### STRONGEST operators at top of table

<table>
<thead>
<tr>
<th>Family Name</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unary</td>
<td>- ! ~</td>
</tr>
<tr>
<td>Multiply</td>
<td>* / %</td>
</tr>
<tr>
<td>Add</td>
<td>+ -</td>
</tr>
<tr>
<td>Compare</td>
<td>&lt; &gt; &lt;= &gt;=</td>
</tr>
<tr>
<td>Equality</td>
<td>== !=</td>
</tr>
<tr>
<td>Logic And</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td>Logic Or</td>
<td></td>
</tr>
</tbody>
</table>

- **VERY strong operators**
- **Really WEAK operator**

**Operators DECREASE in strength toward the Bottom**
Here is a complex Java expression with parentheses and both unary and binary "-" operators.
As before, draw all the operands along the bottom of your workpage.

\[
(-a \times -b - \frac{c}{6}) \times (\frac{b}{c} \times 12 - e - f)
\]
You need to be able
To tell apart a **unary**
**negative** operator from
A **binary subtract** operator.

The **subtracts** have either a
**number**, or a **variable**, or a
close parenthesis `)` just to their
**LEFT**.

ALL the rest are unary
**negative operators**.
They have **VERY STRONG**
order of operation.
Begin Work as far INSIDE Parentheses as you can get.
For this example begin looking ONLY at the blue part.

\[ ( - a * -b - c / - 6 ) * ( b / c * 12 - e - f ) \]
Multiply is fairly strong, But WEAKER than unary -

( - a * -b - c / - 6 ) * ( b / c * 12 - e - f )

The unary - are the strongest operators in the selected region.
One of the strongest operators
In the blue selected region
Is the circled unary negative.

\[( - a \times b - c / - 6 ) \times ( b / c \times 12 - e - f )\]
( -a * b - c / -6 ) * ( b / c * 12 - e - f )

Enclose these parts in a rectangle:
1. The unary - operator
2. Its RIGHT operand a.

NOTE: because the operator Is unary - it ONLY has a RIGHT operand and NO left operand.
\[( - a \times -b - c / - 6 ) \times ( b / c \times 12 - e - f )\]

Draw the branch of the tree corresponding to the enclosed part of the formula. Note that the unary - oval has ONLY ONE leg beneath it. This contrasts with a binary subtract oval (done later) which will have two legs below it.
Within the original selected region, there is another unary - operator and so it is (as usual) one of the strongest. This slide shows its enclosing rectangle AND the tree branch for it.
\(( -a \times -b \div -6 ) \times ( b \div c \times 12 - e - f )\)

Within the original selected region, there is yet another unary - operator and so it is (as usual) one of the strongest. Notice the - with \(-c\) is a **subtract**.

This slide shows its enclosing rectangle AND the tree branch for it.
Now there were only three operators left unboxed in the selected blue region. The subtract - is pretty weak. The safest choice among the strongest operators * and / is the LEFTMOST one. Hence the * is chosen for the next tree branch.

When enclosing it in a box, be careful to **draw around** the two smaller boxes NEXT to the operator.
Now there were only two operators left unboxed in the selected blue region. The strongest operator is the divide `/`. So it is enclosed and drawn next.
Only the subtract - operator was left unboxed in the selected blue region. When only ONE operator is left within The parenthesized area, the parentheses are used as PART OF the box.

Note that the - oval has TWO legs Because the subtract is BINARY.
The remaining slides will proceed more quickly.
If this confuses you, go back to the preceding slides and pay better attention.
Note 1) Work INSIDE parentheses FIRST.
Note 2) Of equally strong operators - choose the leftmost first.
\[
\left( \frac{-a \times -b}{c / 6} \right) \times \left( \frac{b}{c} \times 12 - e - f \right)
\]
\((-a \ast -b - \frac{c}{-6}) \ast (\frac{b}{c} \ast 12 - e - f)\)

Note that the left subtract - is done before the equally strong right subtract - Because the leftmost is done first.
Complex Java Expression
Root gets stub
Note when numbering operators, to always increase down any leg.
Entablement of Tree

Note unary operators have only one leg.