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

For this and next lecture: Read section 2, 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

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:

Please sit next to someone. We will do some work in pairs today.

Prelim (preliminary exam) I
Tuesday, September 30, 7:30-9:00PM

If you have a conflict, you MUST email Maria Witlox mwitlox@cs.cornell.edu by Friday night (tomorrow night) so that we know how many people have conflicts.

Give Maria: Name, NetId. What the conflict is.

Tuesday, we will give you a handout explaining what is on prelim I. But you can see it now, as well as previous prelims, on the course website. Click on exams in the left column.

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.

Remember:
Every method is in a folder (object) or in a file-drawer.
method name, instruction counter:
scope box:
local variables (don’t deal with these now):
parameters:

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

Execution of a method 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.

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

```java
/** = x + y */
public static int sum(int x, int y) {
    int t;
    t = x + y;
    return t;
}
```

C’s file drawer

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).

A function produces a result

```java
/** = 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 function body must end by executing a return statement.

### Syntax of procedure/function/constructor and calls

```java
public <result type> <name> ( <parameter declarations> ) { … }
public <name> ( <parameter declarations> ) { … }
public <class-name> ( <parameter declarations> ) { … }
```

### 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>.

### Execution of statement

- `return <exp>;` terminates execution of the procedure body and yields the value of <exp> as result of function call.

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

```java
/** = 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;  // You can’t use temp down here
        temp = x;  // This is an error.
        x = y;
        y = temp;
    } return x;
}
```

Local variable: a variable declared in a method body

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

```java
/* = x + y */
public static int sum(int x, int y) {
    int t;
    t = x + y;
    return t;
}
```