
python

Programação Orientada a Objetos

POO

o que é POO?

POO

Técnicas de programação tradicionais

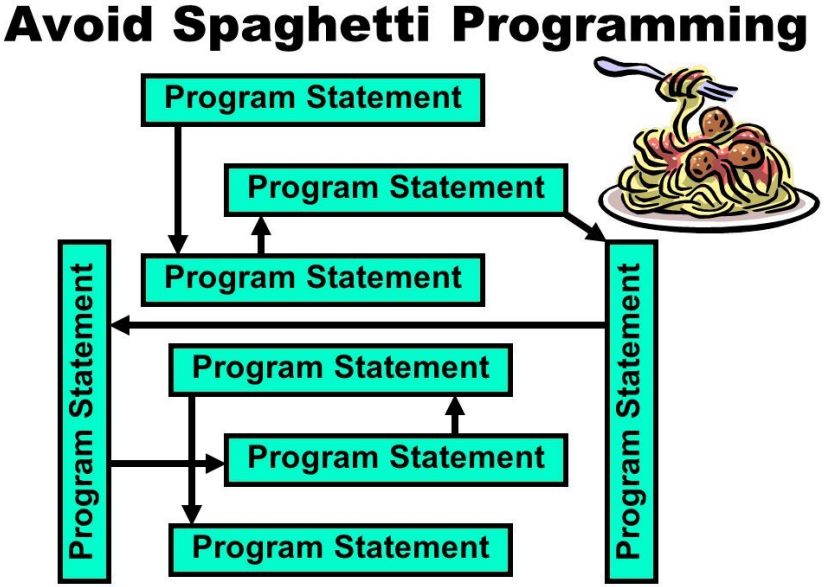
As técnicas de programação tradicionais, como por exemplo a “decomposição funcional”, leva o desenvolvedor a decompor o sistema em partes menores (funções), criando um emaranhado de inúmeras funções que chamam umas às outras.

Geralmente não há separação de conceitos e responsabilidades, causando dependências enormes no sistema, dificultando futuras manutenções no código do programa.

Não existe muito reaproveitamento de código, ao contrário, muitas vezes se tem muito código duplicado.

Programação Orientada a Objetos

E isso pode tornar-se um verdadeiro espaguete.



Programação Orientada a Objetos

O paradigma da Orientação a Objetos, ou Programação Orientada a Objetos (POO ou OOP), eleva a programação e o desenvolvimento de sistemas para um novo patamar.

A OO é um mecanismo moderno que ajuda a definir a estrutura de programas baseada nos conceitos do mundo real, sejam eles reais ou abstratos.

A OO permite criar programas componentizados, separando as partes do sistema por responsabilidades e fazendo com que essas partes se comuniquem entre sí, por meio de mensagens.

Essas partes do sistemas são chamadas de OBJETOS.

Programação Orientada a Objetos

A OO é mais intuitiva e fácil de aprender do que as técnicas tradicionais, pois foca o problema em conceitos do mundo real.

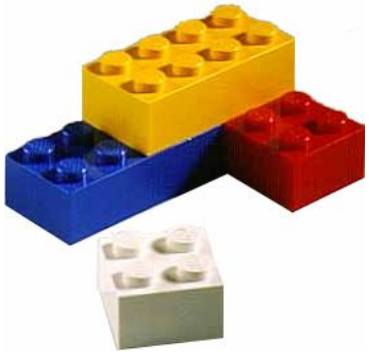
Dentre as vantagens que a OO proporciona, podemos destacar:

- aumento de produtividade
- reuso de código
- redução das linhas de código programadas
- separação de responsabilidades
- componentização
- maior flexibilidade do sistema
- escalabilidade
- facilidade na manutenção, dentre outras vantagens.

Programação Orientada a Objetos

A OO introduz e enfatiza os seguintes conceitos:

- Objeto
- Mensagem
- Classe
- Abstração
- Encapsulamento
- Herança
- Polimorfismo



Objetos

Objetos são a chave para se compreender a tecnologia orientada a objetos. Você olha ao seu redor e tudo o que vê são objetos: carro, mesa, janela, livro, pessoa, etc.

Os objetos do mundo real têm duas características em comum: ESTADO e COMPORTAMENTO.

