស់នូវ Operators

១. ការប្រកាសអថេរក្នុង Java

- តើអថេរ **Variable** គឺជាអ្វី?
 - **Variable** គឺជាអញ្ញតដែលត្រូវបានគេផ្ទុកជាបណ្តោះអាសន្ននៅលើ **Memory** នៅពេល **Run time** ហើយតំលៃទាំងនោះអាច ប្រែប្រួលបានផងដែរ ។ អថេរនីមួយៗមាននូវប្រភេទទិន្នន័យច្បាស់លាស់ដែលកំណត់នូវទំហំរបស់អញ្ញតនោះ។
 - ដើម្បីប្រកាស Variable ក្នុងភាសា Java យើងត្រូវ៖

[**modifier**] **DataType** variable [= value][, variable [= value] ...] ;

ប្រភេទទិន្នន័យ

ឈ្មោះអថេរ

តំលៃរបស់អថេរ

១. ការប្រកាសអថេរក្នុង Java (ត)

- ខាងក្រោមនេះគឺជាឧទាហរណ៍មួយចំនួនក្នុងការប្រកាសអថេរ៖

```
int a, b, c; // Declares three ints, a, b, and c.
```

```
int a = 10, b = 10; // Example of initialization
```

```
byte B = 22; // initializes a byte type variable B.
```

```
double pi = 3.14159; // declares and assigns a value of PI.
```

```
char a = 'a'; // the char variable a is initialized with value 'a'
```

២. កាស្វែងយល់អំពី Java Naming Rule

Java មានគោលការណ៍ កំណត់ឈ្មោះ Variable ដូចតទៅ:

- Case-Sensitive ។ អក្សរតូច និងធំ គ្មានន័យដូចគ្នាទេ។
- មិនកំណត់ចំនួនតួអក្សរ
- ផ្ដើមដោយអក្សរ លើកលែងតែសញ្ញា “ \$ ” និង “ _ ”
- ជៀសវាងប្រើសញ្ញា “ \$ ” ជានិច្ច

▪ ឧទាហរណ៍៖

```
int $a; //$ sign variable name  
int _b; //_ sign variable name
```


២. ការស្វែងយល់អំពី Java Naming Rule (ត)

Java មានគោលការណ៍ កំណត់ឈ្មោះ Variable ដូចតទៅ:

- មិនអនុញ្ញាតឱ្យប្រើ Whitespace

- ឧទាហរណ៍៖ `int car speed; // variable name can't contain white space`



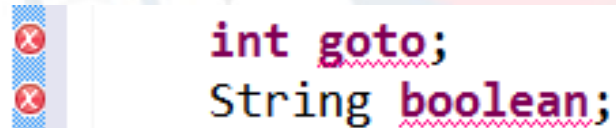
```
int car_speed;  
int emp_id; |
```

- បើឈ្មោះមាន១ពាក្យ ប្រើអក្សរតូចទាំងអស់។ តែបើមានច្រើនជាង១ពាក្យ ចូរប្រើអក្សរទី១ក្នុងពាក្យបន្តបន្ទាប់និមួយៗជាអក្សរធំ ។ ឧ. gearRatio, currentGear...
- ប្រើឈ្មោះជាពាក្យពេញ ដើម្បីងាយអាន និងយល់។ ជៀសវាងប្រើអក្សរកាត់ ឬពាក្យសំងាត់ (encrypt abbr)

២. ការស្វែងយល់អំពី Java Naming Rule (៣)

Java មានគោលការណ៍ កំណត់ឈ្មោះ Variable ដូចតទៅ:

- បើ Variable ផ្ទុកតំលៃថេរ ចូរប្រើអក្សរធំទាំងអស់ ហើយភ្ជាប់ពាក្យនិមួយៗដោយ underscore “_” ។
- មិនត្រូវដាក់ឈ្មោះ ដូចនឹង keyword (Ex: goto, boolean, int, double, etc...)
- ឧទាហរណ៍៖

A screenshot of a code editor showing two lines of Java code. The first line is 'int goto;' and the second line is 'String boolean;'. Both 'goto' and 'boolean' are underlined with red wavy lines, indicating they are invalid because they are Java keywords. To the left of the code, there is a vertical blue bar with two red 'X' icons, one next to each line of code.

```
int goto;  
String boolean;
```

២. កាស្ត្រូយល់អំពី Java Naming Rule (គ)

Java Language Keywords

Here is a list of keywords in the Java programming language. You cannot use any of the following as identifiers in your programs. The keywords `const` and `goto` are reserved, even though they are not currently used. `true`, `false`, and `null` might seem like keywords, but they are actually literals; you cannot use them as identifiers in your programs.

