UNIP
UNIVERSIDADE PAULISTA

UNICLUSTER

Linguagem de programação Orientada a Objetos

Carlos Arruda Baltazar

UNIP

História

A Programação Orientada ao Objeto foi concebida há muito tempo atrás (no inicio da década de 70), a sua origem vem da linguagem Simula (Simula Language), concebida na Noruega na década de 60, e como o nome indica, foi criada para fazer simulações; entretanto, seu uso alavancou um conceito que até então passava desapercebido pela maioria dos projetistas: a similaridade com o mundo real.

História

A primeira linguagem de programação a implementar os conceitos de POO foi a linguagem SIMULA-68; em seguida surgiu a linguagem Smalltalk; criada pela Xerox, que pode ser considerada a linguagem que popularizou e incentivou o emprego da POO.

O resultado foi uma linguagem de pura linhagem O.O. (Orientada a Objeto), poderosíssima, que implementa todos os conceitos de Orientação a Objetos, o que não acontece com as chamadas linguagens OO híbridas que implementam apenas alguns conceitos de orientação ao objeto.

História

Fundamentalmente o que se deseja com esta metodologia são basicamente duas características: reutilização de código e modularidade de escrita.

Formalmente, para ser considerada uma linguagem OO, esta precisa implementar quatro conceitos básicos: abstração, encapsulamento, herança e polimorfismo.

Programação Orientada a Objeto

Programação orientada a objetos (POO) 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.

Programação Orientada a Objeto

Em POO, programas de computadores são projetados por meio da composição de objetos que interagem com outros. 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.

Programação Orientada a Objeto

Programas desenvolvidos utilizando programação estruturada, mesmo particionado em funções, acabam ficando muito extensos, o que acaba gerando uma dificuldade na manutenção. A Programação Orientada a Objetos (POO) surgiu com a finalidade de facilitar o processo de desenvolvimento, tornando-o mais intuitivo e otimizado.

Programação Orientada a Objeto

- Vantagens

✓ Reutilização de código;

✓ Projetos bem definidos;

✓ Trechos de código com atividade bem determinada;

✓ Fácil manutenção;

- Desvantagens

❖ Dificuldade de aprendizado;

❖ Projeto mal definido;

Programação Orientada a Objeto

Quando começamos a utilizar linguagens que possibilitam o paradigma orientado a objetos, é comum errarmos e aplicarmos a programação estruturada achando que estamos usando recursos da orientação a objetos.

Programação Orientada a Objeto

Na programação estruturada, um programa é composto por três tipos básicos de estruturas:

- sequências: são os comandos a serem executados
- condições: sequências que só devem ser executadas se uma condição for satisfeita (exemplos: if-else, switch e comandos parecidos)
- repetições: sequências que devem ser executadas repetidamente até uma condição for satisfeita (for, while, do-while etc)

Essas estruturas são usadas para processar a entrada do programa, alterando os dados até que a saída esperada seja gerada. Até aí, nada que a programação orientada a objetos não faça.

A diferença principal é que na programação estruturada, um programa é tipicamente escrito em uma única rotina (ou função) podendo, é claro, ser quebrado em subrotinas. Mas o fluxo do programa continua o mesmo, como se pudéssemos copiar e colar o código das subrotinas diretamente nas rotinas que as chamam, de tal forma que, no final, só haja uma grande rotina que execute todo o programa.

Programação Orientada a Objeto

A **programação orientada a objetos** surgiu como uma alternativa a essas características da programação estruturada. O intuito da sua criação também foi o de aproximar o manuseio das estruturas de um programa ao manuseio das coisas do mundo real, daí o nome "objeto" como uma algo genérico, que pode representar qualquer coisa tangível.

Esse novo paradigma se baseia principalmente em dois conceitos chave: **classes** e **objetos**. Todos os outros conceitos, igualmente importantes, são construídos em cima desses dois.

Objeto

Na literatura, um objeto pode ser definido como um artefato material ou abstrato que pode ser percebido pelos sentidos e descrito através de suas características, comportamentos e estados atuais. Logo, em programação, o objeto é uma abstração de software que pode ser definido por características e ações:

Objeto

- **Atributos:** são variáveis responsáveis por armazenar as características do objeto;
- **Métodos:** são rotinas que, quando executadas realizam algumas tarefas como alterar o conteúdo dos próprios atributos, bem como realizar interações entre objetos;
- **Estado:** emerge quando valores são definidos para os atributos de um objeto.

