The int Arithmetic Operators

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operator</th>
<th>Kind</th>
<th>Syntax</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;5</td>
<td>+</td>
<td>infix</td>
<td>(x + y)</td>
<td>add (y) to (x)</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>infix</td>
<td>(x - y)</td>
<td>subtract (y) from (x)</td>
</tr>
<tr>
<td>&lt;4</td>
<td>*</td>
<td>infix</td>
<td>(x \times y)</td>
<td>multiply (x) by (y)</td>
</tr>
<tr>
<td></td>
<td>/</td>
<td>infix</td>
<td>(x / y)</td>
<td>divide (x) by (y)</td>
</tr>
<tr>
<td></td>
<td>%</td>
<td>infix</td>
<td>(x % y)</td>
<td>remainder of (x / y)</td>
</tr>
<tr>
<td>&lt;2</td>
<td>++</td>
<td>prefix</td>
<td>++(x)</td>
<td>identity</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>prefix</td>
<td>--(x)</td>
<td>negate (x)</td>
</tr>
<tr>
<td>&lt;1</td>
<td>+</td>
<td>postfix</td>
<td>++(x)</td>
<td>(x = x + 1); result = (x)</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>postfix</td>
<td>--(x)</td>
<td>(x = x - 1); result = (x)</td>
</tr>
</tbody>
</table>

Examples

```java
double price;
price = 17.25;
int quantity = 25;
boolean isValid = false;
double cost;
cost = price;
double extended;
extended = quantity * price;
```
Example

\[ 5 + (4 - 3) / 5 - 2 * 3 \mod 4 \]
\[ = 5 + 1 / 5 - 2 * 3 \mod 4 \]
\[ = 5 + 0 - 2 * 3 \mod 4 \]
\[ = 5 + 0 - 6 \mod 4 \]
\[ = 5 + 0 - 2 \]
\[ = 3 \]

Other Arithmetic Operators

Each of long, float, and double come with 11 operators with the same symbols as int; i.e. the symbols are overloaded. Note:

- The int operators satisfy closure thru circular wrapping
- The / int operator always rounds toward 0 and leads to an exception if the divisor is zero
- The sign of % is the same as that of the dividend
- The real operators satisfy closure by adding Infinity and NaN. Hence, dividing by zero does not lead to exceptions
- \((a \times b) / c\) is not the same as \(a \times (b / c)\) for any type
- \((a + b) - c\) is not the same as \(a + (b - c)\) for real types

Mixed Types and Casting

- Promotion (aka widening conversion) is done automatically when needed
- May lead to loss of precision but the order of magnitude is preserved
- Demotion is not done automatically. Can be done manually thru a cast, e.g. \(\text{int } k = (\text{int}) 12.45\)
- Casting is risky...avoid it.
**Note:**
- The cast operator has a precedence that is higher than * but less than ++
- The = operator has the lowest precedence of all operators
- There are shorthand operators to combine assignment with an operator:
  \[ x \, \text{op}\, y \] is shorthand for \[ x = x \, \text{op}\, y \]
  Ex: \[ x += 1 \] is like \[ x = x + 1 \] or \[ x++ \]

### Relational Operators

They operate on numbers and produce boolean values.

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operator</th>
<th>Operands</th>
<th>Syntax</th>
<th>true if</th>
</tr>
</thead>
<tbody>
<tr>
<td>-7</td>
<td>&lt;</td>
<td>numeric</td>
<td>( x &lt; y )</td>
<td>( x ) is less than ( y )</td>
</tr>
<tr>
<td></td>
<td>&lt;=</td>
<td>numeric</td>
<td>( x \leq y )</td>
<td>( x ) is less than or equal to ( y )</td>
</tr>
<tr>
<td></td>
<td>&gt;</td>
<td>numeric</td>
<td>( x &gt; y )</td>
<td>( x ) is greater than ( y )</td>
</tr>
<tr>
<td></td>
<td>&gt;=</td>
<td>numeric</td>
<td>( x \geq y )</td>
<td>( x ) is greater than or equal to ( y )</td>
</tr>
<tr>
<td></td>
<td>instanceof</td>
<td>x instanceof C</td>
<td>is true if object reference ( x ) points at an instance of class ( C ) or a subclass of ( C ).</td>
<td></td>
</tr>
<tr>
<td>-8</td>
<td>==</td>
<td>any type</td>
<td>( x == y )</td>
<td>( x ) is equal to ( y )</td>
</tr>
<tr>
<td></td>
<td>!=</td>
<td>any type</td>
<td>( x != y )</td>
<td>( x ) is not equal to ( y )</td>
</tr>
</tbody>
</table>
Example

\[ 5 + (4 - 3) / 5 - 2 * 3 \mod 4 \]

Example

\[ 5 + (4 - 3) \mod 4 \]

Example

\[ 5 + 1 \mod 4 \]

Example

\[ 5 + 1 \mod 4 \]

Java/Roumani
**Example**

\[ 5 + (4 - 3) / 5 - 2 * 3 \mod 4 \]

\[ = 5 + 1 / 5 - 2 * 3 \mod 4 \]

\[ = 5 + 0 - 2 * 3 \mod 4 \]

\[ = 5 + 0 - 6 \mod 4 \]

\[ = 5 + 0 - 2 \]
Example

$$5 + (4 - 3) / 5 - 2 * 3 \mod 4$$

$$= 5 + 1 / 5 - 2 * 3 \mod 4$$

$$= 5 + 0 - 2 * 3 \mod 4$$

$$= 5 + 0 - 6 \mod 4$$

$$= 5 + 0 - 2$$

$$= 3$$

Example

$$5 + (4 - 3) / 5 - 2 * 3 \mod 4$$

$$= 5 + 1 / 5 - 2 * 3 \mod 4$$

$$= 5 + 0 - 2 * 3 \mod 4$$

$$= 5 + 0 - 6 \mod 4$$

$$= 5 + 0 - 2$$

$$= 5 - 2$$

Example

$$3 - 2 / 7 * 7.0$$

$$= 3 - 0 * 7.0$$

$$= 3 - 0.0 * 7.0$$

$$= 3 - 0.0$$

$$= 3.0 - 0.0$$

$$= 3.0$$
What does this do? Any errors?

```java
int iVar = 15;
long lVar = 2;
float fVar = 7.6f - iVar / lVar;
double dVar = 1L / lVar + fVar / lVar;
int result = 100 * dVar;
```

What does this do? Any errors?

```java
char letter = 'D';
letter = (char) (letter + 1);
System.out.println(letter);
int code = letter;
System.out.println(code);
int offset = letter - 'A';
System.out.println(offset);
```