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.

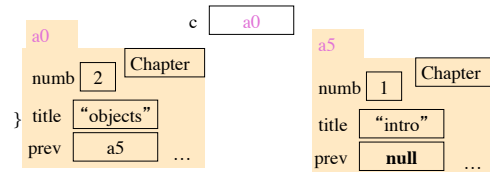For this and next lecture: **Read chapter 2, but NOT 2.3.8!!!!**
Do the self-review exercises in 2.3.4

Sit next to someone. Today, we do some work in pairs. Get out a blank sheet of paper.

The last slide concerns local variables –variables declared within a method body. We don't have time to discuss them. You are responsible for knowing about local variables. Read pp. 76-78 (sec. 2.3.7).

Take advantage: see videos of the 11:15 lecture for CS1110 on www.VideoNote.com. Log in with your Cornell netid

1

/** An instance keeps information about a book chapter */
public class Chapter {
    // class invariant: meanings of fields and constraints on them
    private int number; // the chapter number, in range 0..100
    private String title; // chapter title
    private Chapter prev; // instance for the previous chapter
                                // (null if no previous chapter)

c   a0

a0

numb	2
title	"objects"
prev	a5

Chapter

}

a5

numb	1
title	"intro"
prev	**null**

Chapter

…

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;
}

declaration of parameter t

parameters: t and n

Within the body (between { }) is the sequence of statements to execute when the method is called. ("Follow the recipe".)
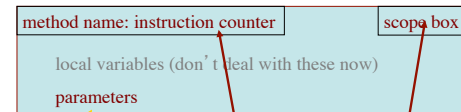
Memorize: a parameter is a variable that is declared within the parentheses of a method header.

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

3

The frame (the box) for a method call

Draw template on a piece of paper

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. **Start with 1.** Helps you keep track of what statement to execute next.

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

The scope box contains:
For an instance method, the name of the object in which it resides
For a static method, the name of the class in which it is defined 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.

public void setAmt(**int** newAmt) {
    amt= newAmt;
}

x   a0   Account

a0

Account

amt   15

setAmt(int newAmt) {amt= newAmt;}

getAmt() {…}

5
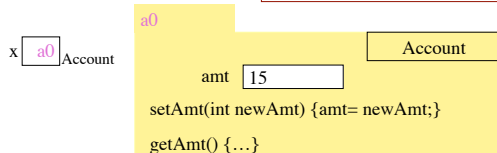
To execute the call **cash= y.getAmt();**

1. Draw frame for call.
2. Assign value of argument to parameters (in the frame).

3. Execute method body. (Look for variables in the frame; if not there, look in the place given by the scope box.)
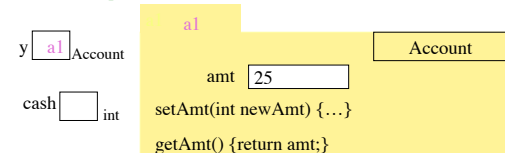
4. Erase the frame for call; use value of return-statement expression as function-call value.

public int getAmt() {
    return amt;
}

y   a1   Account

cash   int

a1

Account

amt   25

setAmt(int newAmt) {…}

getAmt() {return amt;}

5

1

new Chapter("Intro", 1)

1. Draw a frame for the call.

2. Assign arg values to pars.

3. Execute the method body.

4. Erase the frame for the call.

a8

```
Chapter(String t, int n) {
    String d;
    1: d= t;
    2: title= d;
    3: number= n;
    4: previous= null;
}
```

Chapter

title   null

number 0

previous null

Chapter(String t, int n) { … }

Note *local* variable d declared within method body. It should be drawn in frame for call.

7

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

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 {                if-else statement
    z= y;
}
```

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 fields of a new object in order to make the class invariant true.

<name> (<arguments>)   **function call**

<name> (<arguments>) ;   **procedure call**

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

<arguments>: <expression>, <expression>, …, <expression>   10

Scope of local variable is the places where it can be used. The scope is 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;       scope of temp
        temp= x;
        x= y;
        y= temp;
    }

    return x;
}
```

You can't use temp down here

This is an error.

11

2