<code>abstract</code>	<code>continue</code>	<code>for</code>	<code>new</code>	<code>switch</code>
<code>assert^{***}</code>	<code>default</code>	<code>goto[*]</code>	<code>package</code>	<code>synchronized</code>
<code>boolean</code>	<code>do</code>	<code>if</code>	<code>private</code>	<code>this</code>
<code>break</code>	<code>double</code>	<code>implements</code>	<code>protected</code>	<code>throw</code>
<code>byte</code>	<code>else</code>	<code>import</code>	<code>public</code>	<code>throws</code>
<code>case</code>	<code>enum^{****}</code>	<code>instanceof</code>	<code>return</code>	<code>transient</code>
<code>catch</code>	<code>extends</code>	<code>int</code>	<code>short</code>	<code>try</code>
<code>char</code>	<code>final</code>	<code>interface</code>	<code>static</code>	<code>void</code>
<code>class</code>	<code>finally</code>	<code>long</code>	<code>strictfp^{**}</code>	<code>volatile</code>
<code>const[*]</code>	<code>float</code>	<code>native</code>	<code>super</code>	<code>while</code>

៣. ការប្រើប្រាស់ Comments នៅក្នុងភាសា Java

តើអ្វីទៅជា **Comment**?

- **Comments** ត្រូវបានគេសំរាប់ពន្យល់អត្ថន័យរបស់កូដ។ កូដដែលប្រើ Comment នោះប្រក្រាមវាមិនត្រូវបាន read ដោយ compiler នោះទេ។
- ក្នុងភាសាJava Comments ត្រូវបានបែងចែកចេញជា **៣ប្រភេទ**៖
 - **/* text */** ប្រើសំរាប់ការដាក់ comment ទៅលើ block នៃកូដ
 - **// text** ប្រើសំរាប់ការដាក់ comment ទៅលើ line នៃកូដ
 - **/** documentation */** ប្រើសំរាប់ដាក់ comment សំគាល់លើ class ឬក៏ object

၈. ကပြေပြာသံ Comments နေကွဲသကော Java(၈)

```
public class HelloWorld {  
  
    public static void main(String[] args) {  
        // Prints Hello, World! on standard output.  
        System.out.println("Hello World!");  
    }  
}
```

៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class

- នៅក្នុង Java Data Type ត្រូវបានចែកចេញជាពីរប្រភេទគឺ Primitive និង Non-Primitive
- **Primitive Type**: គឺជាប្រភេទ **Data Type** ដែលត្រូវបានបង្កើតមកស្រាប់ជាមួយ Programming Language ហើយវាមានឈ្មោះដូចទៅនឹង **Reserved Keyword**.

Primitive Data Type មាន ៨ គឺ៖

- byte
- int
- float
- boolean
- short
- long
- double
- char

៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (៩)

- **Wrapper Class**: ប្រើដើម្បីបំប្លែង Primitive Data Type ទៅជា object.
- ហេតុអ្វីបានជាយើងប្រើ **Wrapper Class**?
 - Build-in method
 - No need to use casting

`Integer` age = 20;

`String` strAge = age.**toString()**;

