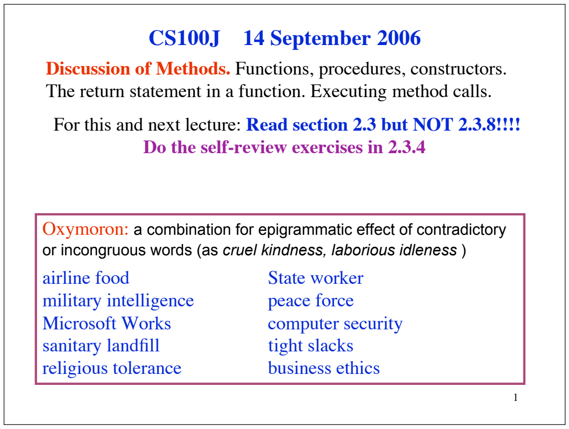


CS100J 14 September 2006

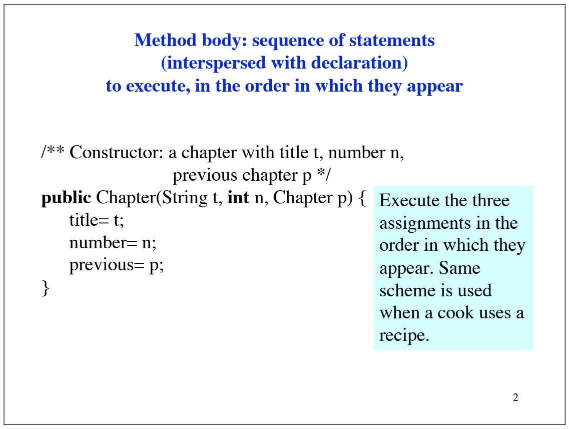
Discussion of Methods. Functions, procedures, constructors. The return statement in a function. Executing method calls.

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
military intelligence
Microsoft Works
sanitary landfill
religious tolerance

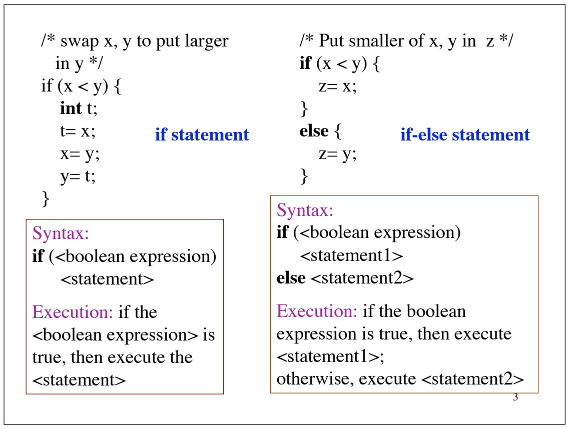
State worker
peace force
computer security
tight slacks
business ethics



Method body: sequence of statements (interspersed with declaration) to execute, in the order in which they appear

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

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



```
/* 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>

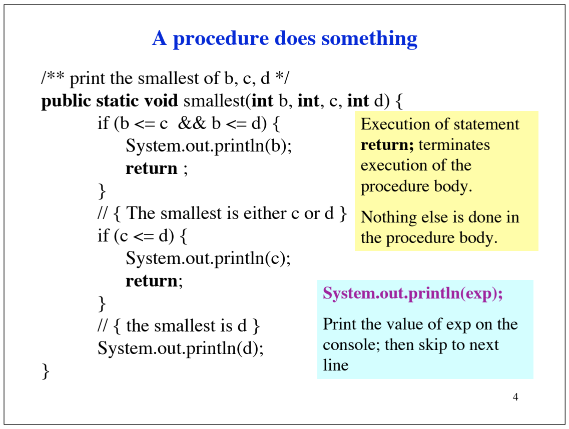
Execution: if the <boolean expression> is true, then execute the <statement>

```
/* 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 procedure does something

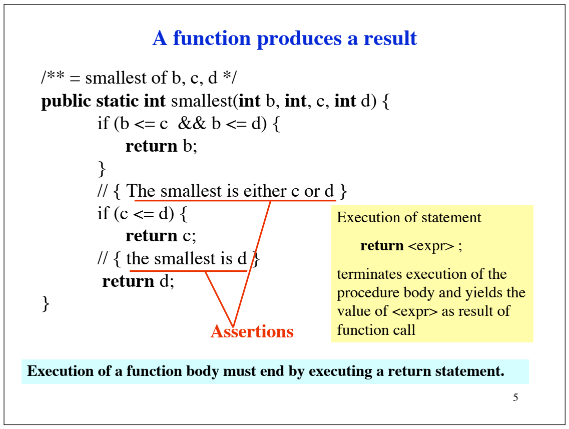
```
/** print the smallest of b, c, d */
public static void smallest(int b, int, c, int d) {
        if (b <= c  && b <= d) {
            System.out.println(b);
            return ;
        }
        // { The smallest is either c or d }
        if (c <= d) {
            System.out.println(c);
      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

Execution of statement **return;** terminates execution of the procedure body.

Nothing else is done in the procedure body.

System.out.println(exp);

Print the value of exp on the console; then skip to next line



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;
}
```

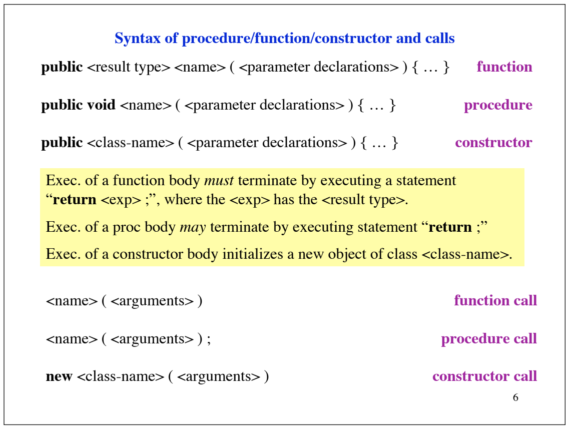
Assertions

Execution of statement

return <expr> ;

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

Execution of a 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>.

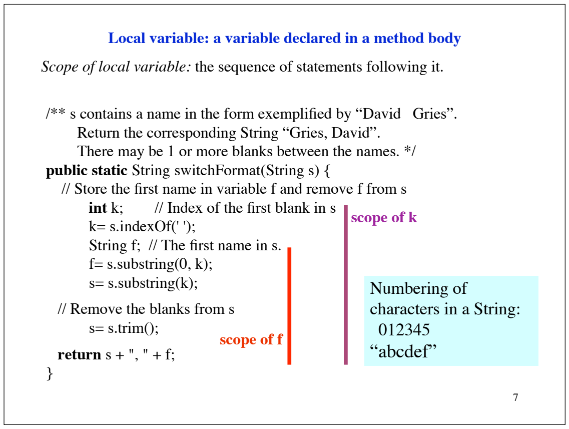
Exec. of a proc body *may* terminate by executing statement "**return** ;"

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**



Local variable: a variable declared in a method body

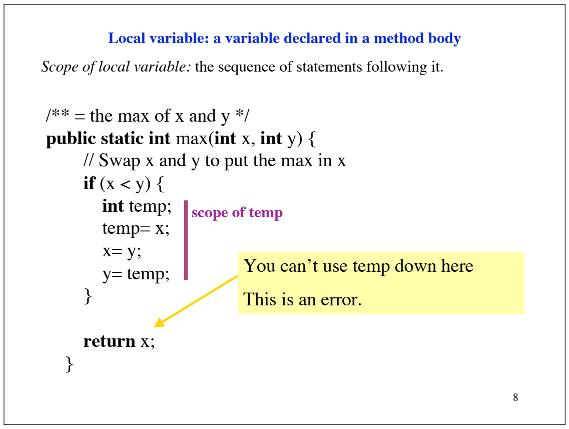
Scope of local variable: the sequence of statements following it.

```
/** 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;
}
```

scope of k

scope of f

Numbering of characters in a String:
 012345
"abcdef"



