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

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

Sit next to someone.

Today, we do some work in pairs.

Get out a blank sheet of paper.

Get out Iclicker.

Conflict with Thursday Oct. 7th 7:30-9pm prelim? Email mwitlox@cs.cornell.edu with netid, other course name, other course instructor's email address BY NEXT THURSDAY.

The last slide: local variables --variables declared within a method body. No time to discuss them. You are responsible for local variables. Read pp. 76-78 (sec. 2.3.7).

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;
    numb= n;
    previous= null;
}

We refer to variables that are declared in a method body as local variables. Local variables are automatically initialized with default values (0, empty string, null)

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?

Within the body (between { }) is the sequence of statements to execute when the method is called. ("Follow the recipe")

The frame (the box) for a method call

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

Draw template on a piece of paper

The scope box contains:

For an instance method, name of object in which it resides
For a static method, name of class in which it is defined

The frame (the box) for a method call

```java
/** Change the previous chapter title to t. Precondition: previous chapter not null. */

public void changePrevTitle(String t) {
    prev.title= t;
}
```

A. It will work
B. It won't compile --it has a syntax error
C. It will compile --but it won't run right

A2 will take you 10-15 minutes.

Due on Thursday, in class.

Do it this weekend, not two minutes before class on Thursday.
**Account**

`setAmt(int newAmt)`

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.

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

**To execute the call**

`x.setAmt(50);`

1. Draw a frame for the call.
2. Assign the values of arguments to parameters (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.

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

**Chapter(String t, int n)**

1. Draw a frame for the call.
2. Assign arg values to pars (in frame).
3. Execute the method body.
4. Erase the frame for the call.

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

**Note**

Local variable `d` declared within method body. It should be drawn in frame for call.

**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.

```java
new Chapter("Intro", 1)
```

**To execute the call**

`new Chapter(Chapter("Intro", 1));`

1. Draw a frame for the call.
2. Assign arg values to pars (in 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.

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

**Note**

Local variable `d` declared within method body. It should be drawn in frame for call.
/* Put smaller of x, y in z */
if (x < y) {
    z = x;
} else {
    z = y;
}

if statement

/* swap x, y to put larger in y */
if (x > y) {
    int t;
    t = x;  // if statement
    if (x <= c && b <= d) {
        return b;
    } else {
        return c;
    }
    // (The smallest is either c or d)
    return d;
}
// (The smallest is d)
return d;

Idiom: if statements and multiple return statements

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

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