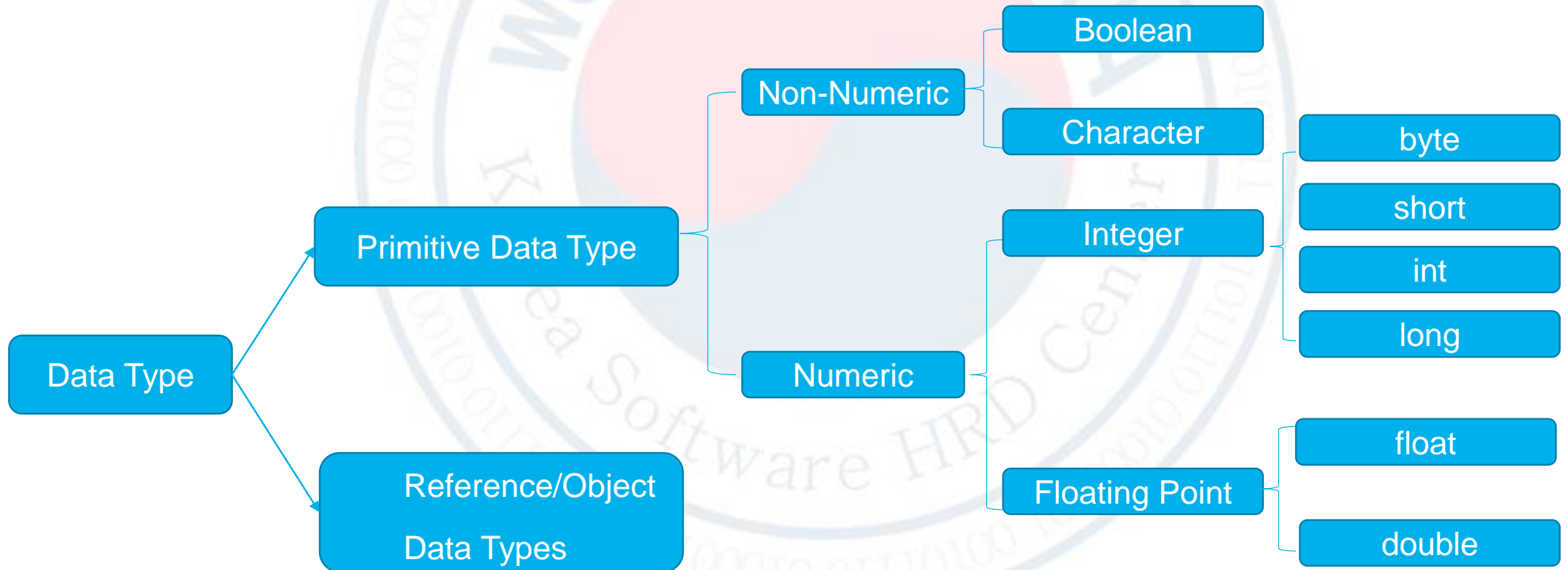
៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (៩)

- នេះគឺជាតារាងនៃ primitive data type ដែលត្រូវជាមួយនិង primitive wrapper class

Primitive Data Types	Wrapper Classes
int	Integer
short	Short
long	Long
byte	Byte
float	Float
double	Double
char	Character
boolean	Boolean

៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (ត)

- Data Type ក្នុងភាសា Java ត្រូវបានចែកជាពីរប្រភេទធំៗគឺ៖



៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (៩)

Primitive Data Type

- byte:
 - ប្រភេទជា byte មានទំហំទិន្នន័យ 8-bits
 - តំលៃអប្បបរមា -128 (-2^7)
 - តំលៃអតិបរមា 127 (inclusive)($2^7 - 1$)
 - តំលៃ Default របស់វាគឺ 0
 - Example: byte a = 100 , byte b = -50

៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (ត)

- short:
 - ប្រភេទជា short មានទំហំទិន្នន័យ 16-bits
 - តំលៃអប្បបរមា -32,768 (-2^{15})
 - តំលៃអតិបរមា 32,767 (inclusive) ($2^{15} - 1$)
 - តំលៃ Default របស់វាគឺ 0
 - Example: short s = 10000, short r = -20000

៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (៩)

- int:
 - ប្រភេទជា int មានទំហំទិន្នន័យ 32-bits
 - តំលៃអប្បបរមា - 2,147,483,648. (-2^{31})
 - តំលៃអតិបរមា 2,147,483,647(inclusive). $(2^{31} - 1)$
 - តំលៃ Default របស់វាគឺ 0
 - Example: int a = 100000, int b = -200000

៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (៩)

- long:
 - ប្រភេទជា long មានទំហំទិន្នន័យ 64-bits
 - តំលៃអប្បបរមា $-9,223,372,036,854,775,808.(-2^{63})$
 - តំលៃអតិបរមា $9,223,372,036,854,775,807$ (inclusive). $(2^{63} - 1)$
 - តំលៃ Default របស់វាគឺ 0L
 - Example: long a = 100000L, int b = -200000L

៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (៩)

- float:
 - ប្រភេទ float មានទំហំទិន្នន័យជា single-precision 32-bit IEEE 754 floating point
 - តំលៃ Default របស់វាគឺ 0.0f
 - ប្រភេទទិន្នន័យ float គឺគ្មានតម្លៃជាក់លាក់ដូច Currency ឡើយ
 - Example: float f1 = 234.5f

៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (ត)

