IBM SkillsBuild

IBM

Noções básicas de programação

O que é Programação Orientada a Objetos (POO)

Índice

Introdução

A Programação Orientada a Objetos (OOP) é um paradigma de programação, ou seja, um modelo ou estilo de programação que nos dá orientações sobre como trabalhar com ela. Ela se baseia no conceito de classes e objetos. Este tipo de programação é usado para estruturar um programa de software em peças simples e reutilizáveis de esquemas de código (classes) para criar instâncias individuais de objetos.

Ao longo da história, diferentes paradigmas de programação têm surgido. Linguagens seqüenciais como COBOL ou linguagens de procedimento como Basic ou C focaram mais na lógica do que nos dados. Linguagens mais modernas como Java, C# e Python usam paradigmas para definir programas, sendo a Programação Orientada a Objetos a mais popular.

Com o paradigma da Programação Orientada a Objetos, o que procuramos é parar de focar na lógica pura dos programas e começar a pensar em objetos, que é a base deste paradigma. Isto nos ajuda muito em grandes sistemas, porque em vez de pensarmos em funções, pensamos nas relações ou interações dos diferentes componentes do sistema.

Um programador projeta um programa de software organizando partes de informação e comportamentos relacionados em um modelo chamado classe. Então, objetos individuais são criados a partir do modelo de classe. Todo o programa de software é executado fazendo com que vários objetos interajam uns com os outros para criar um programa maior.

Por que POO?

A Programação Orientada a Objetos permite que o código seja reutilizável, organizado e fácil de manter. Ele segue o princípio de desenvolvimento de software utilizado por muitos programadores DRY (Don't Repeat Yourself), para evitar a duplicação de código e assim criar programas eficientes. Também evita o acesso indesejado aos dados ou a exposição de código proprietário através de encapsulamento e abstração, que será discutido mais detalhadamente mais adiante..

Classes, objetos e instâncias

Pensar em termos de objetos é muito semelhante a como o faríamos na vida real. Por exemplo, vamos pensar em um carro e tentar modelá-lo em um esquema OOP.

Diríamos que o carro é o elemento principal que tem uma série de características, tais como cor, modelo ou marca. Além disso, ele tem uma série de funcionalidades associadas, tais como partida, parada ou estacionamento.

Portanto, pensar em objetos requer a análise dos elementos que você vai tratar em seus programas, tentando identificar suas características e funcionalidades. Quando tivermos um ecossistema de objetos, eles colaborarão uns com os outros para resolver os objetivos das aplicações.

Talvez no início possa ser um pouco complexo dar esse salto, pensar em objetos, mas com o tempo será uma tarefa que você executará automaticamente.

Lidando com o conceito de classe

Em um esquema de programação orientado a objetos "o carro" seria o que é conhecido como uma "Classe".

A classe contém a definição das características de um modelo (o carro), tais como cor ou marca, juntamente com a implementação de suas funcionalidades, tais como partida ou parada.

As características definidas na classe são chamadas de propriedades e as funcionalidades associadas são chamadas de métodos.

Para entender este importante conceito em Programação Orientada a Objetos, podemos pensar na classe como um livro, que descreve como são todos os objetos do mesmo tipo. A classe carro descreve como são todos os carros do mundo, ou seja, quais propriedades eles têm e quais funcionalidades eles devem ser capazes de executar e, é claro, como eles são executados.

Manuseio do conceito de objetos

A partir de uma classe, podemos criar qualquer número de objetos dessa classe. Por exemplo, a partir da classe "o carro" podemos criar um carro vermelho que é da marca Ford e modelo Fiesta, outro carro verde que é da marca Seat e modelo Ibiza.

Portanto, os objetos são exemplos de uma classe, ou elementos concretos criados a partir de uma classe. Você pode entender a classe como o molde e os objetos como concreções criadas a partir do molde da classe.

