POR CRISTIANE PAGINE

JAVA

ENCAPSULAMENTO

2025

Introdução

O encapsulamento é um dos pilares da Programação Orientada a Objetos (POO) e tem como objetivo proteger os dados de uma classe, restringindo o acesso direto a eles e permitindo que sejam manipulados apenas por métodos específicos. Esse conceito promove maior segurança, modularidade e manutenção do código, garantindo que os atributos de um objeto não sejam alterados de forma inesperada.

Em Java, o encapsulamento é implementado utilizando modificadores de acesso (private, protected, public) e métodos getters e setters, que controlam a leitura e a modificação dos atributos. Dessa forma, uma classe pode esconder sua implementação interna e expor apenas o necessário para outras partes do programa, seguindo o princípio do baixo acoplamento e alta coesão.

Benefícios do Encapsulamento

✅ Proteção de dados → Os atributos da classe não podem ser acessados ou modificados diretamente.

✅ Facilidade de manutenção → As regras de negócio podem ser alteradas sem impactar outras partes do código.

✅ Maior controle sobre os dados → Permite validar informações antes de serem modificadas.

✅ Redução de dependências → Diminui o acoplamento entre classes, tornando o código mais modular.

O encapsulamento é essencial para garantir um código seguro, organizado e fácil de manter, sendo amplamente utilizado no desenvolvimento de sistemas orientados a objetos em Java.

Encapsulamento (definição): Possibilidade de se ocultar detalhes da implementação de uma classe.

Comportamento: será utilizado somente aquilo que o programador da classe permitir.

Manutenção: mantendo os mesmos serviços da classe, podemos alterar sua estrutura interna da classe (refinamentos de código, por exemplo) sem que outras classes (códigos) que dependam dela tenham que ser alteradas.

Segurança: Não se consegue corromper o estado de um objeto por distração, pois partes deste objeto está protegido pelo encapsulamento

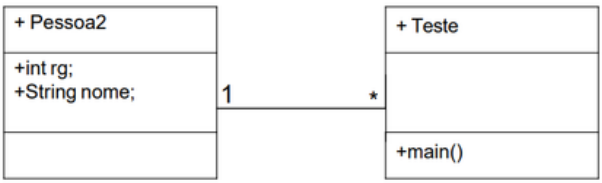
Pode ocorrer em 3 níveis de especificação de acesso:
PÚBLICO (public) (+): todos têm acesso. Um atributo pode ter seu valor alterado a partir de qualquer outro código, mesmo sendo este de uma classe qualquer.

PROTEGIDO (protected) (#): em Java tem acesso quem está no mesmo pacote ou classes que herdem a classe que contenha atributo ou método protegido

PRIVADO (private) (-): Restrição total fora da classe. Só têm acesso membros da própria classe

Enquanto programadores de uma classe podemos gerenciar o acesso a seus membros de acordo com a necessidade, porém existe Convenção que determina esta gerência. O mais usual é: **Atributos privados e Métodos públicos.**

Observe o Diagrama de Classes a seguir. Depois observe o código.




```
public class Pessoa2 {
  public int rg;
  public String nome;

  public void mostraDados(){
   System.out.println("\n RG: "+rg);
   System.out.println("\n Nome: "+nome);
  }
}
```

```
public class Teste {
  public static void main(String arg[]){
    Pessoa2 p3 = new Pessoa2();
    p3.rg=50; //acessa o atributo RG diretamente
(ERRADO)
    p3.nome = "amor"; //acessa o atributo NOME
diretamente (ERRADO)
    p3.mostraDados();
  }
}
```

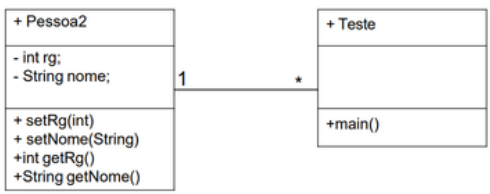
No exemplo acima pela convenção os atributos rg e nome deveriam ser privados e não acessados diretamente.

Encapsulamento e os Métodos getters e setters (definição)

set: possibilita alterar o estado de um objeto permitindo a alteração de valores de seus atributos através da passagens de parâmetros;

get: recupera valor de um atributo através do "retorno" do mesmo.

Observe o Diagrama de Classes a seguir. Depois observe o código:



Nota: lembre-se...
- privado (private)
+ publico (public)

```
public class Pessoa2 {
  private int rg;
  private String nome;
  public void setRg(int rg){
    this.rg=rg;
  }
  public void setNome(String nome){
    this.nome=nome;
  }
  public int getRg(){
    return rg;
  }
  public String getNome(){
    return nome;
  }
}
```

```
public class Teste {
  public static void main(String arg[]){
    Pessoa2 p3 = new Pessoa2();
    p3.setRg(50);
    p3.setNome("amor");
    System.out.println("\n RG: "+p3.getRg() );
System.out.println("\n NOME: "+p3.getNome() );
  }
}
```

A classe Pessoa2:

Representa uma pessoa com os atributos privados rg e nome, acessíveis apenas por métodos públicos:

- setRg(int rg) e setNome(String nome) → Definem os valores.
- getRg() e getNome() → Retornam os valores.

O encapsulamento protege os dados, permitindo acesso controlado.

A classe Teste:

No método main:

1. Cria um objeto p3 de Pessoa2.
2. Define RG e nome (setRg(50), setNome("amor")).
3. Exibe os valores armazenados (getRg(), getNome()).

Demonstra a aplicação do encapsulamento no controle de acesso aos dados.