OOPs Concepts Java (Programação Orientada à Objetos - POO)

Resumindo por William Campos

Os conceitos de POO incluem abstração, encapsulamento, herança e polimorfismo.

Java define os conceitos de POO da seguinte forma:

- Abstração.
  Usando coisas simples para representar a complexidade. Todos sabemos como ligar a televisão, mas não precisamos de saber como ela funciona para a desfrutar. Em Java, abstração significa coisas simples como objetos, classes e variáveis que representam códigos e dados subjacentes mais complexos. Isto é importante porque permite evitar a repetição do mesmo trabalho várias vezes.

- Encapsulamento.
  A prática de manter os campos dentro de uma classe privada, dando depois acesso a esses campos através de métodos públicos. O encapsulamento é uma barreira protetora que mantém os dados e o código seguros dentro da própria classe. Podemos então reutilizar objetos como componentes de código ou variáveis sem permitir o acesso aberto a todo o sistema de dados.

- Herança.
  Uma característica especial da Programação Orientada à Objetos em Java, a Herança permite aos programadores criar novas classes que partilham alguns dos atributos das classes existentes. A utilização da Herança permite-nos construir sobre o trabalho anterior sem reinventar a roda.

- Polimorfismo.
  Permite aos programadores usar a mesma palavra em Java para significar coisas diferentes em contextos diferentes. Uma forma de polimorfismo é a sobrecarga de métodos. É aí que o próprio código implica significados diferentes. A outra forma é a sobreposição de métodos. É quando os valores das variáveis fornecidas implicam significados diferentes. Vamos aprofundar um pouco mais.

Como Funciona a Abstração

A abstração permite aos programadores criar ferramentas úteis e reutilizáveis. Por exemplo, um programador pode criar vários tipos diferentes de objetos, que podem ser variáveis, funções ou estruturas de dados. Os programadores podem também criar diferentes classes de objetos como formas de definir os objetos. Por exemplo, uma classe de variável pode ser um endereço. A classe pode especificar que cada objeto de endereço deve ter um nome, rua, cidade e código postal. Os objetos, neste caso, podem ser endereços de empregados, endereços de clientes ou endereços de fornecedores.

Como funciona o Encapsulamento

O encapsulamento permite-nos reutilizar a funcionalidade sem pôr em risco a segurança. É um conceito POO poderoso e que poupa tempo em Java. Por exemplo, podemos criar um pedaço de código que chama dados específicos a partir de uma base de dados. Pode ser útil reutilizar esse código com outras bases de dados ou processos. O encapsulamento permite-nos fazer isso, mantendo os nossos dados originais privados. Também nos permite alterar o nosso código original sem o quebrar para outros que entretanto o tenham adoptado.

Como Funciona a Herança

A herança é outro conceito de Java POO de economia de trabalho que funciona ao deixar uma nova classe adoptar as propriedades de outra. Chamamos à classe herdeira uma subclasse ou uma classe infantil. A classe original é muitas vezes chamada de pai. Usamos a palavra-chave estende-se para definir uma nova classe que herda propriedades de uma classe antiga.

Como funciona o Polimorfismo

O polimorfismo em Java funciona utilizando uma referência a uma classe dos pais para afetar um objeto na classe da criança. Poderíamos criar uma classe chamada "cavalo", alargando a classe "animal". Essa classe pode também implementar a classe "corrida profissional". A classe "cavalo" é "polimórfica", uma vez que herda atributos tanto da classe "animal" como da classe "corrida profissional". Mais dois exemplos de polimorfismo em Java são a sobreposição de métodos e a sobrecarga de métodos.

Isto permite a um programador utilizar um método de maneiras diferentes, dependendo de ser invocado por um objeto da classe pai ou por um objeto da classe filha.

Na sobrecarga de métodos, um único método pode desempenhar funções diferentes, dependendo do contexto em que é chamado.

Isto significa que um único nome de método pode funcionar de diferentes maneiras, dependendo dos argumentos que lhe são transmitidos.

Melhores Práticas para Conceitos OOP em Java

O objetivo dos conceitos OOP em Java é poupar tempo sem sacrificar a segurança e a facilidade de utilização. As seguintes melhores práticas estão todas orientadas para o avanço desse objetivo principal.

- **DRY (Don't Repeat Yourself - Não repita você mesmo)**. Um conceito central em Java, DRY significa simplesmente que nunca se deve ter dois blocos de código idêntico em dois locais diferentes. Se espera que o seu código Java mude no futuro, encapsule-o tornando todas as variáveis e métodos privados desde o início. medida que o código muda, aumente o acesso a "protegido" conforme necessário, mas não demasiado público.

- Single Responsability(Responsabilidade única). Este princípio das melhores práticas para conceitos POO em Java afirma que uma classe deve ter sempre apenas uma funcionalidade. Dessa forma, a classe pode ser chamada e/ou estendida por si só quando surgirem novas utilizações para ela, sem causar o acoplamento entre diferentes funcionalidades.

- Open Closed Design (Desenho Fechado Aberto). Fazer todos os métodos e classes Fechados para modificação, mas Abertos para uma extensão. Dessa forma o código testado pode permanecer estático mas pode ser modificado para realizar novas tarefas conforme necessário.

Saiba mais (sites em inglês):

OOPs Concepts Java: https://stackify.com/oops-concepts-in-java/#:~:text=The%20main%20ideas%20behind%20Java%27s,of%20them%20without%20compromising%20security.

OOPs Concepts Java 2: https://www.javatpoint.com/java-oops-concepts

Java inheritance: https://www.tutorialspoint.com/java/java_inheritance.htm

Java Polymorphism: http://www.sitesbay.com/java/java-polymorphism

10 Object-Oriented (OOP) Design Principles Java Programmers Should Know: https://javarevisited.blogspot.com/2018/07/10-object-oriented-design-principles.html#ixzz7hoHs09Sz: