

CS2110: Type boolean

The values of primitive type boolean are **true** and **false**. The operators are:

! (meaning negation, of complement. **!true** is **false** and **!false** is **true**)

&& (and, or conjunction. $b \ \&\& \ c$ is **true** iff both b and c are **true**; otherwise it is **false**)

|| (or, or disjunction. $b \ || \ c$ is **true** if b or c (or both) is **true**; otherwise it is **false**)

Operator precedences

Operator **!** has highest precedence, then **&&**, and finally **||**. There is no universal tradition for the relative precedences of **&&** and **||**, and we recommend always using parentheses when they appear next to each other in an expression, as in

$(x < 5 \ \&\& \ y == 5) \ || \ z == 2$

Short circuit evaluation

Operations $b \ \&\& \ c$ and $b \ || \ c$ are evaluated left-to-right using *short-circuit evaluation*. That means that as soon as the answer is known, evaluation stops. There are two cases to explain:

false **&&** c evaluation does *not evaluate* c ; it simply yields the value **false**
true **||** c evaluation *does not evaluate* c ; it simply yields the value **true**

Short-circuit evaluation helps to shorten and simplify code. For example, the following expression is true iff j is not 0 and k / j is most 50; division by 0 does not occur if j is 0:

$j != 0 \ \&\& \ k / j <= 50$

Expressions with boolean values

Relational expressions $d == e$, $d != e$, $d < e$, $d <= e$, $d > e$, and $d >= e$ all evaluate to a boolean value —either **true** or **false**— and can thus be used in boolean expressions.

Operators **&** and **|**

Operators **&** and **|** can also be used but we recommend against their use as boolean operations. They are *bitwise* operations, and we do not discuss them. Short-circuit evaluation is not used for them.

Comparison with other languages

Some languages, e.g. C, use integers as booleans; 0 represents **false** and any other integer represents **true**. This does not work in Java.