OBJECT-ORIENTED PROGRAMMING: CONCEPTS AND TRENDS

TUTORIAL PRESENTATION

John Minor Ross
Indiana University Kokomo
PO Box 9003, Kokomo, IN 46904-9003
317-455-9213 (area code 765 as of Feb. 1997)
jmross@indiana.edu

Object-oriented programming (OOP) has been touted as a new, hot concept for the 90s in many quarters. OOP, however, is almost thirty years old and has evolved into many dialects. These versions include pure OOP languages (e.g., Smalltalk), hybrid OOPs (e.g., C++), and OOP pretenders (e.g., Visual Basic). This tutorial is intended for those not yet conversant in OOP-speak and for attendees looking to select an OOP tool from among the myriad of OOP languages.

OOP is a programming approach designed to promote software reuse. OOP does not replace structured programming but it does change the way programmers approach design and implementation tasks. The essential OO triad of terms are:

- encapsulation -- OO approaches to programming empower data with the knowledge of how to behave. Rather than pass data structures to functions, data is packaged (encapsulated) with the routines, called methods, which are allowed to access and manipulate the data. Such a user-defined data type is referred to as a class (say class 'xyz'). When programmers proceed from this declaration stage to actually defining an instance of the class, they are commonly said to instantiate an object of class xyz.
- inheritance -- Once a class has been declared, the attributes (data) and behaviors (methods) may be close to fitting the needs of some future program. OOP encourages programmers to create new user-defined data types by inheritance to avoid code duplication. An existing class may serve as the base class for a new derived class that inherits the attributes and behaviors of the base class and then is modified to, typically, include additional variables or functionality. Some languages support multiple inheritance (a houseboat inherits from both the house class and the boat class).
- polymorphism -- This term refers to the technique whereby an executing program selects which method (of the same name) to run among objects whose class was derived from the same base class (without explicit if/else or switch/case logic). Traditional approaches are referred to as early or compile-time binding while the new approach uses the term late-binding.

The development of these concepts will be traced from SIMULA-67 to the present. Among the current contenders for the OOP crown are Smalltalk, C++, Ada and OO COBOL. Each has strengths and weaknesses and attendees will hear contrasting perspectives as to which OOP language is 'best.' Finally, the tutorial will discuss the all-to-real risks, fears, uncertainties and rewards of OOP on the job and in the classroom.