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. \(b \land c\) is true iff both \(b\) and \(c\) are true; otherwise it is false)
- || (or, or disjunction. \(b \lor c\) 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

\[(x < 5 \land y == 5) \lor z == 2\]

**Short circuit evaluation**

Operations \(b \land c\) and \(b \lor c\) 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:

\[j \neq 0 \land k/j \leq 50\]

**Expressions with boolean values**

Relational expressions \(d == e, d \neq e, d < e, d <= e, d > e, d >= e\) 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.

**The marks of a boolean tyro**

A tyro is a beginner, a novice. It is pronounced tīrō, like gyro in the word gyroscope. It has nothing to do with gyro, that Greek fast food delicacy, wrapped in pita bread.

There’s nothing wrong with being a boolean tyro. We were all boolean tyros once. But tyros sometimes don’t want other to know they are tyros. If you don’t, stay away from the following two marks of a boolean tyro. First, if you have a boolean variable isFemale, don’t write:

\[\text{if (isFemale == true)}...\]

Instead write:

\[\text{if (isFemale)}...\]

You see, the two expressions isFemale \(==\) true and isFemale evaluate to the same value; they are equal. In the same way, instead of isFemale == false, write !isFemale.

The second mark of a boolean tyro is the use of if-statements like the following:

\[\text{if (isFemale)}...\]
Type boolean

    if (isFemale) return true;
    else return false;

This statement returns true if isFemale is true and false if it is not. So why not just write:

    return isFemale;

Similarly, instead of:

    if (atHome || atWork) b = true;
    else b = false;

write:  b = atHome || atWork;