Estado

O estado de um objeto revela seus dados importantes. Por exemplo, uma pessoa tem: idade, peso, altura, cor de cabelo, cor da pele.

Comportamento

O comportamento são as ações que aquele objeto pode exercer ou executar. Por exemplo, uma pessoa pode: andar, falar, ouvir, pular.

Objetos

Esses objetos podem ser tanto objetos concretos (carro, livro, nota fiscal), quanto conceitos abstratos (conta corrente, venda, pessoa jurídica).

Na Orientação a Objetos, os objetos do mundo real são modelados e representados no mundo computacional, ou seja, dentro do sistema, por meio de objetos de software.

Cada objeto deve ser conhecido, bem definido e ter seu limite e um significado dentro do sistema.

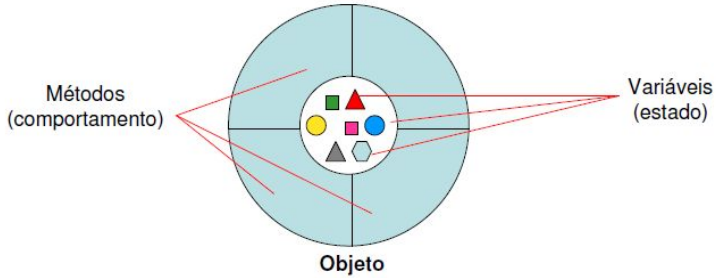
Os objetos de software, assim como os objetos do mundo real, também possuem estado e comportamento

Objetos

Um objeto de software mantém seu estado em uma ou mais de suas variáveis. Ele implementa seu comportamento através de seus métodos.

Método é o mesmo que função ou procedimento.

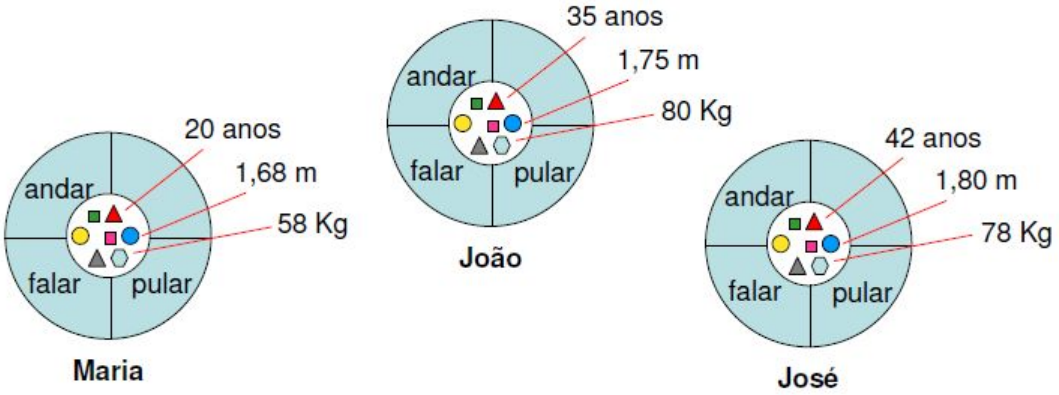
Por definição: Um objeto é um pedaço de software que possui ***variáveis*** (***ento)***.



Objetos

Um sistema pode conter um ou inúmeros objetos ativos. Cada objeto ativo no sistema em particular também é chamado de instância. As diferentes instâncias possuem seu próprio estado.

O e: pas.

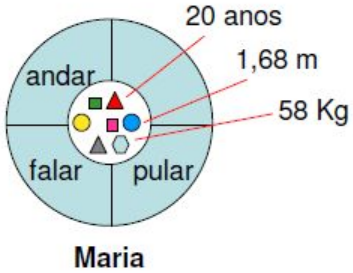


Objetos

Cada instância de pessoa possui um estado diferente em particular, como visto na última figura.

Porém, cada instância, além do estado, também possui seus métodos (comportamento) que operam sobre o próprio estado. Em outras palavras, para pular, cada pessoa vai fazer uma determinada força dependendo da sua idade, altura e peso, por exemplo.

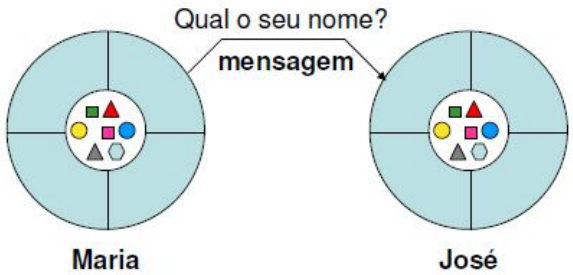
A idéia é que cada objeto seja respo[illegible]os (estado) e seja capaz de realizar as próprias operações que lhe foram atribuídas (comportamento).



Mensagem

Um objeto por si só não significa muito em um sistema. Para ter algum sentido e valor, esses objetos precisam interagir e comunicar-se entre si.

Os o[illegible] [illegible]ensagens.



Quando um objeto **A** quer se comunicar com um objeto **B** é enviada uma mensagem de **A** para **B**.

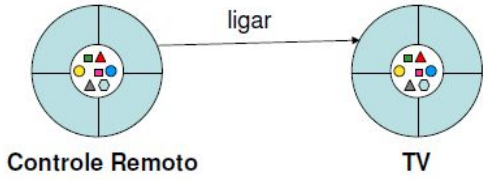
Mensagem

Enviar uma mensagem significa executar um método.

Então, se A envia uma mensagem para B, podemos entender como o objeto A executando um método do objeto B.

As mensagens são compostas por três partes:

- Objeto a quem a mensagem é endereçada
- Nome do método a ser chamad
- Parâmetros que o método recebe



Classes

No mundo real frequentemente percebemos vários objetos de um mesmo tipo. Por exemplo: seu carro é um dos muitos carros existentes no mundo.

Usando a terminologia OO, dizemos que um carro em particular é uma instância da classe de objetos conhecida como carros.

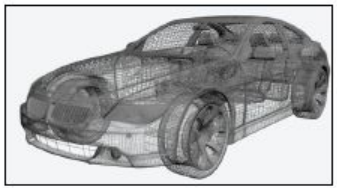
Os carros, em geral, possuem estado (cor, potência do motor, combustível) e comportamento (ligar, acelerar, brecar, mudar marcha) em comum.

O estado de cada carro é independente e pode ser diferente do estado dos outros carros. Cada carro pode ter uma cor diferente, por exemplo.

A partir dessas semelhanças, os fabricantes de veículos tiram vantagem disso para padronizar a construção de carros de um mesmo tipo, definindo um modelo único com características iguais para todos os carros a serem produzidos.

Classes

Na Orientação a Objetos também é possível ter vários objetos do mesmo tipo, que compartilham características em comum.
Tirando vantagem dessa semelhança entre alguns objetos, também é possível criar modelos para esses objetos. Esse modelo é chamado de CLASSE. As classes são tipos que podem ser criados.
Por definição: Uma classe é um modelo (protótipo) que define as variáveis (estado) e os odos os objetos do mesmo ti



Classe

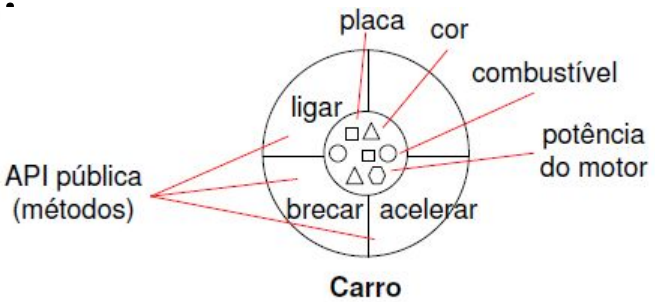
Objeto

Classes

Na classe são definidas as variáveis e implementados os métodos.

Os objetos são criados a partir de suas classes.

A cada objeto criado o sistema aloca memória para o novo objeto e suas variáve



Comumente fazem confusão entre classes e objetos. Lembre-se que

A classe define as características comuns e os objetos são instâncias dessas classes, com estado próprio.

Abstração

Abstração é a habilidade e a capacidade de se modelar conceitos, entidades, elementos, problemas e características do mundo real, de um domínio do problema em questão, levando-se em conta apenas os detalhes importantes para a resolução do problema e desprezando coisas que não têm importância no contexto.

