Executing method calls

Introduction

It is important that you understand how a method call is executed, for several reasons. This knowledge will give you a better understanding of how a program is executed. You may have to execute a call yourself, by hand, in order to debug a program. Knowledge of how a call is executed is useful in analyzing the time and space requirements of a program. Finally, this knowledge will also be extremely valuable when you learn about recursion. So let's begin.

The frame for a call

Here is an object of class `Employee`. We show only one field and one method, function `num`. We have drawn the method body here, too, in order to make clear what is happening. This method yields the number of times its parameter `c` occurs in the name of the employee. We have labeled the statements with numbers so that we can refer to them later. We will illustrate using the call `v.num('e')`.

When a call is executed, a *frame* for the call is created. It contains information that is needed in order to execute the call. How it is stored in a computer doesn't matter; here, we describe how we draw frames when we have to draw them. So this is what goes into the frame, which we draw as a box.

First, in the upper left is a box that contains the name of the method and the *program counter*. The program counter will be used during execution to indicate which statement of the method body to execute next. It is initially set to 1, since the first statement is numbered 1.

Second, the *scope* box indicates where the method that is being executed resides. In this case, it is in object `a0`, so the scope box contains `a0`. If this were a static method of class `Employee`, the scope box would contain the name `Employee`, since the method resides in file drawer `Employee`.

Third, the frame contains all the parameters of the method —written as variables, since that is what the parameters are.

Fourth and finally, the frame contains the local variables of the method. The method body has two local variables, `s` and `k`, so they appear in the frame.

Each item in the frame is there for a purpose. The name tells us which method it is, and the scope box tells us where the method resides. The program counter is used when executing the method body to keep track of which statement to execute next. And the parameters and local variables are used during execution of the method body.

Steps in executing a method call

We now give the steps in executing a method call, using the call `v.num('e')`.

1. *The first step is to draw the frame for the call.* Put in the method name and program counter; the scope box, indicating where the method resides; the parameters; and the local variables. The variables are not initialized.
2. *The second step is to evaluate the arguments and store their values in the parameters*. In this case, the argument value `'e'` is stored in parameter `c`.
3. *The third step is to execute the method body.* As this is done, update the program counter as statements are executed, so that you can keep track of execution. When a reference to a variable or method occurs, look first in the frame for the call. If it is not found there, then look, in an inside-out manner, starting at the place given by the scope box. In this case, look in object `a0`.
4. The fourth step is to erase the frame for the call. If the computer is doing this, the storage used to contain the items in the frame are given back to the system, to be used for another purpose. If this is a function call, the value of the return expression is used as the value of the function call.

And that is all there is to executing a method call: draw the frame, assign argument values to parameters, execute the method body, and erase the frame. Please memorize this sequence of steps and practice carrying them out yourself, in our self-help exercises.

As said earlier, knowledge of execution for a method call can help answer questions. For example, suppose you are asked when local variables are created? That's easy, you say, relying on the 4 steps. They are created before execution of the method body, when the frame for the call is created.