Outros exemplos concretos de classes e objetos

Por outro exemplo, vejamos como modelaríamos em um esquema POO uma fração, ou seja, aquela estrutura matemática que tem um numerador e um denominador que divide o numerador, por exemplo 3/2.

A fração será a classe e ela terá duas propriedades, o numerador e o denominador. Ela poderá então ter vários métodos, como simplificar, adicionar com outra fração ou número, subtrair com outra fração, etc.

A partir da definição de uma fração (a classe) podemos construir um número indeterminado de objetos do tipo fração. Por exemplo, podemos ter a fração de objeto 2/5 ou 3/9, 4/3, etc. Estes são todos os objetos da classe fração inteira.

Nossa classe de fração pode ser usada em qualquer número de programas, por exemplo, em um programa matemático, ela fará uso da classe de fração e construir muitos objetos de fração para fazer vários cálculos matemáticos. Mas você poderia usar essa mesma classe de fração em um programa de contabilidade ou de faturamento.

Outro exemplo pode ser a classe de coordenadas, que tem duas propriedades, o valor x e y. Você poderia ter outra classe "coordenada 3D", que precisaria de 3 propriedades x, y e z. As coordenadas poderiam ser adicionadas a outra coordenada, exibida em um gráfico, encontrar o caminho mais curto entre duas coordenadas, e assim por diante.

Estes são apenas exemplos simples de classes, que também servem para destacar a reusabilidade que podemos conseguir com objetos. A classe de coordenadas pode ser usada em um número infinito de programas gráficos, mapas, etc. Por outro lado, a classe de carro pode ser usada em um programa de gerenciamento de oficina de automóveis ou em um programa de gerenciamento de estacionamentos. Da classe de carro, diferentes objetos do tipo carro serão criados para realizar as operações na aplicação da oficina, bem como na aplicação do estacionamento.

Os objetos colaboram uns com os outros

Em línguas puramente orientadas a objetos, teremos apenas classes e objetos. As classes nos permitirão definir um número indeterminado de objetos, que colaboram uns com os outros para resolver os problemas.

Com muitos objetos de diferentes classes, poderemos realizar as ações que queremos implementar na funcionalidade da aplicação.

Além disso, as próprias aplicações como um todo também serão definidas por meio de classes. Ou seja, a oficina de automóveis será uma classe, a partir da qual poderemos criar o objeto oficina de automóveis, que utilizará objetos carro, objetos de ferramenta de classe, objetos de mecânica de classe, objetos de peça sobressalente de classe, e assim por diante.

Para continuar, vamos definir mais formalmente os conceitos que acabamos de introduzir acima.

Classes em Programação Orientada a Objetos

Como você deve ter entendido, as classes são declarações de objetos; elas também poderiam ser definidas como abstrações de objetos. Isto significa que a definição de um objeto é a classe. Quando programamos um objeto e definimos suas características e funcionalidades, estamos na verdade programando uma classe. Nos exemplos anteriores estávamos na verdade falando sobre as classes carro ou fração porque estávamos apenas definindo, ainda que de uma forma aproximada, suas formas.

Propriedades nas classes

As propriedades ou atributos são as características dos objetos ou espécimes de uma classe. Quando definimos uma propriedade, normalmente especificamos seu nome e tipo.

Embora não seja uma forma muito correta de falar, podemos ter a idéia de que propriedades são algo como variáveis onde armazenamos os dados relacionados aos objetos.

Métodos em classes

Elas são as funcionalidades associadas aos objetos, que são implementadas ou programadas dentro das classes. Ou seja, quando estamos programando as classes, as funções que criamos dentro associadas a essas classes, chamamos de métodos.

Embora os métodos sejam como funções, é importante chamá-los de métodos para deixar claro que são funções que existem dentro do contexto de uma classe, funções que podemos invocar em todos os objetos criados a partir de uma classe.

Objetos em Programação Orientada a Objetos

Os objetos são cópias de uma classe. A partir de uma classe eu posso criar cópias (objetos) dessa classe, que terá, portanto, as características e funcionalidades definidas nessa classe.

