

Programação Orientada a Objetos

O termo Programação Orientada a Objetos foi criado por Alan Kay, autor da linguagem de programação Smalltalk. Mas mesmo antes da criação do Smalltalk, algumas das idéias da POO já eram aplicadas, sendo que a primeira linguagem a realmente utilizar estas idéias foi a linguagem Simula 67, criada por Ole Johan Dahl e Kristen Nygaard em 1967.

Exemplo de linguagens modernas utilizadas por grandes empresas em todo o mundo que adotaram essas idéias: Java, C#, C++, Object Pascal (Delphi), Ruby, Python, Lisp, etc.

A maioria delas adota as idéias parcialmente, dando espaço para o antigo modelo procedural de programação (paradigma de programação baseado no conceito de chamadas a procedimento), como acontece no C++ por exemplo, onde temos a possibilidade de usar POO, mas a linguagem não força o programador a adotar este paradigma de programação, sendo ainda possível programar da forma procedural tradicional.

Idéias básicas da POO

A POO foi criada para tentar aproximar o mundo real do mundo virtual: a idéia fundamental é tentar simular o mundo real dentro do computador. A POO foi criada com intuito de resolver os problemas da programação estruturada (*exemplo: baixo reaproveitamento de código*).

Na POO o programador é responsável por moldar o mundo dos objetos, e explicar para estes objetos como eles devem interagir entre si. Os objetos "conversam" uns com os outros através do **envio de mensagens**, e o papel principal do programador é especificar quais serão as mensagens que cada objeto pode receber, e também qual a ação que aquele objeto deve realizar ao receber aquela mensagem em específico.

Programação estruturada versus POO

Uma linguagem estruturada permite que você aglomere os trechos de maior uso e transforme em uma sub-rotina ou função. Normalmente você consegue parametrizar estas funções. Um bom exemplo é criar uma função que escreve em um dado arquivo.

OO permite que você 'abraia' este tipo de programação. Você possui entidades que podem possuir atributos e métodos, num análogo às variáveis e funções, como se fossem programas independentes. Programar orientado à objeto é diferente na modelagem dos dados.

Exemplo, ao invés de você ter centenas de funções no mesmo contexto para atividades diferentes, você agrupa em objetos com comportamentos semelhantes

O que é um Objeto?

Um objeto é um elemento computacional que representa, no domínio da solução, alguma entidade (abstrata ou concreta) de interesse do sistema sob análise. Objetos similares são agrupados em classes. No paradigma da orientação a objetos, tudo pode

ser potencialmente representado como um objeto. Sob o ponto de vista da programação, um objeto não é muito diferente de uma variável no paradigma de programação convencional.

Quando se define uma variável em Java, essa variável terá:

Um espaço em memória para registrar o seu estado atual (um valor)

Um conjunto de operações associadas que podem ser aplicadas a ela, através dos operadores definidos na linguagem que podem ser aplicados a valores inteiros (soma, subtração, inversão de sinal, multiplicação, divisão inteira, resto da divisão inteira, incremento, decremento).

Da mesma forma, quando se cria um objeto, esse objeto adquire um espaço em memória para armazenar seu estado (os valores de seu conjunto de atributos, definidos pela classe) e um conjunto de operações que podem ser aplicadas ao objeto (o conjunto de métodos definidos pela classe).

Uma **mensagem** é um pequeno texto que os objetos conseguem entender e, por questões técnicas, não pode conter espaços. Junto com algumas dessas mensagens ainda é possível passar algumas informações para o objeto (parâmetros), dessa forma, dois objetos conseguem trocar informações entre si facilmente.

Vantagens da POO

A POO tem alcançado tanta popularidade devido às vantagens que ela traz. Entre elas podemos citar:

- Reusabilidade de código
- Escalabilidade de aplicações
- Manutenibilidade
- Apropriação

A reusabilidade de código é, sem dúvida, reconhecida como a maior vantagem da utilização de POO, pois permite que programas sejam escritos mais rapidamente. Todas as empresas sofrem de deficiência em seus sistemas informatizados para obter maior agilidade e prestar melhores serviços a seus clientes. Um levantamento feito na AT&T, a gigante das telecomunicações nos EUA, identificou uma deficiência da ordem de bilhões de linhas de código. Uma vez que a demanda está sempre aumentando, procura-se maneiras de desenvolver sistemas mais rapidamente, o que está gerando uma série de novas metodologias e técnicas de construção de sistemas (por exemplo, ferramentas CASE). A POO, através da reusabilidade de código, traz uma contribuição imensa nesta área, possibilitando o desenvolvimento de novos sistemas utilizando-se muito código já existente. A maior contribuição para reusabilidade de código é apresentada pela herança.