Classes

Uma classe pode ser considerada um modelo para a criação de objetos. Portanto são as classes que dão “vida” ao objeto, pois nela são definidas as características do objeto através dos atributos e também são criadas as suas funcionalidades através dos métodos. Assim, uma classe tem o papel de definir e estabelecer o comportamento do objeto. Também, é necessário lembrar que vários objetos podem ser criados a partir de uma única classe, o que diferencia os objetos são suas características.

Mas como assim ?

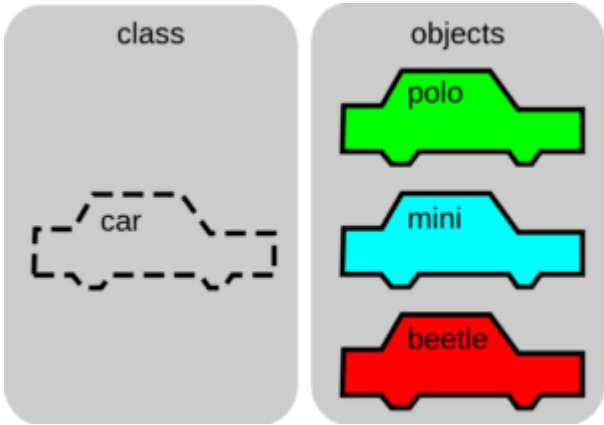
Imagine que você comprou um carro recentemente e decide modelar esse carro usando programação orientada a objetos. O seu carro tem as características que você estava procurando: um motor 2.0 híbrido, azul escuro, quatro portas, câmbio automático etc. Ele também possui comportamentos que, provavelmente, foram o motivo de sua compra, como acelerar, desacelerar, acender os faróis, buzinar e tocar música. Podemos dizer que o carro novo é um *objeto*, onde suas características são seus *atributos* (dados atrelados ao objeto) e seus comportamentos são ações ou *métodos*.

Mas como assim ?

Seu carro é um objeto seu mas na loja onde você o comprou existiam vários outros, muito similares, com quatro rodas, volante, câmbio, retrovisores, faróis, dentre outras partes. Observe que, apesar do seu carro ser único (por exemplo, possui um registro único no Departamento de Trânsito), podem existir outros com exatamente os mesmos atributos, ou parecidos, ou mesmo totalmente diferentes, mas que ainda são considerados *carros*. Podemos dizer então que seu objeto pode ser classificado (isto é, seu *objeto pertence à uma classe*) como um carro, e que seu carro nada mais é que uma *instância* dessa *classe* chamada "carro".

Mas como assim ?

- Assim, abstraindo um pouco a analogia, uma classe é um conjunto de características e comportamentos que definem o conjunto de objetos pertencentes à essa classe. Repare que a classe em si é um conceito abstrato, como um molde, que se torna concreto e palpável através da criação de um objeto. Chamamos essa criação de *instanciação da classe*, como se estivéssemos usando esse molde (classe) para criar um objeto.



Modificadores

- Público (*public*): a classe, atributo ou método são vistos pelas demais classes se estiverem dentro do mesmo pacote;

- Protegida (*protected*): o acesso a uma classe, atributo ou método se restringem apenas a própria classe e suas subclasses;

- Privado (*private*) – o acesso o acesso a uma classe, atributo ou método só é permitido somente pela própria classe.