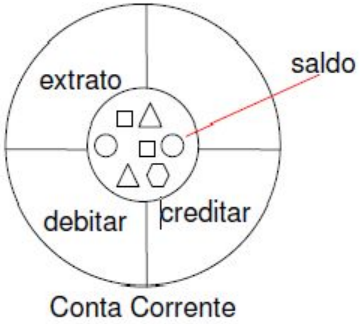
Se pensarmos no conceito de "conta corrente" bancária e abstrairmos este conceito, podemos identificar detalhes comuns, como o número da conta, número da agência e saldo; em operações como débito em conta, depósito e extrato da conta. Basicamente essas são as características de conta corrente para todos os bancos, apesar de um ou outro banco ter uma política de descontos de taxas etc.

Encapsulamento

Na OO, encapsulamento é o mecanismo utilizado para disponibilizar métodos que operam sobre os dados e que protegem o acesso direto indevido aos atributos de uma instância fora da classe onde estes foram declarados.

Esta proteção consiste em se usar modificadores de acesso mais restritivos sobre os atributos definidos na classe e fornecendo métodos que alteram os valores destes atributos de a

O encapsulamento ajuda a prevenir o probl cia externa indevida sobre os dados de um objeto, como objetos que possam alterar os dados de outros objetos



Herança

Herança é um mecanismo da OO que permite criar novas classes a partir de classes já existentes, aproveitando-se das características existentes na classe a ser extendida.

Este mecanismo é muito interessante pois promove um grande reuso e reaproveitamento de código existente.

Com a herança é possível criar classes derivadas (subclasses) a partir de classes bases (superclasses). As subclasses são mais especializadas do que as suas superclasses, mais genéricas.

As subclasses herdam todas as características de suas superclasses, como suas variáveis e métodos.

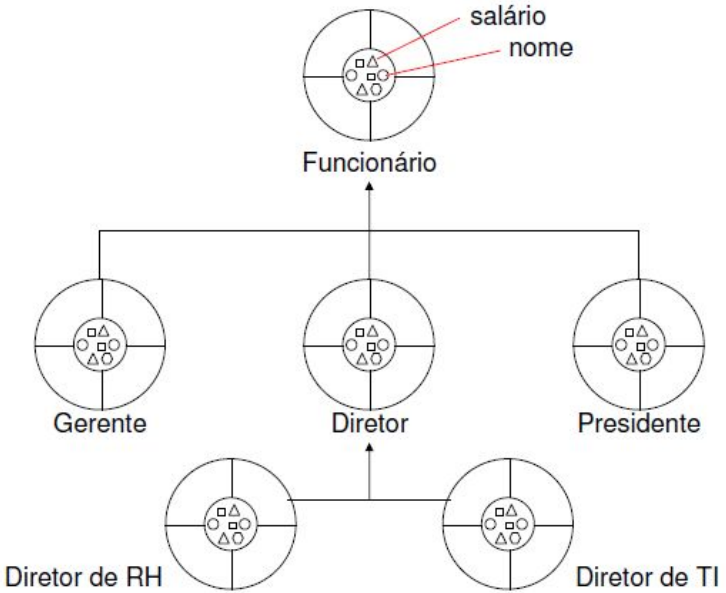
Herança

Imagine que dentro de uma organização empresarial, o sistema de RH tenha que trabalhar com os diferentes níveis hierárquicos da empresa, desde o funcionário de baixo escalão até o seu presidente.

Todos são funcionários da empresa, porém cada um com um cargo diferente. Mesmo a secretária, o pessoal da limpeza, o diretor e o presidente possuem um número de identificação, além de salário e outras características em comum.

Essas características em comum podem ser reunidas em um tipo de classe em comum, e cada nível da hierarquia ser tratado como um novo tipo, mas aproveitando-se dos tipos já criados, através da herança.

Herança


salário
nome
Funcionário
Gerente
Diretor
Presidente
Diretor de RH
Diretor de TI

Herança

Os subtipos, além de herdarem todas as características de seus supertipos, também podem adicionar mais características, seja na forma de variáveis e/ou métodos adicionais, bem como reescrever métodos já existentes na superclasse (polimorfismo).

A herança permite vários níveis na hierarquia de classes, podendo criar tantos subtipos quanto necessário, até se chegar no nível de especialização desejado.

