Object-Oriented Programming

Object-oriented programming is a set of techniques for creating objects and assembling them into a system that provides a desired functionality. An object is a software model composed of data and operations. An object's operations are used to manipulate its data. Objects may model things such as queues, stacks, windows, or circuit components. When assembled into a system, the objects function as delegators, devices, or models. Messages are passed between the objects to produce the desired results.

The eventual goal of object-oriented programming is to reduce the cost of software development by creating reusable and maintainable code. This is achieved by using three features of object-oriented programming: encapsulation, polymorphism, and inheritance. Encapsulation consists of data abstraction and data hiding. Data abstraction is a process of isolating attributes or qualities of something into an object model. With data hiding an object may contain its own private data and methods, which it uses to perform its operations. By using polymorphism, an object's operation may respond to many different types of data (e.g., graphical and textual). Finally, using inheritance, an object can inherit attributes from other objects. The object may then only need to add the attributes, methods, and data structures that make it different from the object from which it inherited its basic characteristics.

For the design of the object testing framework described in the accompanying article, we used an object-oriented software design methodology called object modeling technique (OMT). This methodology provides a collection of techniques and notation for designing an object-oriented application.

One important aspect of object-oriented design, or any software design, is deciding on who (i.e., module or object) is responsible for doing what. A technique provided in OMT involves using an index card to represent object classes. These cards are called CRC (class, responsibility, and collaboration) cards. The information on one of these cards includes the name of the class being described, a description of the problem the class is supposed to solve, and a list of other classes that provide services needed by the class being defined.