Diagrama de Classes - UML

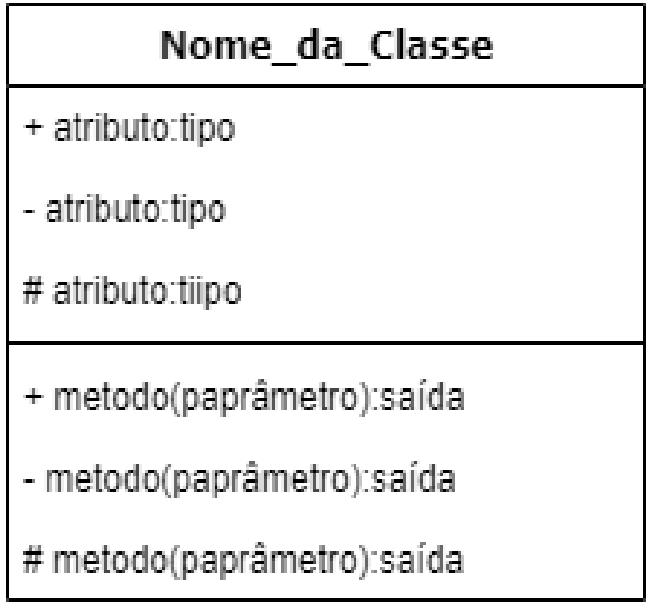
Basicamente, UML (Unified Modeling Language) é uma linguagem de notação (um jeito de escrever, ilustrar, comunicar) para uso em projetos de sistemas.

Esta linguagem é expressa através de diagramas. Cada diagrama é composto por elementos (formas gráficas usadas para os desenhos) que possuem relação entre si.

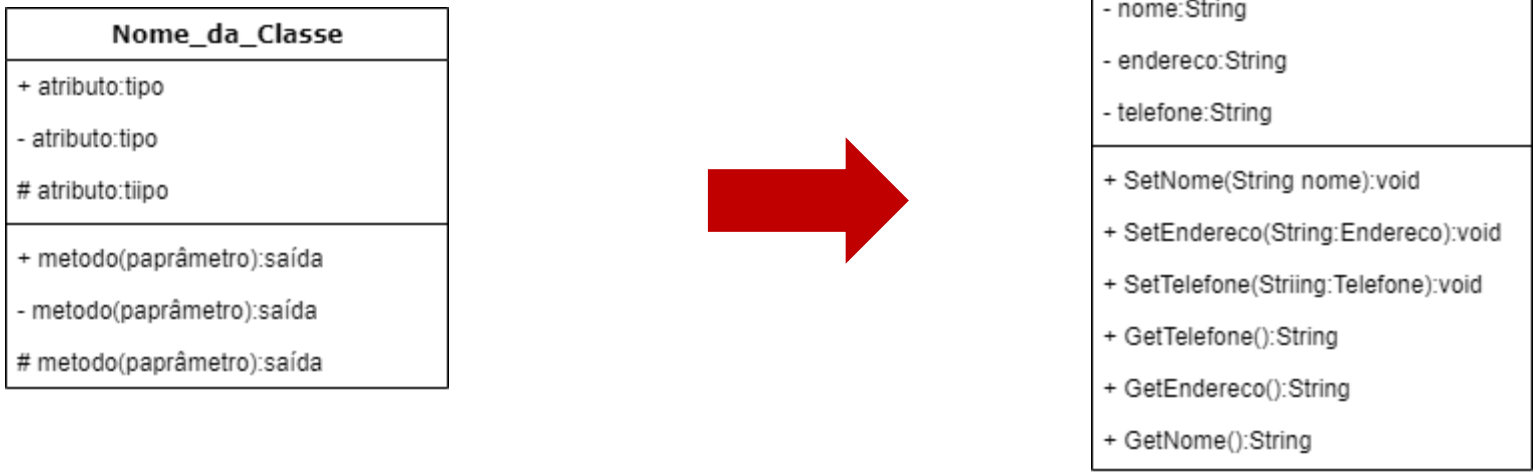
Diagrama de Classes - UML

Em programação, um diagrama de classes é uma representação da estrutura e relações das classes que servem de modelo para objetos. É uma modelagem muito útil para o desenvolvimento de sistemas, pois define todas as classes que o sistema necessita possuir.

Diagrama de Classes - UML



Exemplo


Nome_da_Classe
+ atributo:tipo
- atributo:tipo
atributo:tiipo
+ metodo(paprâmetro):saída
- metodo(paprâmetro):saída
metodo(paprâmetro):saída
Pessoa
- nome:String
- endereco:String
- telefone:String
+ SetNome(String nome):void
+ SetEndereco(String:Endereco):void
+ SetTelefone(Striing:Telefone):void
+ GetTelefone():String
+ GetEndereco():String
+ GetNome():String

OBRIGADO