Quando criamos uma instância, temos que especificar a classe a partir da qual ela será criada. Esta ação de criar um objeto a partir de uma classe é chamada de instantiating (que vem de uma tradução errada da palavra instace, que em inglês significa espécime). Por exemplo, um objeto da fração de classe é, por exemplo, 3/5. O conceito ou definição de fração seria a classe, mas quando já estamos falando de uma fração específica 4/7, 8/1000, ou qualquer outra, chamamos de objeto.

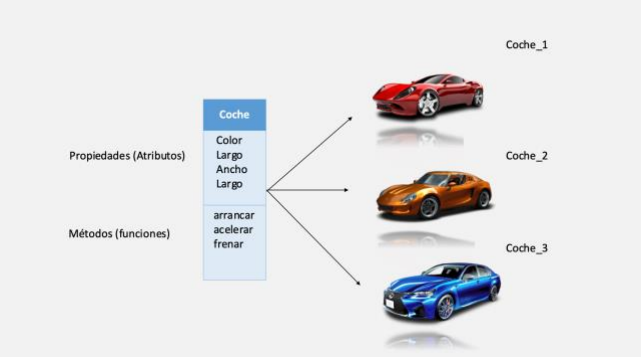
Para criar um objeto é preciso escrever uma instrução especial que pode ser diferente dependendo da linguagem de programação utilizada, mas que será algo semelhante a isto.

miCoche = new Coche()

Com a palavra "new", especificamos que uma instância da seguinte classe deve ser criada.

Neste caso, "Coche" seria o nome da classe. Com a palavra "novo" é criada uma nova instância de carro, um objeto concreto da classe carro e esse objeto é armazenado na variável "miCoche".

Dentro dos parênteses poderíamos colocar parâmetros com os quais inicializar o objeto da classe do carro, como sua cor, ou sua marca.



Estados em objetos

Quando temos um objeto, suas propriedades tomam valores. Por exemplo, quando temos um carro, a propriedade da cor tomará um valor particular, como o vermelho ou o cinza metálico. O valor concreto da propriedade de um objeto é chamado de estado.

Para acessar o estado de um objeto para ver seu valor ou alterá-lo, é utilizado o operador de pontos.

miCoche.color = "rojo"

O objeto é miCoche, depois colocamos o operador de ponto e finalmente o nome do bem que queremos acessar. Neste exemplo, estamos mudando o estado da propriedade " color" do objeto para o valor " rojo". Fazemos isso com uma simples atribuição.

Mensagens em objetos

Uma mensagem em um objeto é a ação de fazer uma chamada de método. Por exemplo, quando dizemos a um objeto de carro para começar, estamos passando-lhe a mensagem "start".

Para enviar mensagens aos objetos usamos o operador de ponto, seguido do método que queremos invocar e parênteses, como nas chamadas de função.

miCoche.ponteEnMarcha()

Neste exemplo, passamos a mensagem "ponteEnMarcha" para o objeto "miCoche".

Após o nome do método que queremos invocar, temos que colocar parênteses, como quando invocamos uma função. Dentro dos parênteses estariam os parâmetros, se o método os exigir.

4 Princípios da Programação Orientada a Objetos

Encapsulamento

O encapsulamento contém todas as informações importantes de um objeto dentro do objeto e só expõe informações selecionadas para o mundo exterior.

Esta propriedade assegura que a informação de um objeto seja escondida do mundo exterior, agrupando em uma Classe as características ou atributos que têm acesso privado, e os comportamentos ou métodos que têm acesso público.

O encapsulamento de cada objeto é responsável por suas próprias informações e por seu próprio estado. A única maneira de modificar isto é através dos próprios métodos do objeto. Portanto, os atributos internos de um objeto devem ser inacessíveis do exterior e só podem ser modificados através da chamada das funções correspondentes. Isto é para manter o estado a salvo de usos indevidos ou inesperados.

