INTRODUÇÃO AO POLIMORFISMO PHP-POO

"O Polimorfismo é um mecanismo por meio do qual selecionamos as funcionalidades utilizadas de forma dinâmica por um programa no decorrer de sua execução." (Devmedia)

"Com o Polimorfismo, os mesmos atributos e objetos podem ser utilizados em objetos distintos, porém, com implementações lógicas diferentes." (Devmedia)

Este é o último pilar do paradigma de Orientação a Objetos, procedendo, portanto, aqueles outros dois -encapsulamento e herança- abordados anteriormente. Algumas bibliográficas, vale citar, consideram a abstração um quarto-primeiro pilar, pois seria um a mais, mas existente antes do encapsulamento. No nosso caso, estamos considerando que este ao invés de um pilar, é um conceito do próprio encapsulamento.

Independentemente disto, **o polimorfismo é** um processo em que considerados alguns métodos bem definidos, estes, ao invés de utilizados em apenas uma classe ou outra, e sendo acessível a um objeto de uma mesma classe ou outra, pode ser adaptado dinamicamente para atender a quaisquer outras simultaneamente.

Sendo assim, **considere, para fins de exemplo**, a classe raíz "mamífero", seu método "movimentar()", que imprime que o objeto passou a se movimentar, e como, e também suas subclasses e descendentes. Assim como a própria classe raíz, as classes herdeiras poderiam ter acesso a este mesmo método, que seria inserido no arquivo fonte de cada uma, tendo, também, tarefas descritas diferentemente se necessário.

Poderiam haver as subclasses "humano" e "canguru", que apesar de serem dois mamíferos, não possuem estruturas e formas de locomoção semelhantes, e assim, ambas chamando o mesmo método "movimentar()", teriam saídas diferentes. Para o objeto da classe "canguru" seria apresentada a frase "começou a se movimentar por meio de pulos", e para o objeto da classe "humano" seria a frase "começou a se movimentar por meio de passos".

Poderíamos citar mais exemplos, como o de formas geométricas semelhantes. Se "quadrado" fosse uma classe abstrata, com um método "desenhar a si mesmo", e possuísse as subclasses "retângulo" e "losango", não poderíamos simplesmente chamar o mesmo método pelo nome em ambas as figuras, sem alterar as configurações da tarefa, pois apesar de serem todos igualmente quadriláteros, são diferentes. Do contrário, sem alterações nas configurações das tarefas do método, sempre que o retângulo, e/ou o losango, viessem a chamar o método de sua superclasse "quadrado", o que seria desenhado seria sempre um quadrado.

CONCEITO DE ASSINATURA DO MÉTODO

"*Em programação de computadores, especialmente em programação orientada a objetos, um **método** é normalmente identificado por sua **assinatura de método** única, que normalmente inclui o nome do **método** e o número, tipo e ordem de seus parâmetros*." (Wikipedia)

Diremos, portanto, considerando a frase acima, que **cada método possui uma assinatura**, ou uma espécie de identificação, não bastando somente seu próprio nome.

Por conseguinte, para que dois métodos possuam a mesma exata assinatura, devem compartilhar não apenas o nome, mas os parâmetros e seus tipos.

TIPOS DE POLIMORFISMO

Nas mais diversas bibliografias são abordados 6 tipos, mas apenas dois serão apresentados, por serem mais comumente utilizados.

Polimorfismo de Sobreposição (com métodos de mesma assinatura): É o conceito que fora utilizado na introdução ao polimorfismo, deste texto. Nos permite reescrever um método, ou seja, podemos reescrever nas classes filhas métodos criados inicialmente na classe pai, os métodos que serão sobrepostos, diferentemente dos sobrecarregados, devem possuir o mesmo nome, tipo de retorno e quantidade de parâmetros do método inicial, porém o mesmo será implementado com especificações da classe atual.

Polimorfismo de Sobrecarga (com métodos de diferentes assinaturas): A sobrecarga de métodos (overload) é um conceito do polimorfismo que consiste basicamente em criar variações de um mesmo método, ou seja, a criação de dois ou mais métodos com nomes totalmente iguais em uma classe. A Sobrecarga permite que utilizemos o mesmo nome em mais de um método contanto que suas listas de argumentos sejam diferentes para que seja feita a separação dos mesmos.