Object Oriented Programming

Definitions, Or What Is In A Name?

There is confusion over the definition of OOP, and not just in this class. I have seen people use one class in a program and claim it is object oriented. I saw an entire, large, project that converted to OOP. In their requirements and design documentation they did a text substation (find and replace). Where they had "module" they put "class". Where they had "function" they put "method". And overnight (really a couple of weeks) they were object-oriented. :-) (

Wikipedia has a good overview of the definition and discussion of some of the issues. But the most pertinent point is probably these two sentences:

> "Although discussions of object-oriented technology can get trapped in the details of one language vs. the other, the real key to the object-oriented approach is that it is a modelling approach first. The object-oriented approach is considered a logical extension of good design practices that go back to the very beginning of computer programming rather than being revolutionary as claimed by some proponents."
> (http://en.wikipedia.org/wiki/Object-oriented_programming, 24 Jul 14)

A procedural approach involves mastering and controlling the flow of execution through the program. This is mostly what we cover in 161. You can execute statements sequentially, or repeat blocks of code, or you can skip or select blocks of code. You think in terms of where do I put the data (variables). How do I process, convert, or change the data. Where do I put the results? And of course, which part of the flow am I in?

An object oriented approach involves thinking about the problem domain and how it can be divided. You divide the domain into sub domains, mapping them to classes. Due to common elements you will find inheritance useful. As I explain in the lectures the domain can be physical, such as planes, and gates, or it can be abstract, such as a flight, and take off position.

Notice "good design practices"! It is as much a way of thinking as anything else. That takes time, which increases the effort required to write code, which increases the cost. And delays delivery. So many organizations do not want to deal with it. This can also muddy the waters when talking about OOP.

Yes you use procedural features inside the class but the interaction of the parts (classes) is through a well-defined interface.

As the article points out, too often the discussion gets hung up in the programming language. But to a first approximation, yes, if you're doing OOP then you will have a class hierarchy and all parts of your program will be in a class in that hierarchy.

Programming languages now enter the "dispute". True object oriented languages; such as Smalltalk, and Objective-C, require all classes to be derived from a single parent (or true superclass). Other languages, such as C++, allow you to have multiple trees with separate roots. I would argue they are still object oriented. All of these will have one 'function', such as main(), that calls the other piece and gets the show started. If you have any other pieces that are not in the class hierarchy I would consider the program to be a hybrid.

I hope this helps. As I said at the beginning there is much dispute about this definition.