- double:
 - ប្រភេទជា double មានទំហំទិន្នន័យជា double-precision 64-bit IEEE 754 floating point.
 - តំលៃ Default របស់វាគឺ 0.0d
 - ប្រភេទទិន្នន័យជា float គឺគ្មានតម្លៃជាក់លាក់ដូច Currency ឡើយ
 - Example: double d1 = 123.4

៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (ត)

- boolean:
 - ប្រភេទជា boolean មានទំហំទិន្នន័យ 1-bit
 - តំលៃរបស់ Boolean មានពីរប្រភេទគឺ true និង false
 - តំលៃ Default របស់វាគឺ false
 - Example: boolean one = true

៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (៩)

- char:
 - ប្រភេទជា char មានទំហំទិន្នន័យជា single 16-bit Unicode character
 - តំលៃអប្បបរមា គឺ '\u0000' (or 0).
 - តំលៃអតិបរមា គឺ '\uffff' (or 65,535 inclusive)
 - តំលៃរបស់ char ត្រូវបានផ្ទុកជាលក្ខណៈតួអក្សរ
 - Example: char letterA ='A'

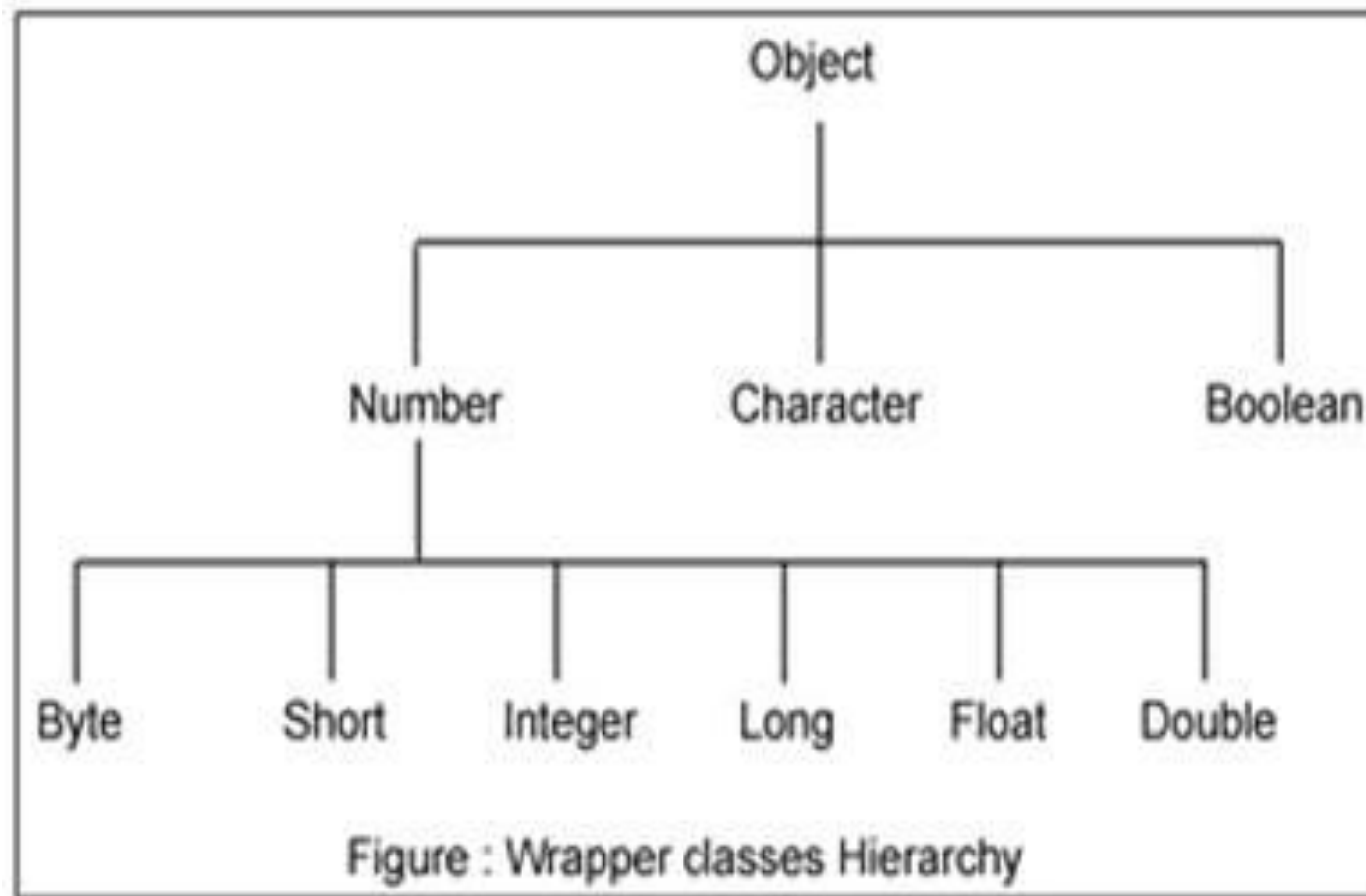
៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (៩)

