Programação orientada a objectos

Princípio

A ideia «object-oriented programming» baseia-se no facto do mundo à nossa volta ser formado por objectos. Por exemplo, quando se observa uma planta distinguimos nela várias partes constituintes, como raízes, caule, folhas e flores. Sabemos que a sua constituição mais íntima é feita de moléculas e átomos, mas sob o ponto de vista macroscópico (prático) o universo que mais interessa considerar limita-se a essas unidades estruturais. Por analogia, a programação tradicional, de natureza procedimental, trata dos átomos, enquanto a programação orientada a objectos se ocupa da planta. Dito de outro modo, a programação tradicional baseia-se em conceitos como fluência de dados ou lógica matemática, mas a programação orientada a objectos modela directamente as aplicações.

Exemplos

Assim, numa aplicação financeira o programa orientado a objectos vê objectos Clientes a enviar mensagens de débito e crédito para objectos Contabilidade, os quais podem cooperar para manter objectos Deve e Haver.

Da mesma maneira, um sistema CAD programado numa linguagem orientada a objectos modela um circuito eléctrico como um conjunto de objectos Componentes, cada um com uma colecção de objectos Pinos que se ligam a objectos Redes, de tal modo que quando um Componente recebe um sinal num dos seus Pinos de saída, este Pino envia uma mensagem à sua Rede, a qual comunica o sinal a todos os outros Pinos a ela ligados.

Diferença

Nos sistemas de programação procedimentais os dados e os procedimentos são entidades separadas. O programador é responsável pela aplicação de procedimentos activos a estruturas de dados passivas. Além disso, deve garantir que o procedimento opere correctamente sobre os tipos de dados a que se aplica.

Os sistemas de programação orientados a objectos não encaram um objecto como um dado passivo, mas antes como uma combinação do seu próprio estado com os métodos que o manipulam.

Linguagens

Os sistemas de programação orientados a objectos vêm dos anos 60, com o Simula67, mas só recentemente começaram a ser acessíveis à comunidade americana (e daí à internacional) de programadores, sobretudo com a linguagem Smalltalk-80, a qual foi mantida em secreto desenvolvimento no Xerox Palo Alto Research Center até 1981 e só em 1986 proporcionou a primeira conferência sobre «linguagens, aplicações e sistemas em programação orientada a objectos». A nova filosofia de linguagem, no entanto, já suscitou desenvolvimentos como Objective-C e CLOS (Common Lisp Object System).

Conceitos

Os principais conceitos das linguagens orientadas a objectos são:

- **Encapsulamento:** um objecto consiste numa representação encapsulada (estado) e num conjunto de mensagens (operações). Em vez de organizar os programas em procedimentos que repartem os dados globais, os dados são englobados nos procedimentos que dão acesso a esses dados («abstracção de dados» que existe em linguagens como Modula-2 ou Ada mas não se verifica em Pascal ou C);
- **Mensagens:** ao se dizer mensagens de passagem em vez de procedimentos quere-se indicar uma ligação mais leve entre o objecto e o utilizador, deixando em aberto a questão de saber como é que a mensagem se realiza (salto para uma subrotina ou comunicação interprocessos num sistema multitarefa ou distribuído);
- **Herança:** a herança comum é a característica que mais distingue este sistema de programação. Exemplo: numa aplicação bancária a Conta designa um objecto na forma geral das contas, enquanto Conta à Ordem e Conta a Prazo são objectos que têm propriedades adicionais: além do número da conta e do seu proprietário, têm balanços e taxas próprias, e só a Conta à Ordem permite cheques; é evidente que uma conta pode herdar as propriedades comuns à outra. A organização de uma árvore taxonómica facilita a herança, mas na prática será preciso executar estruturas de heranças múltiplas: as vantagens da herança múltipla têm de ser pesadas com os inconvenientes da complexidade.