POR CRISTIANE PAGINE

JAVA

INTRODUÇÃO À PROGRAMAÇÃO ORIENTADA A OBJETOS EM JAVA

2025

Tecnologias Java

São suportes para desenvolvimento de aplicações com diferentes funcionalidades em Java. Todas se baseiam nos mesmos conceitos: bytecodes e JVM.

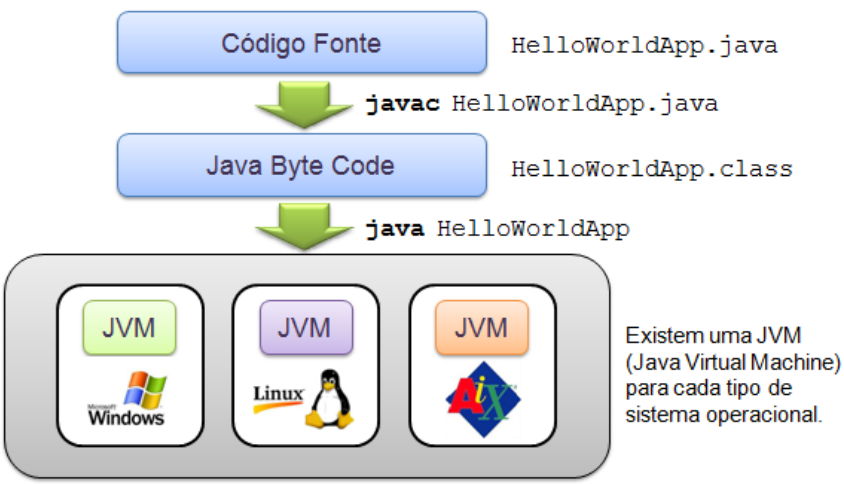
- **J2SE (Standard Edition):** O J2SE é utilizado para fazer softwares desktop, que operam na tela do computador normalmente. Ele consiste em codificar as classes do programa que ao ser compilado gera um arquivo JAR que funciona como executável da aplicação em qualquer computador que tenha a JVM instalada.

- **J2EE (Enterprise Edition):** O J2EE é utilizado para desenvolver aplicações que rodam na Web, por exemplo um sistema financeiro que funciona a partir de um site web. Ou seja é uma forma de criar páginas HTML que se comunicam com classes Java, formando assim a lógica da aplicação.

- **J2ME (Micro Edition):** O J2ME serve para desenvolver aplicações para telefones celulares, palmtops e outros dispositivos móveis.

Compilação e Execução

1. No processo de compilação, ao invés do programa ser compilado para código de máquina da plataforma que vai ser executado, o programa é compilado para bytecode
2. O bytecode é genérico, isto é, não é específico para nenhum sistema operacional em particular
3. Quando um programa Java é executado, o arquivo bytecode é interpretado pelo interpretador da tecnologia java, que é denominado Java Virtual Machine. Existe uma JVM diferente para cada plataforma onde a tecnologia Java pode ser executada e deverá existir uma instalada no computador no qual será executado um programa Java. Os browsers, por exemplo, incorporam uma JVM para a execução de applets.



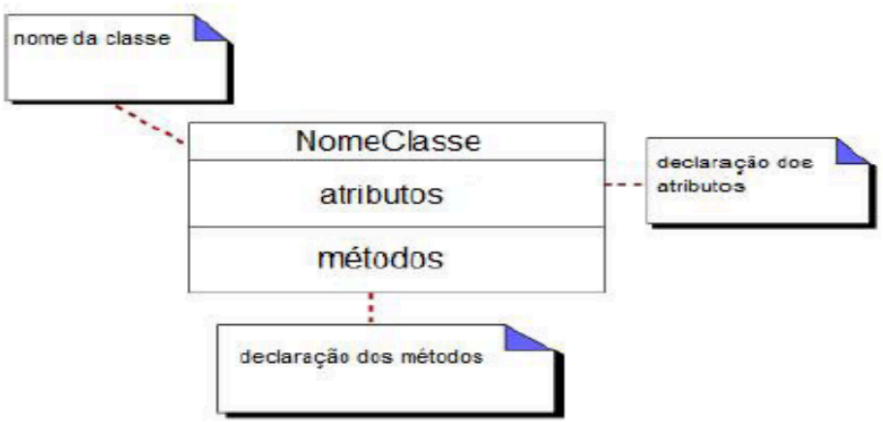
Introdução POO

A Programação Orientada a Objetos (POO) é um paradigma que imita a forma como pensamos e organizamos o mundo real. Em Java, a POO é amplamente utilizada devido à sua abordagem intuitiva e robusta. Este guia apresenta conceitos, exemplos práticos e exercícios para você dominar os fundamentos da POO em Java.

