02

# Aplique princípios de encapsulamento a uma classe

A **assinatura** de um método é o que realmente deve importar para o usuário de alguma classe. Segundo os bons princípios do encapsulamento, a implementação dos métodos deve estar *encapsulada* e não deve fazer diferença para o usuário.

O que é importante em uma classe é **o que ela faz** e não **como ela faz**. *O que ela faz* é definido pelos comportamentos expostos, ou seja, pelos métodos e suas assinaturas.

O conjunto de assinaturas de métodos visíveis de uma classe é chamado de **interface de uso**. É através dessas operações que os usuários vão se comunicar com os objetos dessa classe.

Mantendo os detalhes de implementação de nossas classes "escondidos", evitamos que mudanças na forma de implementar uma lógica quebre vários pontos de nossa aplicação.

Uma das formas mais simples de começar a encapsular o comportamento de uma classe é escondendo seus atributos. Podemos fazer isso facilmente usando a palavra-chave `private`:

```
public class Pessoa{
    private String nome;
}
```

COPIAR CÓDIGO

Caso precisemos acessar um desses atributos a partir de outra classes, teremos que criar um método para liberar o acesso de leitura desse atributo. Seguindo a especificação dos *javabeans*, esse método seria um *getter*. Da mesma forma , se precisarmos liberar a escrita de algum atributo, criamos um método *setter* :

```
public class Pessoa{
    private String nome;

    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

COPIAR CÓDIGO

Com essa abordagem, poderíamos fazer uma validação em nossos métodos, para evitar que nossos atributos fiquem com estado inválido. Por exemplo, podemos verificar se o nome possui pelo menos 3 caracteres:

```
public class Pessoa{
    private String nome;
    private String sobrenome;

    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        if(nome!= null && nome.trim().length() >= 3)
            this.nome = nome;
```

```
        else{
            throw new IllegalArgumentException(
             "Nome deve possuir " + "pelo menos 3
caracteres");
        }
    }
}
```

COPIAR CÓDIGO

Encapsulamento é muito mais do que atributos privados e *getters* e *setters* . Não é nosso foco aqui discutir boas práticas de programação, e sim o conhecimento necessário para passar na prova. Em questões sobre encapsulamento sempre, fique atento à alternativa que esconde mais detalhes de implementação da classe analisada. A prova pode utilizar tanto o termo *encapsulation* como *information hiding* para falar sobre encapsulamento (ou esconder informações).