Programação orientada a objetos

Origem: Wikipédia, a enciclopédia livre.

Programação orientada a objetos (**POO**, ou **OOP** segundo as suas siglas em inglês) é um paradigma de programação baseado no conceito de "objetos", que podem conter dados na forma de campos, também conhecidos como *atributos*, e códigos, na forma de procedimentos, também conhecidos como métodos. Uma característica de objetos é que um procedimento de objeto pode acessar, e geralmente modificar, os campos de dados do objeto com o qual eles estão associados (objetos possuem uma noção de "this" (este) ou "self" (próprio)).

Em POO, programas de computadores são projetados por meio da composição de objetos que interagem com outros.[1][2] Há uma diversidade significante de linguagens de POO, mas as mais populares são aquelas baseadas em classes, significando que objetos são instâncias de classes, que, normalmente, também determinam seu tipo.

Muitas das linguagens de programação mais utilizadas (como C ++, Object Pascal, Java, Python, etc.) são linguagens de programação multiparadigmas que suportam programação orientada a objetos em maior ou menor grau, tipicamente em combinação com programação procedural imperativa. Linguagens orientadas a objeto significativas incluem Java, C++, C#, Python, Lua, PHP, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp e Smalltalk.

Índice

Características

A programação orientada a objetos utiliza objetos, porém nem todas das técnicas e estruturas associadas são suportadas diretamente em linguagens que afirmam suportar a POO. As características listadas abaixo são, entretanto, comuns entre linguagens consideradas fortemente orientadas a classes e objetos (ou multiparadigma com suporte a POO), com exceções notáveis mencionadas.[3][4][5][6]

Compartilhada com linguagens predecessoras não-POO

- Variáveis que podem armazenar informações formatadas em um número pequeno de tipos de dados nativos como inteiros e caracteres alfanuméricos. Isto pode incluir estruturas de dados como cadeia de caracteres, listas e tabelas hash que são nativas ou resultado de combinação de variáveis utilizando ponteiros de memória.

Objetos e classes

Baseada em classes vs baseada em protótipos

Ligação dinâmica/passagem de mensagens

Encapsulamento

Encapsulamento é um conceito de programação orientada a objetos que liga os dados e funções que manipulam os dados e que mantem ambos seguros de interferência externa e má utilização. O encapsulamento de dados leva ao importante conceito de POO de ocultação de dados.

Composição, herança e delegação

Objetos podem conter outros objetos em suas variáveis de instância. Isto é conhecido como composição de objetos. Por exemplo, um objeto na classe Empregado pode conter (apontar para) um objeto na classe Endereço, além de suas próprias variáveis de instância como "primeiro_nome" e "cargo". A composição de objetos é usada para representar relacionamentos "possui-um": cada empregado possui um endereço, desta forma cada objeto Empregado possui um lugar para armazenar um objeto Endereço.

Linguagens que suportam classes quase sempre suportam herança. Isto permite que classes sejam organizadas em uma hierarquia que representa relacionamentos "é-um-tipo-de". Por exemplo, a classe Empregado pode herdar da classe Pessoa. Todos os dados e métodos disponíveis à classe pai também aparecerão na classe filha com os mesmos nomes. Por exemplo, a classe Pessoa pode definir variáveis "primeiro_nome" e "último_nome" com o método "fazer_nome_completo()". Elas também estarão disponíveis na classe Empregado, que pode adicionar as variáveis "posição" e "salário". Esta técnica permite reutilização fácil dos mesmos procedimentos e definições de dados, além de potencialmente espelhar relacionamentos do mundo real de uma forma intuitiva. Em vez de utilizar tabelas de banco de dados e sub-rotinas de programação, o desenvolvedor utiliza objetos que o usuário pode estar mais familiarizado: objetos de seu domínio de aplicação.[7]

Subclasses podem sobrescrever os métodos definidos por superclasses. Herança múltipla é permitida em algumas linguagens, apesar disto poder complicar a resolução de sobrescrições. Algumas linguagens possuem suporte especial para mixin, assim como em qualquer linguagem com herança múltipla, uma mesclagem é simplesmente uma classe que não representa um relacionamento é-um-tipo-de. Mixins são normalmente utilizadas para adicionar os mesmos métodos para várias classes. Por exemplo, MesclagemConversãoUnicode pode fornecer um método unicode_para_ascii() quando incluído na classe LeitorDeArquivo e na classe FragmentadorDePáginaWeb, que não compartilham um pai comum.

Classes abstratas não podem ser instanciadas em objetos, elas existem apenas para o propósito de herança em outras classes "concretas" que podem ser instanciadas. Em Java, a palavra-chave `final` pode ser usada para prevenir uma classe de ser "sub-classeada".

Polimorfismo

Sub-tipagem, uma forma de polimorfismo, é quando o código de chamada pode ser agnóstico quanto a se um objeto pertence a uma classe pai ou a um de seus descendentes. Por exemplo, uma função pode chamar "fazer_nome_completo()" em um objeto, que funcionará se o objeto for da classe Pessoa ou da classe Empregado. Esse é outro tipo de abstração que simplifica o código externo à hierarquia de classes e permite uma forte separação de interesses.