Escalabilidade pode ser vista como a capacidade de uma aplicação crescer facilmente sem aumentar demasiadamente a sua complexidade ou comprometer o seu desempenho. A POO é adequada ao desenvolvimento de grandes sistemas uma vez que

pode-se construir e ampliar um sistema agrupando objetos e fazendo-os trocar mensagens entre si. Esta visão de sistema é uniforme, seja para pequenos ou grandes sistemas (logicamente, deve-se guardar as devidas proporções).

O encapsulamento proporciona ocultamento e proteção da informação. Acessos a objetos somente podem ser realizados através das mensagens que ele está habilitado a receber. Nenhum objeto pode manipular diretamente o estado interno de outro objeto. De modo que, se houver necessidade de alterar as propriedades de um objeto ou a implementação de algum método, os outros objetos não sofrerão nenhum impacto, desde que a interface permaneça idêntica. Isto diminui em grande parte os esforços despendidos em manutenção. Além disso, para utilizar um objeto, o programador não necessita conhecer a fundo a sua implementação.

O polimorfismo torna o programa mais enxuto, claro e fácil de compreender. Sem polimorfismo, seriam necessárias listas enormes de métodos com nomes diferentes mas comportamento similar. Na programação, a escolha de um entre os vários métodos seria realizada por estruturas de múltipla escolha (case) muito grandes. Em termos de manutenção, isto significa que o programa será mais facilmente entendido e alterado.

A herança também torna a manutenção mais fácil. Se uma aplicação precisa de alguma funcionalidade adicional, não é necessário alterar o código atual. Simplesmente cria-se uma nova geração de uma classe, herdando o comportamento antigo e adiciona-se novo comportamento ou redefine-se o comportamento antigo.

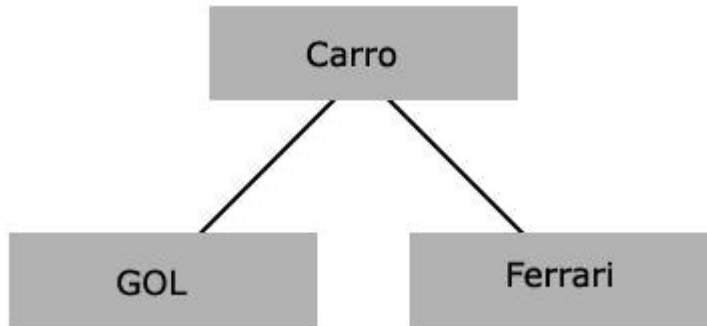
Outra vantagem da POO é também uma desvantagem: apropriação. Problemas do mundo real são freqüentemente solucionados facilmente utilizando as técnicas de orientação a objetos. As nossas mentes são classificadores naturais de coisas. Tipicamente nós classificamos coisas em hierarquias de classes. A POO leva vantagem desse fato e nos fornece uma estrutura que funciona como a nossa mente. Apropriação significa agrupar os objetos de comportamento similar em classes e essas classes em hierarquias de herança.

Herança

Herança é um mecanismo que permite que características comuns a diversas classes com comportamentos comuns ou parecidos, sejam abstraídas e centralizadas em uma classe base, ou superclasse. Tem uma relação “É um”.

Exemplo:

Um Jaguar “é um” carro. Logo ele foi herdado da classe carro, pois contém as características comuns de um carro.



Polimorfismo

É a capacidade que um mesmo método pode se comportar de diferentes maneiras em diferentes classes. Enquanto a herança é voltada mais as classes, o polimorfismo está voltado aos métodos. Imagine o método Correr na classe Carro. Agora imagine o mesmo método na subclasse GOL e na subclasse Ferrari. Em cada subclasse o método irá se comportar de maneira diferentes.

Formas de polimorfismo:

- Subclasses;
- Sobrescrita de método;
- Sobrecarga de método ou construtor

Classes

As classes são declarações de objetos, também se poderiam definir como abstrações de objetos. Isto quer dizer que a definição de um objeto é a classe. Quando programamos um objeto e definimos suas características e funcionalidades na verdade o que estamos fazendo é programar uma classe. Nos exemplos anteriores, na verdade falávamos das classes carro ou fração porque somente estivemos definindo, embora por alto, suas formas.

