CS100J 7 February 2008

Congratulations!! You now know the basics of OO (object-orientation). There are more odds and ends, which we will be discussing, but the basics have been covered. We now turn to:

Discussion of Methods: Executing method calls. If-statements. The return statement in a function. Local variables.

For this and next lecture: **Read section 2.3 but NOT 2.3.8!!!!**
Do the self-review exercises in 2.3.4

Oxymoron: a combination for epigrammatic effect of contradictory or incongruous words (as *cruel kindness, laborious idleness*)

airline food	State worker
military intelligence	peace force
Microsoft Works	computer security
sanitary landfill	tight slacks
religious tolerance	business ethics

Method body: sequence of statements enclosed in { } (interspersed with declarations) to execute, in the order in which they appear

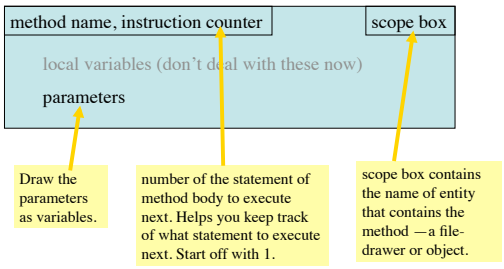
```
/** Constructor: a chapter with title t,
    number n, and previous chapter null.*/
public Chapter(String t, int n) {
  title= t;
  number= n;
  previous= null;
}
```

Execute the three assignments in the order in which they appear. Same scheme is used when a cook uses a recipe.

We explain exactly how a method call is executed so that you can understand how parameters and arguments work.

The frame (the box) for a method call

Remember: Every method is in a folder (object) or in a file-drawer.



Draw the parameters as variables.

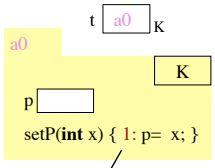
number of the statement of method body to execute next. Helps you keep track of what statement to execute next. Start off with 1.

scope box contains the name of entity that contains the method —a file-drawer or object.

Execution of a method call.

Execute the call t.setP(7);

1. Draw a frame for the call.
2. Assign the value of the argument to the parameter (in the frame).
3. Execute the method body. (Look for variables in the frame; if not there, look in the place given by the scope box.)
4. Erase the frame for the call.

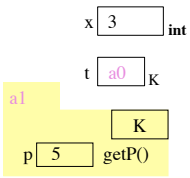


The first (and only) statement is #1.
Procedure setP has one parameter: x.
The call has one argument: expression 7.


```
public class K {
  int p ;
  public int getP() {
     return p;
  }
}
```

1:

x= t.getP() + 1;



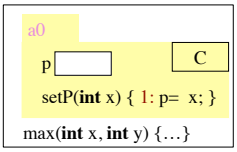
Execute a function call

1. Draw a frame for the call.
2. Assign the value of the argument to the parameter (in the frame).
3. Execute the method body. (Look for variables in the frame; if not there, look in the place given by the scope box.)
4. Erase the frame for the call. (and, if it is a function use the value of the return-statement expression as the function call value

Local variable: a variable declared within a method body

```
/** = x + y */
public static int max(int x, int y) {
  1  int t;
  2  t= x + y;
  3  return t;
}
```

1. Draw a frame for the call.
2. Assign arg values to pars.
3. Execute the method body.
4. Erase frame for call. (If it is a function use value of return-statement expr. as function call value



C's file drawer

Evaluate this call.

C.max(5, 6);

This time, when you create the frame for the call, draw parameters **and** local variables:

```
/* swap x, y to put larger
   in y */
if (x < y) {
   int t;
   t= x;
   x= y;
   y= t;
}
```

if statement

Syntax:
if (<boolean expression)
    <statement>

Execution: if the <boolean expression> is true, then execute the <statement>

```
/* Put smaller of x, y in  z */
if (x < y) {
   z= x;
}
else {
   z= y;
}
```

if-else statement

Syntax:
if (<boolean expression)
    <statement1>
else <statement2>

Execution: if the boolean expression is true, then execute <statement1>; otherwise, execute <statement2>

A function produces a result

```
/** = smallest of b, c, d */
public static int smallest(int b, int, c, int d) {
    if (b <= c  && b <= d) {
        return b;
    }
    // { The smallest is either c or d }
    if (c <= d) {
        return c;
    }
    // { the smallest is d }
    return d;
}
```

Execution of statement

return <expr> ;

terminates execution of the procedure body and yields the value of <expr> as result of function call

Assertion

Execution of function body must end by executing a return statement.

Syntax of procedure/function/constructor and calls

public <result type> <name> (<parameter declarations>) { … } **function**
public void <name> (<parameter declarations>) { … } **procedure**
public <class-name> (<parameter declarations>) { … } **constructor**

Exec. of a function body *must* terminate by executing a statement "**return** <exp> ;", where the <exp> has the <result type>.

Exec. of a proc body *may* terminate by executing statement "**return** ;"

Exec. of a constructor body initializes a new object of class <class-name>.

<name> (<arguments>) **function call**
<name> (<arguments>) ; **procedure call**
new <class-name> (<arguments>) **constructor call**

<arguments>: <expression>, <expression>, …, <expression>

Local variable: a variable declared in a method body

Scope of local variable: the sequence of statements following it.

```
/** = the max of x and y */
public static int max(int x, int y) {
    // Swap x and y to put the max in x
    if (x < y) {
        int temp;
        temp= x;
        x= y;
        y= temp;
    }

    return x;
}
```

scope of temp

You can't use temp down here

This is an error.

Local variable: a variable declared in a method body

Scope of local variable: the sequence of statements following it.

```
/** s contains a name in the form exemplified by "David  Gries".
    Return the corresponding String "Gries, David".
    There may be 1 or more blanks between the names. */
public static String switchFormat(String s) {
    // Store the first name in variable f and remove f from s
    int k;      // Index of the first blank in s
    k= s.indexOf(' ');
    String f;  // The first name in s.
    f= s.substring(0, k);
    s= s.substring(k);

    // Remove the blanks from s
    s= s.trim();

    return s + ", " + f;
}
```

declaration
assignment

scope of k

scope of f

Numbering of characters in a String:
012345
"abcdef"