Wrapper Class

- Primitive Data Type នីមួយៗ មាន Class មេរបស់វា(Wrapper Class)ជានិច្ច។
- Wrapper Class ជាផ្នែកមួយរបស់ java.lang package ដែលមិនចាំបាច់ import ទេ។
- ការប្រើ Wrapper Class មានគោលបំណងពីរសំខាន់ៗ គឺ៖
 - ធ្វើអោយ Primitive Type ក្លាយជា Object ដែលអាចអោយ Object អាចមានលទ្ធភាពប្រើប្រាស់Methods ទាំងឡាយនៅក្នុង wrapper class ដូចជា បោះចូល ArrayList, Hashset, HashMap etc. collection.

៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (៩)

ខាងក្រោមនេះបង្ហាញពីរចនាសម្ព័ន្ធរបស់ Wrapper Class



៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (៩)

Method	Purpose
intValue()	returns the value of this Integer as an int

```
int b=10;
```

```
Integer a = new Integer(b);
```

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

```
→ 10
```

៤. ការប្រើប្រាស់ Primitive Data Type / Wrapper Class (ត)

- **Reference Data Type** គឺជា Data Type ប្រើប្រាស់សំរាប់បង្កើត Object។ សរុបសេចក្តីមកវាគឺជា class ផ្ទាល់តែម្តង។ ឧទាហរណ៍ដូចជា class Employee, class Animal, class Student, ...ជាដើម។
- Default value របស់ Reference Data Type គឺ Null
- Example: `Animal animal = new Animal("giraffe");`

៥. ការប្រើប្រាស់ Operators

- Operators គឺជាសញ្ញាពិសេសដែលដើរតួក្នុងការធ្វើប្រមាណវិធីជាមួយនិង អង្គ មួយ ពីរ ឬ បី។ ហើយវានិងផ្តល់តម្លៃមួយមកវិញ។
- វាពិតជាមានសារប្រយោជន៍ណាស់នៅពេលដែលយើងដឹងពីលំដាប់លំដោយរបស់ operator ក្នុងការធ្វើប្រមាណវិធី។ ខាងក្រោមនេះគឺជាតារាងដែលបង្ហាញពីលំដាប់លំដោយ operator ដែលនៅខ្ពស់ជាងគេគឺមានអាទិភាពមុនគេ។ បើវានៅជួរជាមួយគេគឺវាមានអាទិភាពដែលគ្នា មួយណានៅមុខមួយនោះគឺធ្វើការប្រតិប្បត្តិមុនគេ។ binary operators ទាំងអស់គឺគិតពីឆ្វេងទៅស្តាំ លើកលែងតែ assignment operators ដែលគិតពីស្តាំមកឆ្វេង។

៥. ការប្រើប្រាស់ Operators (ត)

Operator Precedence	
Operators	Precedence
postfix	<i>expr++ expr--</i>
unary	<i>++expr --expr +expr -expr ~ !</i>
multiplicative	<i>* / %</i>
additive	<i>+ -</i>
shift	<i><< >> >>></i>
relational	<i>< > <= >= instanceof</i>
equality	<i>== !=</i>
bitwise AND	<i>&</i>
bitwise exclusive OR	<i>^</i>
bitwise inclusive OR	<i> </i>
logical AND	<i>&&</i>
logical OR	<i> </i>
ternary	<i>? :</i>
assignment	<i>= += -= *= /= %= &= ^= = <<= >>= >>>=</i>

៥. ការប្រើប្រាស់ Operators (ត)

Assignment Operator

- Operators ដែលយើងជួបជាញឹកញាប់គឺ assignment operator "="។ វាធ្វើការផ្តល់តម្លៃពី operand ខាងស្តាំទៅឆ្វេង។
 - Operator នេះក៏អាចប្រើជាមួយក្នុងការ assign object ផងដែរ។
- ឧ.

```
int cadence = 0;  
int speed = 0;  
int gear = 1;
```

៥. ការប្រើប្រាស់ Operators (ត)