Usamos um carro como exemplo para explicar o encapsulamento. O carro compartilha informações públicas através das luzes de freio ou sinais de giro para indicar curvas (interface pública). Em contraste, temos a interface interna, que seria o mecanismo de propulsão do carro, que está escondido sob o capô. Ao dirigir um carro, é necessário indicar seus movimentos a outros motoristas, mas não expor dados particulares sobre o tipo de combustível ou temperatura do motor, pois são muitos dados, o que confundiria outros motoristas.

Abstração

Abstração é definida quando o usuário interage apenas com atributos e métodos selecionados de um objeto, utilizando ferramentas simplificadas de alto nível para acessar um objeto complexo.

Na programação orientada ao objeto, os programas são freqüentemente muito grandes e os objetos se comunicam muito uns com os outros. O conceito de abstração facilita a manutenção de grandes códigos, onde diferentes mudanças podem ocorrer ao longo do tempo.

Assim, a abstração é baseada no uso de coisas simples para representar a complexidade. Objetos e classes representam o código subjacente, escondendo detalhes complexos do usuário. É, portanto, uma extensão do encapsulamento. Continuando com o exemplo do carro, você não precisa conhecer todos os detalhes de como o motor funciona para poder dirigi-lo..

Inheritance

A herança define relações hierárquicas entre classes, para que atributos e métodos comuns possam ser reaproveitados. As classes principais estendem atributos e comportamentos às classes infantis. Ao definir atributos e comportamentos básicos em uma classe, podem ser criadas classes menores, ampliando assim a funcionalidade da classe pai e adicionando atributos e comportamentos adicionais.

Voltando ao exemplo dos animais, uma única classe animal pode ser usada e um atributo do tipo animal pode ser adicionado que especifica o tipo de animal. Diferentes tipos de animais precisarão de métodos diferentes, por exemplo, aves precisam ser capazes de pôr ovos e peixes precisam ser capazes de nadar. Mesmo quando os animais têm um método comum, como o movimento, a implementação precisaria de muitas declarações "se" para garantir o comportamento correto do movimento. Por exemplo, as rãs saltam, enquanto as cobras deslizam. O princípio da herança nos permite resolver este problema.

Polimorfismo

O polimorfismo consiste em projetar objetos para compartilhar comportamentos, o que nos permite processar objetos de diferentes maneiras. É a capacidade de apresentar a mesma interface para diferentes formas ou tipos de dados subjacentes. Ao usar a herança, os objetos podem substituir os comportamentos primários compartilhados por comportamentos secundários específicos.

O polimorfismo permite que o mesmo método execute comportamentos diferentes de duas maneiras: sobreposição de métodos e sobrecarga de métodos.

Muitas coisas são construídas em torno destes princípios de programação orientada a objetos. Por exemplo, os princípios SOLID, ou padrões de projeto, que são receitas que se aplicam a problemas recorrentes que foram encontrados e se repetem em vários projetos.

Benefícios da Programação Orientada a Objetos

- Reutilização do código.
- Transformar coisas complexas em estruturas simples e reprodutíveis.
- Evita a duplicação de códigos.
- Permite o trabalho em equipe graças ao encapsulamento, pois minimiza a possibilidade de duplicação de funções quando várias pessoas trabalham no mesmo objeto ao mesmo tempo.
- Como a classe está bem estruturada, permite a correção de erros em vários lugares do código.
- Ela protege informações através de encapsulamento, uma vez que os dados do objeto só podem ser acessados através de propriedades e métodos privados.
- A abstração nos permite construir sistemas mais complexos de uma forma mais simples e organizada.

Neste curso, veremos o conceito de OOP aplicado às linguagens de programação Javascript e Python. No caso do Javascript, a OOP não tem muito impacto, mas Python é uma linguagem totalmente orientada a objetos. Em Python, todos os elementos são considerados como objetos, com suas propriedades e métodos. Portanto, o uso da OOP em Python será constante. Iremos mais fundo quando chegarmos a essa parte do programa de ensino.