The overriding (or bottom-up) rule

Consider the object of class \(S\) on the right. Class \(S\) was declared as a subclass of class \(C\). We show some (but obviously not all) of the methods in the three partitions.

Variables \(ob\), \(c\), and \(s\) were declared like this:

\[
\begin{align*}
S \ s &= \textbf{new} \ S(\ldots) \\
C \ c &= s \\
\text{Object} \ ob &= s
\end{align*}
\]

Consider three possible calls on functions \(\text{toString}\) and on \(\text{equals}\):

\[
\begin{align*}
\text{s.toString()} & \quad \text{s.equals(some object)} \\
\text{c.toString()} & \quad \text{c.equals(some object)} \\
\text{ob.toString()} & \quad \text{ob.equals(some object)}
\end{align*}
\]

By the compile-time reference rule, all these calls are syntactically legal and will be compiled. We ask this question: At runtime, which method \(\text{toString}\) will be called, the one in partition \(\text{Object}\) or the one in partition \(\text{S}\)? The answer is given by this rule:

**Overriding or bottom-up rule:**

Let \(p.m(\ldots)\) be a legal call on method \(m(\ldots)\). To determine which method is called, start at the bottom of object \(p\) and search upward until the appropriate method \(m\) is found.

Applying this rule, in all three cases, method \(\text{toString}\) in partition \(\text{S}\) will be called. Similarly, in all three cases, function \(\text{equals}\) in partition \(\text{C}\) will be called.

This is an important point: at runtime, in determining which method is called when \(\text{ob.toString()}\) is called, the type of variable \(\text{ob}\) does not matter. What only matters is the object to which \(\text{ob}\) points.

**Overriding or bottom-up rule for variables**

The same rule applies for references to fields, like \(s.f\) (if there was a field \(f\)). But remember, we do not consider redeclaring fields. It can be done in Java, but we do not consider it and never do it. Thus, the object will have at most one field \(f\).

**Use of “super.”**

To the right is method \(\text{toString}\) in partition \(\text{S}\). It returns the string “this is object \(S@2\)”. The insertion of “\(\text{super.}\)” changes the bottom-up rule to start at the partition above partition \(\text{S}\), so that method \(\text{toString}\) in partition \(\text{Object}\) is called. You know that in this case it returns “\(S@2\)”.

Thus, we have the “\(\text{super.}\)” rule:

In any method \(m\) in a partition named \(p\), the call \(\text{super.}m(\ldots)\) calls the method \(m\) found by using the bottom-up rule starting at the partition above partition \(p\).