About CS 1110 prelim (preliminary exam) I
Thursday, 19 Feb 7:30-9:00PM, in Baker Lab 200

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. If the conflict is another prelim, indicate the course and name and email of that course's instructor.

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.

1

CS1110 5 February 2009

Congratulations!! You now know the basics of OO (object-orientation).

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

For this and next lecture: **Read chapter 2, but NOT 2.3.8!!!!**
Do the self-review exercises in 2.3.4

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

2

We write programs in order to do things.
Methods are the key "doers".

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

parameters: t and n

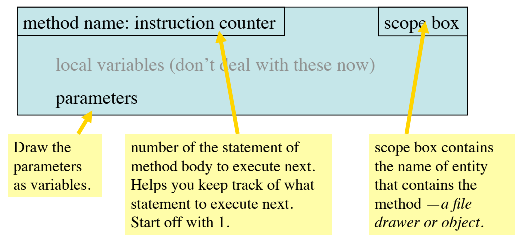
Within the body (between the {}s), execute the assignments in the order in which they appear. ("Follow the recipe".)

But how is a method call executed ---
How do parameters and arguments work?

3

The frame (the box) for a method call

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
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*.

(see handout): The scope box for x.getAmt() should have "a0".
For Account.getBankNameNow(), it should be "Account".

4

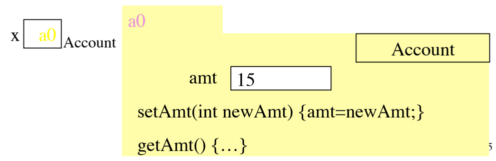
To execute the call x.setAmt(50);

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.



5

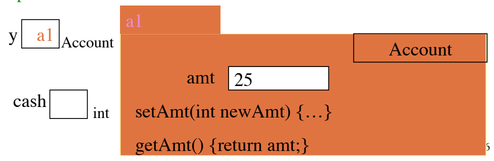
To execute the call cash=y.getAmt();

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; use the value of the return-statement expression as the function-call value.



6

Draw the frame for Account.getBankNameNow();

Note the *local* variable **d** declared within the method body.
(This method happens to have no params.)

7

```
/* 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>

8

Idiom: if statements and multiple return staements

```
/** = 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.

9

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>

10

Scope of local variable: the sequence of statements following it within the containing "block".

```
/** = 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.

11

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
declaration    int k;       // Index of the first blank in s
assignment     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"

12