Local variable: a variable declared in a method body

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

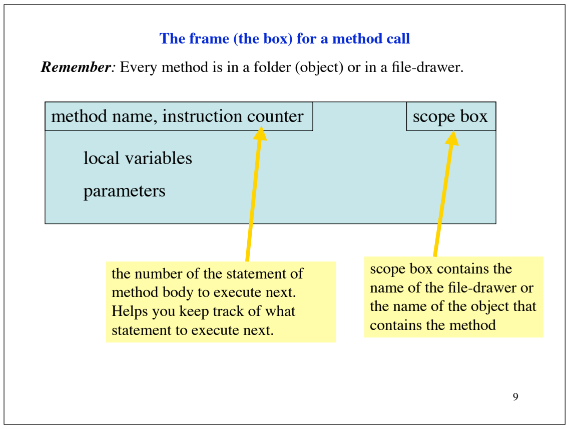
```
/** = the max of x and y */
public static int max(int x, int y) {
      // Swap x and y to put the max in x
      if (x < y) {
         int temp;
         temp= x;
         x= y;
         y= temp;
      }

      return x;
   }
```

scope of temp

You can't use temp down here

This is an error.

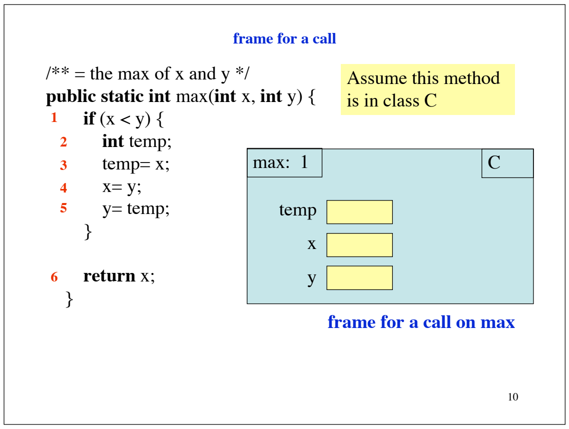


The frame (the box) for a method call

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

the number of the statement of method body to execute next. Helps you keep track of what statement to execute next.

scope box contains the name of the file-drawer or the name of the object that contains the method



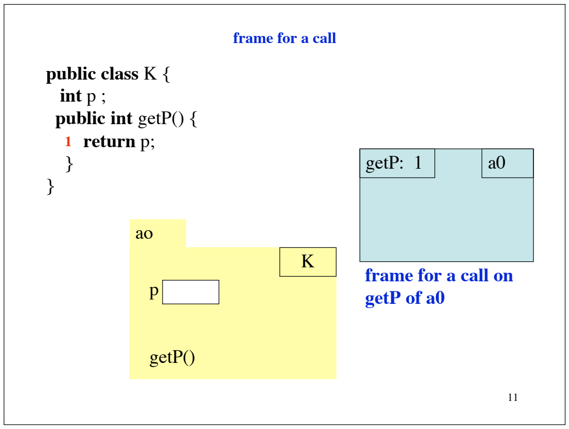
frame for a call

```
/** = the max of x and y */
public static int max(int x, int y) {
 1    if (x < y) {
  2      int temp;
  3      temp= x;
  4      x= y;
  5      y= temp;
      }

 6    return x;
   }
```

Assume this method is in class C

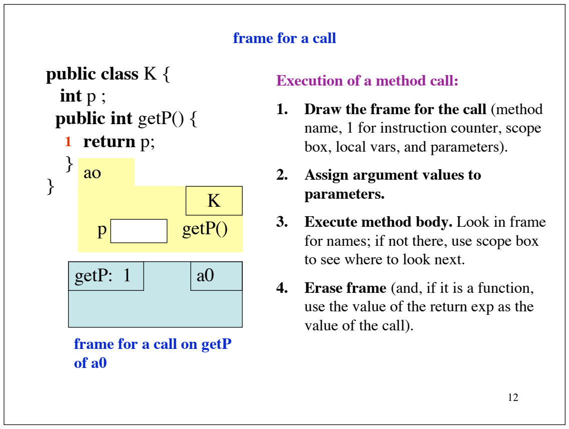
frame for a call on max



frame for a call

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

frame for a call on getP of a0



frame for a call

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

frame for a call on getP of a0

Execution of a method call:

1. **Draw the frame for the call** (method name, 1 for instruction counter, scope box, local vars, and parameters).
2. **Assign argument values to parameters.**
3. **Execute method body.** Look in frame for names; if not there, use scope box to see where to look next.
4. **Erase frame** (and, if it is a function, use the value of the return exp as the value of the call).