1. Conceitos Fundamentais
Classe e Objeto:

- Uma classe é um modelo ou estrutura que define as propriedades (atributos) e comportamentos (métodos) de um tipo de objeto.
- Um objeto é uma instância de uma classe.

Anatomia de uma classe:



Exemplo de uma classe:

```
public class Pessoa {
    private String nome;
    private int idade;

    // Construtor
    public Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }

    // Métodos
    public void apresentar() {
        System.out.println("Oi, meu nome é " + nome +
" e tenho " + idade + " anos.");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Pessoa p = new Pessoa("Ana", 25);
        p.apresentar();
    }
}
```

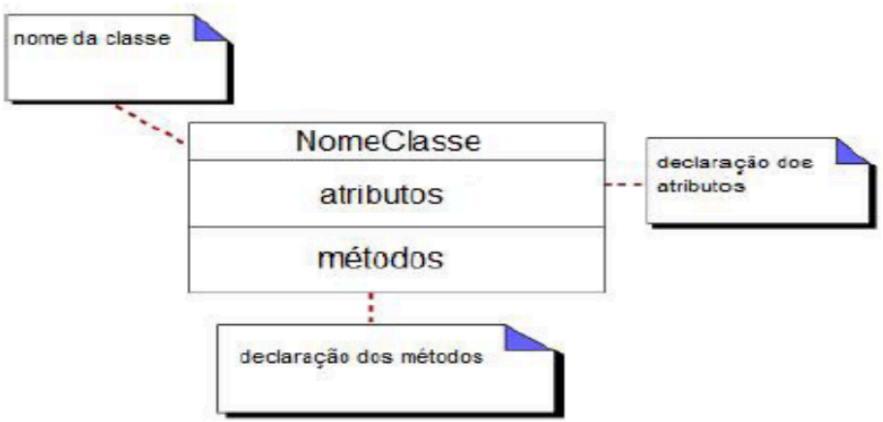
A Programação Orientada a Objetos (POO) é um paradigma que imita a forma como pensamos e organizamos o mundo real. Em Java, a POO é amplamente utilizada devido à sua abordagem intuitiva e robusta. Este guia apresenta conceitos, exemplos práticos e exercícios para você dominar os fundamentos da POO em Java.

1. Conceitos Fundamentais

Classe e Objeto:

- Uma classe é um modelo ou estrutura que define as propriedades (atributos) e comportamentos (métodos) de um tipo de objeto.
- Um objeto é uma instância de uma classe.

Anatomia de uma classe:



Nome da Classe: O nome da classe deve começar com uma letra maiúscula e seguir a convenção CamelCase. Exemplo: public class Carro.

Atributos (Membros): São variáveis que representam as características do objeto. Eles definem o estado do objeto e são declarados dentro da classe, mas fora dos métodos.

Exemplo:

```
public class Carro {
    String cor;
    String modelo;
    int ano;
}
```

Métodos: São blocos de código que definem o comportamento do objeto. Eles podem ser públicos, privados, protegidos ou padrão e podem ou não retornar um valor.

Exemplo:

```
public class Carro {
    String cor;
    String modelo;
    int ano;

    void acelerar() {
       System.out.println("O carro está acelerando.");
    }
}
```

Construtores: São métodos especiais chamados quando um objeto da classe é criado. Eles inicializam os atributos do objeto.

Exemplo:

```
public class Carro {
    String cor;
    String modelo;
    int ano;

    Carro(String cor, String modelo, int ano) {
        this.cor = cor;
        this.modelo = modelo;
        this.ano = ano;
    }
}
```

Modificadores de Acesso: Determinam a visibilidade dos membros da classe. Existem quatro modificadores de acesso em Java: public, private, protected e default. Eles controlam quem pode acessar e modificar os membros da classe.

