POO com Java

Daniel Pereira

1 – Introdução

A **POO** ou Programação Orientada a Objetos, que em inglês é chamada de **OOP** (Object-Oriented Programming), tem como objetivo aproximar o mundo digital do mundo real.

Antigamente, a programação era realizada em baixo nível, sem comandos claros, utilizando linguagem de máquina, que variava entre os computadores. Isso tornava o trabalho dos engenheiros difícil e a programação muito trabalhosa.

Depois disso**,** surgiu a **Programação Linear** que focava em sequências lineares de instruções, tornando o código mais legível e fácil de seguir.

Depois surgiu a **Programação Estruturada**, que teve origem para organizar o código em estruturas lógicas, melhorou o desenvolvimento e a manutenção de sistemas mais complexos.

Com o crescimento dos sistemas, a Programação Estruturada começou a falhar em alguns conceitos, daí surgiu a **Programação Modular**, Este conceito permitiu que desenvolvedores criassem sistemas mais organizados e reutilizáveis, ajudando a lidar com a complexidade crescente dos projetos.

Em sequência, surgiu o paradigma de **POO**: A Programação Orientada a Objetos introduz a modelagem de entidades do mundo real como objetos. Os conceitos de encapsulamento (ocultar detalhes internos), herança (reutilizar e estender funcionalidades) e polimorfismo (tratar diferentes tipos de objetos de forma uniforme) são fundamentais para a POO e contribuem para a manutenção e escalabilidade do código. A Programação Orientada a Objetos foi desenvolvida na década de 1960, com contribuições significativas de vários pesquisadores. No entanto, o conceito é frequentemente associado a **Alan Kay**, que trabalhou no projeto Smalltalk na Xerox PARC. Smalltalk foi uma das primeiras linguagens a implementar os princípios da POO de forma consistente.

Antes da POO, a programação enfrentava problemas significativos devido ao uso extensivo de dados globais e procedimentos. Os dados globais geravam alto acoplamento entre partes do código, tornando-o difícil de manter, pois mudanças em uma parte poderiam impactar outras de maneira inesperada. Além disso, a lógica do programa se tornava confusa, especialmente com muitos procedimentos dependendo de variáveis globais, o que complicava a identificação de bugs. A reutilização de código era limitada, pois os procedimentos frequentemente eram escritos para contextos específicos, dificultando sua aplicação em outros lugares. Isso tornava

a escalabilidade de projetos maiores um desafio, devido à falta de organização e modularidade. A POO resolveu esses problemas ao encapsular dados e métodos em objetos, promovendo baixo acoplamento, maior facilidade de manutenção, reutilização de código através de herança e polimorfismo, e uma estrutura mais modular que facilita a organização e escalabilidade do código.

2 – Como funciona a POO?

POO é um paradigma que organiza o software em torno de "objetos", que são instâncias de "classes". Esses objetos encapsulam dados e comportamentos relacionados, permitindo uma modelagem mais intuitiva e modular. Dentre os principais conceitos estão:

1. Classes e Objetos:

Classe: É um molde ou template que define as características (atributos) e comportamentos (métodos) dos objetos. Por exemplo, uma classe "Carro" pode ter atributos como "cor" e "modelo" e métodos como "acelerar" e "frear".

Objeto: É uma instância de uma classe. Cada objeto pode ter valores específicos para os atributos definidos na classe. Por exemplo, um objeto "carro1" pode ser um "Carro" de cor "vermelha" e modelo "Fusca".

2. Encapsulamento: Este conceito se refere à prática de ocultar os detalhes internos de um objeto e expor apenas o que é necessário. Os atributos de um objeto são geralmente privados e acessados através de métodos públicos (getters e setters). Isso ajuda a proteger o estado interno do objeto e a controlar como ele é modificado.

3. Herança: Permite que uma classe (subclasse) herde atributos e métodos de outra classe (superclasse). Isso promove a reutilização de código. Por exemplo, uma classe "Caminhão" pode herdar de "Veículo", assim compartilhando características comuns, mas também pode ter métodos ou atributos adicionais.

4. Polimorfismo: Permite que diferentes classes sejam tratadas como instâncias da mesma classe base, principalmente através de métodos com o mesmo nome, mas implementações diferentes. Isso é útil em situações onde uma função pode operar em objetos de diferentes classes. Por exemplo, se "Carro" e "Caminhão" têm um método "dirigir", você pode chamar "dirigir" em qualquer um desses objetos, e o comportamento adequado será executado.

5. Abstração: A abstração permite focar nas características essenciais de um objeto, ignorando os detalhes menos relevantes. Isso é frequentemente implementado através de classes abstratas e interfaces, que definem métodos que devem ser implementados pelas classes que herdam ou que implementam essas interfaces.

3 – Downloads e Instalações Necessárias

01 - Começando pelo Java SE JDK, a versão atual (setembro/2024) é a JDK Development Kit 23
https://www.oracle.com/br/java/technologies/downloads

Windows x64 Installer
https://download.oracle.com/java/23/latest/jdk-23_windows-x64_bin.exe

Importante instalar a JDK antes do Netbeans! É recomendado instalar tudo padrão sem mudar o caminho da instalação.

02 - Em seguida o Netbeans, a versão atual (setembro/2024) é a NetBeans 23
https://netbeans.apache.org/front/main/download/nb23

Windows x64 Installer
https://www.apache.org/dyn/closer.lua/netbeans/netbeans-installers/23/Apache-NetBeans-23-bin-windows-x64.exe

É recomendado instalar tudo padrão sem mudar o caminho da instalação.

03 - Por último, instalar o JavaFX Scene Builder. Será necessário uma conta Oracle para fazer o download
www.oracle.com/java/technologies/javafxscenebuilder-1x-archive-downloads.html

Instalar tudo por padrão. Importante ser instalado depois do NetBeans!

04 - Caso o JavaFX não funcione ou tenha algum problema relacionado com o JavaFX FXML, recomenda-se instalar o JDK8

https://www.oracle.com/br/java/technologies/javase/javase8-archive-downloads.html

05 - Caso tenha algum problema para executar os arquivos .jar com JavaFX, talvez seja necessário atualizar a JRE.