Arithmetic Operators

- Java ផ្តល់ឱ្យនូវ operators ដែលធ្វើការបូក ដក គុណ និងចែក។ ហើយក្នុងនោះមានសញ្ញា % ដែលធ្វើប្រមាណវិធីចែក តែវាផ្តល់នូវតម្លៃ មកវិញជា សំណល់។
- យើងក៏អាចផ្គុំសញ្ញា arithmetic ជាមួយនិងសញ្ញា assignment ដើម្បីបង្កើតជា compound assignments ។ឧ. `X +=1;` និង `x=x+1;` គឺមានន័យដូចគ្នា យកតម្លៃ x បូកបន្ថែមមួយ។ សញ្ញា + ក៏អាចប្រើក្នុងការភ្ជាប់ string ពីចូលគ្នា។

Operator	Description
+	Additive operator (also used for String concatenation)
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Remainder operator

៥. ការប្រើប្រាស់ Operators (ត)

Unary Operators

- សញ្ញា unary គឺតម្រូវតែមួយ operand ទេ។ វាធ្វើការដូចជា បង្កើនឬបន្ថយតម្លៃរបស់ operand នោះ ហើយវាអាចធ្វើឲ្យតម្លៃរបស់ operand នោះក្លាយជាវិជ្ជមាន និងអវិជ្ជមាន និងមួយទៀតគឺ បញ្ជាក់សត្វតម្លៃរបស់ Boolean។
- សញ្ញា increment/decrement operators អាចដាក់ពីមុខឬពីក្រោយ operand លទ្ធផលរបស់វាគឺ បង្កើនតម្លៃមួយ។ វាខុសគ្នាត្រង់ prefix version (++result) វាធ្វើការកើនតម្លៃមុនការប្រើប្រាស់ រីឯ postfix version (result++) គឺវាយកតម្លៃដើមមកប្រើប្រាស់សិនទើបកើនតម្លៃក្រោយ។

៥. ការប្រើប្រាស់ Operators (ត)

Unary Operators

Operator	Description
+	Unary plus operator; indicates positive value (numbers are positive without this, however)
-	Unary minus operator; negates an expression
++	Increment operator; increments a value by 1
--	Decrement operator; decrements a value by 1
!	Logical complement operator; inverts the value of a boolean

៥. ការប្រើប្រាស់ Operators (ត)

Equality and Relational Operators

- សញ្ញា equality និង relational គឺប្រើសម្រាប់កំណត់ថា operand មួយគឺធំជាង តូចជាង ស្មើ ឬ មិនស្មើនឹង operand ដទៃទៀត។ ជាទូទៅ operators ទាំងអស់នេះគឺមើលទៅប្រហាក់ប្រហែល គ្នា តែត្រូវចំណាំថាយើងត្រូវប្រើ "==" មិនមែន "=" នៅពេលដែលធ្វើការ test រកតម្លៃពីរនៃ primitive ថាស្មើគ្នាអត់។

== equal to

>= greater than or equal to

!= not equal to

< less than

> greater than

<= less than or equal to

៥. ការប្រើប្រាស់ Operators (ត)

Conditional Operators

- សញ្ញា && និង || ដើរតួជា Conditional-AND និង Conditional-OR operations ជាមួយ Boolean expression ពីរឡើង។
- Conditional operator មួយទៀតគឺ ?: ដែលមានលក្ខណៈដូច if-then-else statement។ សញ្ញា នេះត្រូវបានគេហៅថា ternary operator ព្រោះវាប្រើ operand បីទៅបី។ នៅក្នុងឧទាហរណ៍ខាងក្នុង គឺថាបើ someCondition is true, វា នឹង assign តម្លៃនៃ value1 ទៅ result. តែបើ false, វា នឹង assign តម្លៃនៃ value2 ទៅ result

៥. ការប្រើប្រាស់ Operators (ត)

Type Comparison Operators (instanceof)

- instanceof operator ប្រើសម្រាប់ប្រៀបធៀប object មួយទៅកាន់ប្រភេទជាក់លាក់។ យើងអាចប្រើវាសម្រាប់ធ្វើការ test ថាបើ object មួយជា instance របស់ class មួយរឺអត់ ដែលវាបានធ្វើការ implement ពី interface ជាក់លាក់ណាមួយ។
- ពេលយើងប្រើ instanceof operator ត្រូវចាំថា null គឺមិនមែនជា instance នៃអ្វីទាំងអស់។