Classes e Objetos: Uma classe é uma definição abstrata de um tipo de objeto, enquanto um objeto é uma instância específica dessa classe. Várias instâncias (objetos) de uma mesma classe podem ser criadas em um programa.

O Método main em Aplicações Java Standalone

O método main é o ponto de entrada de qualquer aplicação Java. É por ele que a Máquina Virtual Java (JVM) inicia a execução do programa. Sem o método main, a JVM não sabe por onde começar.

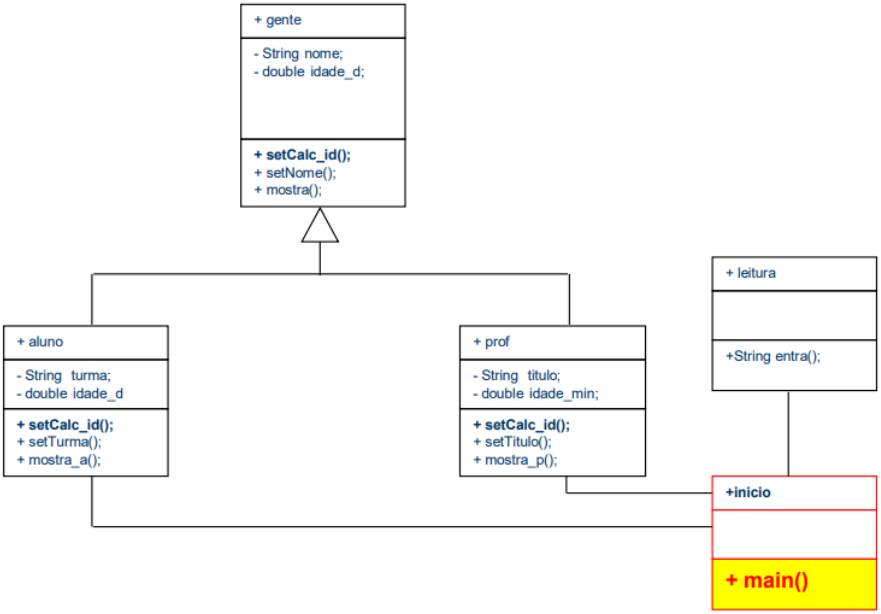
Estrutura do Método main:

```
public static void main(String[] args) {
// Código a ser executado
}
```

- public: Torna o método acessível de qualquer lugar.
- static: Permite que o método seja chamado sem a necessidade de instanciar um objeto da classe.
- void: Indica que o método não retorna nenhum valor.
- main: É o nome específico do método que a JVM procura.
- String[] args: É um array de strings que contém os argumentos passados para o programa na linha de comando.

Resumindo:
O método main(): Responsável pelo “start” do sistema.
Todas a classes podem ter o método main(), porém caso seja um conjunto de classes que fazem parte de um mesmo sistema, apenas uma será responsável pelo start do aplicativo.

A seguir um exemplo:



Pontos sobre Java a considerar ao desenvolver uma classe:

- Java reconhece letras maiúscula e minúscula (sensitive case).
- O nome do arquivo ".java" de ser o mesmo nome da classe (descrito internamente dentro do código).

```
import java.io.*;
public class Saida {
        public static void main(String arg[]){
System.out.println("Olá Mundo !");
    }
}
```

Um pouco sobre Garbage Collector

O Garbage Collector é uma funcionalidade essencial em Java que automatiza a gestão da memória, liberando o espaço ocupado por objetos que não estão mais sendo utilizados pelo programa. Essa automação evita que o desenvolvedor precise se preocupar com a alocação e desalocação manual de memória, o que poderia levar a erros como vazamentos de memória. O processo de coleta de lixo geralmente envolve a marcação dos objetos ainda em uso e a subsequente remoção dos objetos não marcados. A JVM (Máquina Virtual Java) divide a memória em gerações para otimizar esse processo, coletando com mais frequência objetos mais recentes. Ao compreender o funcionamento do Garbage Collector, o programador pode escrever código mais eficiente e evitar problemas de desempenho relacionados à memória.