Orientação a Objetos

Programação Orientada a Objetos

A Programação Orientada a Objetos (**POO**) é um paradigma de programação que organiza o código em torno de **objetos**, que são instâncias de **classes**. Esses objetos representam entidades do mundo real e possuem atributos (características) e métodos (comportamentos).

A ideia central da POO é modelar o software de forma que ele reflita as características e interações dos objetos no mundo real. Isso facilita a organização do código, a reutilização e a manutenção do sistema.

É baseada em quatro pilares fundamentais.

Pilares da Orientação a Objetos

- Abstração
- Encapsulamento
- Herança
- Polimorfismo

Pilares da Orientação a Objetos

O pilar de **abstração** é um princípio que permite a criação de modelos simplificados de objetos do mundo real. Utilizamos esse pilar para identificar e modelar as características e comportamentos essenciais de um objeto, ignorando detalhes irrelevantes ou secundários.

O **encapsulamento** restringe o acesso direto aos atributos de um objeto, permitindo que apenas métodos específicos manipulem esses dados. Isso protege a integridade do objeto e evita que dados sejam alterados indevidamente.

Pilares da Orientação a Objetos

A **herança** permite que uma classe herde atributos e métodos de outra, promovendo a reutilização de código.

O **polimorfismo** permite que um mesmo método tenha comportamentos diferentes dependendo do objeto que o utiliza. Ele pode ocorrer de duas formas:

- Polimorfismo de *sobrecarga*: Métodos com o mesmo nome, mas assinaturas diferentes.
- Polimorfismo de *sobrescrita*: Uma subclasse redefine um método da superclasse.

Benefícios da Orientação a Objetos

- Organização do código
- Reutilização
- Facilidade de manutenção
- Escalabilidade
- Modularidade

Classes e Objetos

Uma **classe** pode ser entendida como um **modelo** ou **estrutura** que define atributos (características) e métodos (comportamentos) para representar uma entidade do mundo real. Os atributos representam como esse entidade *é*, já os métodos representam o que a entidade *faz*.

Um **objeto** é uma instância de uma classe, ou seja, uma representação concreta baseada nesse modelo. No mundo real, por exemplo, podemos pensar em uma empresa onde cada funcionário possui um nome, um cargo e um salário. Em Java, isso poderia ser representado por uma classe Funcionario, que define esses atributos e os comportamentos relacionados. A partir dessa classe, podemos criar diferentes objetos, como um funcionário chamado João, que ocupa o cargo de desenvolvedor e recebe um salário específico, e outro chamado Maria, que trabalha como gerente, cada um com seus próprios valores atribuídos.

Criando a classe e os atributos

```
public class Funcionario {
    no usages
    String nome;
    no usages
    String cargo;
    no usages
    double salario;
}
```

Criação da classe **Funcionario** com a definição dos três atributos que são comuns a todos os funcionários, que são, por exemplo: *nome*, *cargo* e *salario*

Criando um objeto a partir da classe Funcionario

```java
Funcionario desenvolvedor = new Funcionario();
desenvolvedor.nome = "João";
desenvolvedor.cargo = "Desenvolvedor Backend Java";
desenvolvedor.salario = 8500;
```

Como ainda não aplicamos o pilar de **encapsulamento**, essa é a forma que atribuímos valor aos atributos de cada instância.

Criação da instância **desenvolvedor**, que terá como atributos os itens definidos pelo modelo (classe) Funcionario, que são: *nome*, *cargo* e *salario*.

Criando métodos na classe Funcionario

```java
public void exibirInformacoes() {
    System.out.printf("\nFuncionario %s - Salário: %.2f",
            nome, salario);
}
```

Método sem retorno e sem parâmetro criado na classe **Funcionario**, que poderá ser chamado por todas as instâncias criadas a partir da mesma

Criando métodos na classe Funcionario

```java
public void reajustarSalario(double percentual) {
    salario += salario * (percentual / 100);
    System.out.printf("\nNovo salario de %s é %.2f ", nome, salario);
}
```

Método sem retorno e com um parâmetro criado na classe **Funcionario**, que poderá ser chamado por todas as instâncias criadas a partir da mesma

Objetos utilizando os métodos

```
desenvolvedor.exibirInformacoes();
desenvolvedor.reajustarSalario(5);
```

Para que a instância chame o método, basta referenciá-lo no formato descrito.

Exemplo da saída no **terminal**.

```
Funcionario João - Salário: 8500,00
Novo salario de João é 8925,00
```

Orientação a objetos com classes, atributos e métodos

O uso de **classes**, **atributos** e **métodos** na programação orientada a objetos em Java traz organização, reutilização e modularidade ao código.

Classes permitem estruturar o sistema de forma clara, agrupando dados (atributos) e comportamentos (métodos) relacionados.

Isso facilita a manutenção, evita repetição de código e melhora a legibilidade. Além disso, a criação de objetos a partir de classes permite modelar o mundo real de forma intuitiva, promovendo encapsulamento e reutilização eficiente do código.

Compartilhe um resumo de seus novos conhecimentos em suas redes sociais.
#aprendizadoalura

alura | **Escola** Programação