Let’s write some code.

- Write a procedural program for calculating the average salaries for each department (Accounting, HR, IT) in a company.

- Assume I have the following input file:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Joe, Smith, HR, $40,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>George, Jetson, Accounting, $50,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Bill, Gannon, HR, $45,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Fred, Flintstone, IT, $65,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Barney, Bubble, Accounting, $55,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>James, Rick, IT, $70,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>John, Doe, HR, $30,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Jane, Doe, IT, $85,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Marge, Simpson, HR, $35,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>Alice, Krampus, Accounting, $20,000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OBJECT ORIENTED CONCEPTS

• In some cases we’ll learn the concepts prior to learning the terminology.

• Other times we’ll hear the terms, before we know what they mean.

• In either case, what we call the concepts are just as important as the concepts themselves.

• If you want to talk “OO” with others, you will need to learn the vocabulary.

I am Dungeon Master, your guide in the world of Object-Orienting Programming

Sometimes by looking back you can see a clearer path through what lies ahead.

Just follow that path. But beware. You must never touch - the beauty - that breathes the beast.

The right road is not the left.

Beware, for only beauty can defeat the eye of the beholder.
What makes a language an **Object-Oriented**?

- The goal of OOP is **abstraction**.
- In general you will see three big terms used:
  - Encapsulation
  - Inheritance
  - Polymorphism *(this has multiple meanings)*
- Weisfeld adds an additional one:
  - Composition

**WHAT IS PROCEDURAL PROGRAMMING?**

- Procedural Programming consists of writing **procedures** that take in some data as **input** and provide **output**.
- Data is **distinct** from the process. It has no relation to the procedure.
- Programs **process** data as defined by the **procedure**.
- Likewise the **functions** written in the program have no direct **relationship** with each other.
- Programmer is all knowing, beginning to end.
PROCESSING DATA

• My first job after college was programming in COBOL.

• The department I worked in was called Administrative Data Processing (ADP).

• I wrote very long programs that would process data.

HOW IS OOP DIFFERENT?

• In Object Oriented Programming (OOP) the data and the functions that operate on that data are bundled together into an object.

  – Objects are self-contained entities that maintain their own data.

  – A program uses objects that interact to solve a problem.

  – Abstraction hides the unnecessary details from the programmer.
WHAT IS NOT OO PROGRAMMING?

• You can write *procedural code* in any language, including Java.

• Simply using *functions* or *classes* in your program does *not* make it Object Oriented.

PROCEDURAL DATA DESIGN.

• In addition to the parameters, all the functions *share* access to the *global* variables.

• If data is *changed* by FunctionA, the other functions use the *new* value.
OBJECT ORIENTED DATA DESIGN.

- The data and related functions are *encapsulated* together.
- The *object* is a self contained *entity*, and other objects cannot inadvertently *change* it’s data.

LEARNING OO

“The most difficult problem in teaching *object-oriented* programming is getting the learner to give up the *global knowledge* of control that is possible with procedural programs, and rely on the *local knowledge* of objects to accomplish their tasks.”

(Beck and Cunningham OOPSLA ’89)
WHAT DID YOU COME UP WITH?

• A *procedural* approach may take the form of:

  • Read the file, line by line and load the data into variables or perhaps an array or list.

  • Once the data is loaded calculate the average salary for each department, by summing up and dividing by the number of individuals in the department.

  • Print out or display the resulting calculations.

WHAT WOULD AN OBJECT ORIENTED APPROACH LOOK LIKE?

• Use a *FileLoader* object to load the data and create objects (Departments and Employees).

• Each *Employee* object knows information about itself, like it’s id, name, salary.

• Each *Department* object knows information about itself, like which Employees are in the department.

• Once the data is loaded I can have each Department object provide me with the average salary of its employees.
HOW DO OBJECTS INTERACT?

In OO terminology objects in our program interact by sending *messages* to each other.

FileLoaderFrank, please load the Edit1.txt file.

Ok. Let me introduce you to DepartmentAccounting, DepartmentHR and DepartmentIT

DepartmentAccounting, what is your average salary?

$49,000

DepartmentIT, what is your average salary?

$73,333.33

ANOTHER PROGRAM

• Using a *procedural* design, what would I do if I wanted to create a program that found the maximum salary in a dept?

  Most likely you will copy the program and change it to use new calculations.

□ How does the same problem work with my OO design?

  I can *reuse* my existing objects, and just add a *new behavior* to the department object.
ONE MORE EXAMPLE...

• Create a procedural program that takes input from the user as to their weight, and outputs to the screen their weight on the other planets.

• The formulas you will need are as follows:
  – Mercury = .37 * weight
  – Venus = .876 * weight
  – Mars = .381 * weight
  – Jupiter = 2.637 * weight
  – Saturn = 1.151 * weight
  – Uranus = .79 * weight
  – Neptune = 1.12 * weight
  – Pluto = .025 * weight

• weight = the weight on Earth.

PROCEDURAL VIEW

– Mercury = .37 * weight
– Venus = .876 * weight
– Mars = .381 * weight
– Jupiter = 2.637 * weight
– Saturn = 1.151 * weight
– Uranus = .79 * weight
– Neptune = 1.12 * weight
– Pluto = .025 * weight
OO MODEL

Calculations are distributed and contained within various objects.

- $0.37 \times \text{weight}$
- $2.637 \times \text{weight}$
- $0.381 \times \text{weight}$
- $2.637 \times \text{weight}$
- $0.37 \times \text{weight}$
- $0.025 \times \text{weight}$
- $0.15 \times \text{weight}$
- $0.15 \times \text{weight}$
- $0.79 \times \text{weight}$
- $0.79 \times \text{weight}$
- $0.81 \times \text{weight}$