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

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).

A note on assignment A1

```java
/** Have this Organism eat victim. 
   * Precondition: victim is not null and this Organism is alive ... */
public void eat(Organism victim)
```

Strategy: To gain understanding, develop test cases.
Substrategy: Draw the objects, filling in (only) the relevant components, and determine how they should be changed.

```
b.eat(v);
a1

Organism
alive true
mercury 9
eatenBy null
numVictims 3
eat(victim){ … }  
```

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

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

Within the body (between { }), execute the assignments in the order in which they appear. (“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?

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
```

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.

To execute the call x.setAmt(50);

```
x = new Account; 
setAmt(15); 
getAmt();
```
To execute the call  
\texttt{cash} \leftarrow \texttt{y.getAmt}() ;
1. Draw frame for call.
2. Assign value of argument to 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.

\begin{array}{|c|c|}
\hline
\texttt{y} & \texttt{Account} \\
\hline
\texttt{amt} & \texttt{25} \\
\hline 
\end{array}

\texttt{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.

\begin{array}{|c|c|c|c|c|}
\hline
\texttt{title} & \texttt{null} & \texttt{Chapter} \\
\hline
\texttt{number} & \texttt{0} & \texttt{previous} & \texttt{null} \\
\hline 
\end{array}

// { The smallest is either c or d }
return d;

To execute the call  
\texttt{cash} \leftarrow \texttt{y.getAmt}() ;
1. Draw frame for call.
2. Assign value of argument to 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.

\begin{array}{|c|c|}
\hline
\texttt{y} & \texttt{Account} \\
\hline
\texttt{amt} & \texttt{25} \\
\hline 
\end{array}

Idiom: if statements and multiple return statements

\begin{array}{|c|c|c|c|c|}
\hline
\texttt{int} & \texttt{t} & \texttt{temp} & \texttt{x} & \texttt{y} \\
\hline 
\end{array}

new Chapter("Intro", 1)