The Call Stack
David Gries and Scott Wehrwein

When a Java program is being executed (by you or a computer) a call stack is maintained. This call stack contains a frame for each method call that has not been completed.

We show how to execute a method call using function p as an example.

Assume that a method is being executed and it contains the assignment \( z = p(1+4) \). A frame for this method is at the top of the call stack — it contains local variable \( z \). The function call \( p(1+4) \) is to be carried out or evaluated.

**Algorithm**

We now state the algorithm for carrying out a general method call, using this method \( p(1+4) \) call to illustrate.

1. **Push a frame for the call onto the call stack.**
2. **Evaluate the arguments of the call (from left to right) and store their values in the parameters.** In this case, \( 1+4 \) is evaluated and its value, 5, is stored in parameter \( n \).
3. **Execute the method body, using the frame for the call to access parameters and local variables.**
   - We execute the assignment \( k = n+1 \). The value of the expression is 6, so we store 6 in \( n \).
   - Execution of the return statement ends execution of the body.
   - The value 6 is to be returned.
4. **Pop the frame for the call from the call stack. If this is a function call (and it is), push the value to be returned onto the call stack.**

(Note that that value will be then be popped from the stack and used as the value of the call.)

That’s it! Memorize this 4-step algorithm to execute a method call.