Resumo POO

Lourenço Bogo

29 de dezembro de 2020

Sumário

1 Introdução

O terceiro e último grande tópico abordado foi o paradigma de programação conhecido como Orientação a Objetos. O paradigma foi inventado com o propósito de podermos abstrair nossos dados e esconder alguns detalhes de suas representações. Os códigos desse paradigma tendem a ficar mais modularizados e intuitivos.

Como já dito antes, o paradigma tem como objetivo principal a abstração dos dados, ou seja, podemos montar `classes` para representar mais abstratamente o dado que temos. Isso é muito útil, pois nos permite organizar nossos dados com mais facilidade e de maneira mais intuitiva.

2 Objetos

Tudo (ou quase tudo, dependendo da linguagem) nesse paradigma é representado por objetos. Um objeto é formado por um conjunto de dados e um conjunto de procedimentos (métodos), que nos permite alterar esses dados e produzir valores.

Isso nos conduz ao primeiro ponto principal do paradigma que é o encapsulamento. Só podemos acessar e alterar os dados de um certo objeto através da sua interface de funções (os métodos), nos dando duas vantagens principais:

- Abstração, já que alteramos o estado do objeto através de métodos que podem ser extremamente complexos
- Segurança, já que como o único jeito de alterar o estado do objeto é através dos métodos, se os métodos garantidamente sempre produzirem novos estados consistentes, não conseguiremos quebrar o programa.

Outra coisa essencial do paradigma é que, como já mencionado, podemos (e devemos) alterar os estados dos objetos, ou seja, temos efeitos colaterais, diferente do paradigma funcional, onde tínhamos idealmente apenas funções puras.

3 Classes

Uma classe é como uma fábrica de objetos de uma certa categoria. Todos os objetos criados a partir de uma classe terão um escopo interno com mesmos nomes e os mesmos métodos. Classes seguem uma estrutura hierárquica, ou seja, podemos definir uma classe B que é 'filha' de uma certa classe A. Nesse caso, dizemos que B está **herdando** de A, e temos as seguintes propriedades:

- B terá todas as variáveis de instância que A
- B terá os mesmos métodos que A
- B pode definir novas variáveis e métodos (incremento)
- B pode **REDEFINIR** os métodos de A (Redefinição)

Ao invés de herdar, podemos também delegar o trabalho para um objeto de outra classe, ou seja, podemos ter uma certa classe D, que contém um objeto da classe C. Desse modo, podemos usar os métodos definidos em C usando o objeto dentro de D.

Delegar é quase sempre melhor que herdar, a grande exceção pra isso é quando queremos usar a propriedade de redefinição da herança. Nesses casos, herdar é vantajoso.

4 Implementações de Herança

Em Java e Smalltalk, fazemos uma busca dinâmica pelos métodos, ou seja, quando queremos enviar uma mensagem (chamar um método), o que fazemos é procurar pelo nome desse método na classe e nas superclasses do objeto para o qual a mensagem está sendo enviada.

Já em C++, fazemos a execução direta do código, ou seja, cada objeto inclui ponteiros para cada um dos métodos de suas classes. Isso torna C++ a linguagem orientada a objetos mais rápida que existe e que pode existir, porém tira um pouco de expressividade.

5 Polimorfismo

Herança nos permite o uso de uma propriedade chamada polimorfismo. Suponhamos que temos uma classe Animal. Dessa classe, herdamos uma outra classe Cachorro e uma outra classe Gato. Cachorro e Gato podem ter implemetações diferentes para os métodos da classe animal (usando redefinição), porém uma função que recebe um Animal irá funcionar igualmente bem em objetos da classe Cachorro e objetos da classe Gato, já que os dois herdam da mesma classe Animal.

Isso é polimorfismo e é uma das propriedades da programação orientada a objeto que dá mais expressividade para o paradigma.

6 Diferentes tipos de OO

Em C++, tipos são basicamente a mesma coisa que classe. Precisamos redefinir métodos explicitamente usando a palavra chave `virtual` e não temos o conceito de interface, o mais próximos que podemos usar são classes abstratas puras.

Já em Smalltalk, tipos são apenas conjuntos de métodos e como a verificação dos tipos é feita dinamicamente, temos polimorfismo natural.

Em Java, tipos é uma classe mais uma interface implementada por essa classe. O polimorfismo é garantido pelo pelo uso das interfaces.

7 Implementação

Para implementarmos POO, temos que ter algumas coisas em mente:

- Um objeto sempre armazena o conteúdo das suas variáveis de instância
- O código dos métodos é compartilhado entre objetos de mesma classe, assim ocupamos menos espaço
- Todos os métodos têm um parâmetro implícito que é o próprio objeto que está chamado esse método. Assim os métodos podem alterar o estado desse objeto.