Desvantagens da POO

Apesar de das inúmeras vantagens, a POO tem também algumas desvantagens, que incluem:

- Apropriação
- Fragilidade
- Linearidade de desenvolvimento

A apropriação é apresentada tanto como uma vantagem como uma desvantagem, porque a POO nem sempre soluciona os problemas elegantemente. Enquanto que a mente humana parece classificar objetos em categorias (classes) e agrupar essas classes em relacionamentos de herança, o que ela realmente faz não é tão simples. Em vez

disso, objetos com características mais ou menos similares, e não precisamente definidas, são reunidos em uma classificação. A POO requer definições precisas de classes; definições flexíveis e imprecisas não são suportadas.

Na mente humana, essas classificações podem mudar com o tempo. Os critérios para classificar objetos podem mudar significativamente. A apropriação utilizada na POO torna-a muito rígida para trabalhar com situações dinâmicas e imprecisas. Além disso, algumas vezes não é possível decompor problemas do mundo real em uma hierarquia de classes. Negócios e pessoas têm frequentemente regras de operações sobre objetos que desafiam uma hierarquia limpa e uma decomposição orientada a objetos. O paradigma de objetos não trata bem de problemas que requerem limites nebulosos e regras dinâmicas para a classificação de objetos.

Fragilidade: Desde que uma hierarquia orientada a objetos requer definições precisas, se os relacionamentos fundamentais entre as classes chave mudam, o projeto original orientada a objetos é perdido. Torna-se necessário reanalisar os relacionamentos entre os objetos principais e reprojeter uma nova hierarquia de classes. Se existir uma falha fundamental na hierarquia de classes, o problema não é facilmente consertado. A mente humana adapta-se continuamente, e geralmente adequadamente, a situações novas. Ela encontra maneiras de classificar objetos no ambiente automaticamente. Enquanto a nossa mente tem essa capacidade, os ambientes POO não são tão bem equipados. Em virtude dessa fragilidade, a POO requer análise e projeto frontal para assegurar que a solução é adequada.

Linearidade : com isto existe uma tendência em criar uma abordagem linear de desenvolvimento, em vez de cíclica. Infelizmente, alguns problemas são “perversos”, o que significa que não se sabe como resolver um problema até efetivamente resolvê-lo. Tais problemas desafiam até mesmo os melhores projetistas e analistas de sistemas. Utilizar a abordagem de Desenvolvimento Rápido de Aplicações (RAD - Rapid Application Development, é utilizada pelo Delphi, SQL-Windows e outros ambientes de desenvolvimento para Windows adequados à prototipagem) com ciclos entre o projeto do protótipo, construção e realimentação do usuário é algumas vezes uma boa maneira de resolver problemas “perversos”. Entretanto, a POO necessita de um projeto cuidadoso da hierarquia de classes, o que pode elevar os custos da sua utilização em um ambiente RAD.

O estilo de programação com POO

O estilo de programação induzido pelo paradigma de objetos tem características muito precisas:

- Modularidade:** o estilo impõe alta modularidade na tarefa de projeto e programação. O conceito de objetos e sua descrição em classes incorporando dados e operações constituem um critério de modularização altamente efetivo e propiciam uma boa tradução prática de encapsulamento de informações.
- Suporte a generalização/especialização:** o estilo suporta diretamente a estruturação de conceitos em hierarquias de generalização/especialização que representa a deficiência

fundamental de abordagens de programação limitadas a tipos abstratos de dados (TADs).

- Visão balanceada entre dados e processos: o estilo tenta induzir, ao contrário de outros métodos, uma visão integrada de dados e processos na modelagem conceitual de aplicações.

- Composição “botton-up” de aplicação: o estilo privilegia a composição de uma nova aplicação a partir de composição de classes de aplicações preexistentes. Nesse sentido, a composição da aplicação é “bottom-up” (de baixo para cima, ao contrário de “top-down”).

- A atividade incremental e evolutiva: o estilo favorece uma abordagem incremental de programação, onde pequenas modificações podem ser rapidamente efetuadas e testadas, quer a partir da criação de novas classes, quer a partir da rápida especialização de um classe preexistente. •Reusabilidade de código: o estilo favorece a reutilização de código já existente, na forma de bibliotecas de classes que serão diretamente instanciadas ou modificadas pelo usuário.

- Robustez : o estilo favorece a criação de aplicações robustas, ou seja, muito estáveis em situações sujeitas a falhas. A maioria das linguagens orientadas a objetos possui mecanismos de tratar adequadamente erros vindos do usuário e também exceções do sistema.

- Programação em grande escala: o estilo é adequado ao desenvolvimento de sistemas grandes e complexos com equipes formadas por vários programadores. Algumas características apresentadas acima contribuem substancialmente para este item, como a modularidade e a reusabilidade.