Podemos tratar subtipos como se fossem seus supertipos, por exemplo o sistema de RH pode tratar uma instância de Presidente como se fosse um objeto do tipo Funcionário, em determinada funcionalidade.

Porém não é possível tratar um supertipo como se fosse um subtipo, a não ser que o objeto em questão seja realmente do subtipo desejado e a

Herança

Algumas linguagens de programação permitem herança múltipla, ou seja, uma classe pode estender características de várias classes ao mesmo tempo. É o caso do C++.

Outras linguagens não permitem herança múltipla, por se tratar de algo perigoso se não usada corretamente. É o caso do Java.

Na Orientação a Objetos as palavras classe base, supertipo, superclasse, classe pai e classe mãe são sinônimos, bem como as palavras classe derivada, subtipo, subclasse e classe filha também são sinônimos.

Polimorfismo

Formalmente polimorfismo quer dizer “várias formas”.

No caso da OO, polimorfismo denota uma situação na qual um objeto pode se comportar de maneiras diferentes ao receber uma mensagem, dependendo do seu tipo de criação.

O poliformismo é alcançado com auxílio do uso de herança nas classes e a reescrita (overriding) de métodos das superclasses nas suas subclasses.

Duas subclasses de uma mesma classe podem ter implementações completamente diferentes de um mesmo método, o que leva os objetos a se comportarem de forma diferente, dependendo do seu tipo (classe).

Polimorfismo

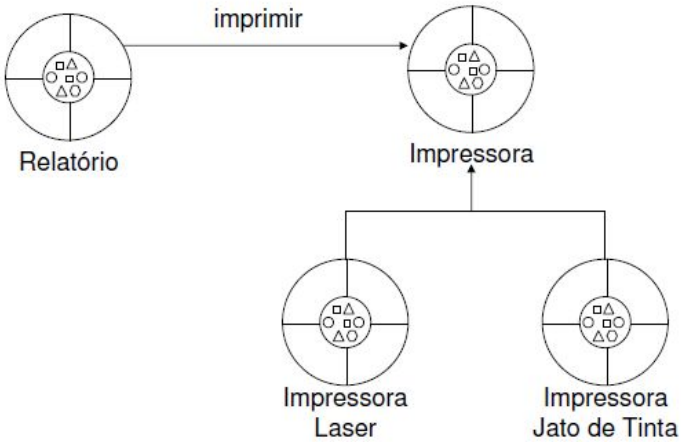
Exemplificando:

Podemos imaginar um programa que faça a impressão de um relatório, por meio de uma classe chamada Impressora, que é uma interface de acesso às funcionalidades da impressora usada, por meio de um driver fornecido pelo fabricante.

Uma impressora a laser tem um mecanismo de impressão totalmente diferente de uma impressora a jato de tinta, mas isso não importa para o programa.

Ele manda uma simples mensagem de imprimir para a impressora, e o modo como a impressora imprime no papel varia de acordo com o tipo de impressora usada, ou seja, a impressão se dá de formas diferentes para a mesma mensagem de imprimir.

Polimorfismo



Polimorfismo

Algumas linguagens promovem o polimorfismo principalmente através do uso de classes abstratas e interfaces, como é o caso da tecnologia Java.

Classes abstratas são classes que não podem gerar instâncias de objetos e que possuem um ou mais métodos sem implementação, deixando para suas subclasses a tarefa de implementar estes métodos abstratos.

Interfaces são um tipo de contrato que algumas classes têm de seguir, ou seja, as interfaces apenas definem métodos abstratos que as classes que implementam esta interface têm de implementar.

Conclusão

O paradigma da Orientação a Objetos traz um ganho significativo na qualidade da produção de software, porém grandes benefícios são alcançados quando as técnicas de programação OO são colocadas em prática com o uso de uma tecnologia que nos permita usar todas as características da OO; além de agregar à programação o uso de boas práticas de programação e padrões de projeto (design patterns).

Esse é um dos motivos do sucesso da tecnologia Java, que suporta a OO completamente e também fornece mecanismos para se usar os design patterns conhecidos.

Além do conhecimento da Orientação a Objetos, o conhecimento da UML (Unified Modelling Language) ajuda muito no desenho e