[0501] Intro to OOP

So in the late 1980s, the programming community moved from Structured Programming to the "next big thing" which was Object Oriented Programming. Why?

To fully understand OO programming and OO design, we need to understand what you cannot do without OO. Specifically, we need to understand "The Wall" that Structured Programming hit.

As we discussed in class, Structured Programming was all about taking high-level TASKS and breaking them down into smaller and smaller tasks until you, as a programmer, could implement each task directly in code. Where Structured Programming fell down was in the area of DATA - and specifically DATA STRUCTURES.

When programmers using a Structured Programming language like Pascal tried to create data structures like Stacks, Queues, Lists, etc. they discovered that, because Pascal only had structs, the data structure code could not protect the internal state of the data structure itself. For instance, a stack needs a "stack pointer" to keep track of its state, but in Pascal, there was no way to prevent maliciously code (or code with bugs in it) from maliciously (or accidentally) changing the stack pointer in a way that would cause the stack itself to break.

That was a HUGE problem. Programmers were unable to guarantee that their program's data couldn't be corrupted. Yikes!

The key problem with pure Structured Programming was that it was focused almost exclusively on PROCESS - the tasks and order of statement execution in a program. What programmers soon realized is that process is only HALF of the equation. The other half is the DATA.

OOP turns the whole programming process upside down. It was truly a revolution. In OOP, you start designing your program by identifying and creating mini "data modules" that manage the data your program needs to deal with. Only AFTER you've identified and created all the data modules that your program needs should you then string it all together (ironically, by using structured programming techniques).

What are Objects?

OBJECTS ARE DATA COMBINED WITH THE CODE THAT MANAGES THAT DATA. Data + Code - that's what an object is.

More specifically, Objects are instances of Classes created by the "new" operator using memory from the Heap.

More specifically, Objects are the combination of low-level data combined with code that specifically protects and makes use of that data in a reusable manner.

Key Point: A Class is a TEMPLATE that the "new" operator uses to make Objects. When you see the word "Class" think of the word "Template."

Over the years, programmers have found many different ways to use classes to make better programs. We will explore many of those methods over the rest of this course. We'll start with the most common type of class - and the easiest to understand - POJOs.