Recursão aberta

História

O seu uso popularizou-se em princípios da década de 1990. Na atualidade, existe uma grande variedade de linguagens de programação que suportam a orientação a objetos.

Os conceitos da POO têm origem na Simula 67, uma linguagem desenhada para fazer simulações, criado por Ole-Johan Dahl e Kristen Nygaard, do Centro de Computação Norueguês em Oslo. Neste centro trabalhava-se em simulações de naves, que foram confundidas pela explosão combinatória de como as diversas qualidades de diferentes naves podiam afetar umas às outras. A ideia surgiu ao agrupar os diversos tipos de naves em diversas classes de objetos, sendo responsável cada classe de objetos por definir os seus "próprios" dados e comportamentos. Foram refinados mais tarde em Smalltalk, desenvolvido em Simula em Xerox PARC (cuja primeira versão foi escrita sobre Basic) mas desenhado para ser um sistema completamente dinâmico no qual os objetos se podiam criar e modificar "durante a caminhada" (em tempo de execução) em vez de ter um sistema baseado em programas estáticos.

A POO foi-se convertendo no estilo de programação dominante em meados dos anos 1980, em grande parte devido à influência de C++, uma extensão da linguagem de programação C. A sua dominação foi consolidada graças ao auge das interfaces gráficas de utilizador, para as quais a POO está particularmente bem adaptada. Neste caso, fala-se também de **programação dirigida por eventos**.

As características de orientação a objetos foram agregadas a muitas linguagens existentes durante esse tempo, incluindo Ada, BASIC, Lisp mais Pascal, entre outros. A adição destas características às linguagens que não foram desenhadas inicialmente para elas conduziu muitas vezes a problemas de compatibilidade e na capacidade de manutenção do código. As linguagens orientadas a objetos "puros", por seu lado, careciam das caraterísticas das quais muitos programadores haviam vindo a depender. Para saltar este obstáculo, fizeram-se muitas tentativas para criar novas linguagens baseadas em métodos orientados a objetos, mas permitindo algumas características imperativas de maneiras "seguras". A linguagem de programação Eiffel de Bertrand Meyer foi uma prematura e moderadamente acertada linguagem com esses objetivos, mas agora foi essencialmente substituída por Java, em grande parte devido à aparição da Internet e à implementação da máquina virtual Java na maioria dos navegadores web. PHP na sua versão 5 foi modificado; suporta uma orientação completa a objetos, cumprindo todas as características próprias da orientação a objetos.

Resumo

A POO é um paradigma surgido nos anos 1970, que utiliza objetos como elementos fundamentais na construção da solução. Um objeto é uma abstração de algum fato ou ente do mundo real, com atributos que representam as suas caraterísticas ou propriedades, e métodos que emulam o seu comportamento ou atividade. Todas as propriedades e métodos comuns aos objetos encapsulam-se ou agrupam-se em classes. Uma classe é um modelo, um protótipo para criar objetos; no geral, diz-se que cada objeto é uma instância ou exemplar de uma classe.

Veja-se também

- Base de dados orientada a objetos
- Engenharia de software baseada em componentes

Referências

1. Kindler, E.; Krivy, I. (2011). «Object-Oriented Simulation of systems with sophisticated control». International Journal of General Systems: 313–343
2. Lewis, John; Loftus, William (2008). *Java Software Solutions Foundations of Programming Design 6th ed*. [S.l.]: Pearson Education Inc. ISBN 0-321-53205-8, section 1.6 "Object-Oriented Programming"
3. Deborah J. Armstrong. *The Quarks of Object-Oriented Development*. A survey of nearly 40 years of computing literature which identified a number of fundamental concepts found in the large majority of definitions of OOP, in descending order of popularity: Inheritance, Object, Class, Encapsulation, Method, Message Passing, Polymorphism, and Abstraction.
4. John C. Mitchell, *Concepts in programming languages*, Cambridge University Press, 2003, ISBN 0-521-78098-5, p.278. Lists: Dynamic dispatch, abstraction, subtype polymorphism, and inheritance.
5. Michael Lee Scott, *Programming language pragmatics*, Edition 2, Morgan Kaufmann, 2006, ISBN 0-12-633951-1, p. 470. Lists encapsulation, inheritance, and dynamic dispatch.
6. Pierce, Benjamin (2002). *Types and Programming Languages*. [S.l.]: MIT Press. ISBN 0-262-16209-1, section 18.1 "What is Object-Oriented Programming?" Lists: Dynamic dispatch, encapsulation or multi-methods (multiple dispatch), subtype polymorphism, inheritance or delegation, open recursion ("this"/"self")
7. Jacobsen, Ivar; Magnus Christerson; Patrik Jonsson; Gunnar Overgaard (1992). *Object Oriented Software Engineering*. [S.l.]: Addison-Wesley ACM Press. pp. 43–69. ISBN 0-201-54435-0

Ligações externas

- Que é a programação orientada a objetos (http://www.desarrolloweb.com/articulos/499.php).
