Modificadores de Acesso

A programação orientada a objetos surgiu como uma alternativa de organização do código de forma que seu desenvolvimento e manutenção fossem a mais próximas o possível das entidades/objetos que elas representam no mundo real. Se um programa em questão é para uma fábrica de carros, por exemplo, é muito provável que na modelagem deste programa existam as classes Carro, Porta, Roda, Volante... Se for para um supermercado, é provável que existam as classes Carrinho, Produto, Caixa... e assim por diante.

No Java, todos os objetos são representados por **classes**. Nessas classes, ficam armazenados todos os comportamentos e características que esta entidade conterá. Esses comportamentos e características são formadas por **métodos e atributos** (às vezes até mesmo outras classes internas), respectivamente. Os **objetos** em si são instâncias de classes, ou seja, os objetos são o que dão vida a estrutura projetada em uma classe. Fazendo uma analogia, a classe seria como a planta baixa de uma casa e o objeto a sua construção. Ou seja, com um único planejamento, é possível ter várias réplicas de uma casa. Um diz respeito ao planejamento e o outro a execução.

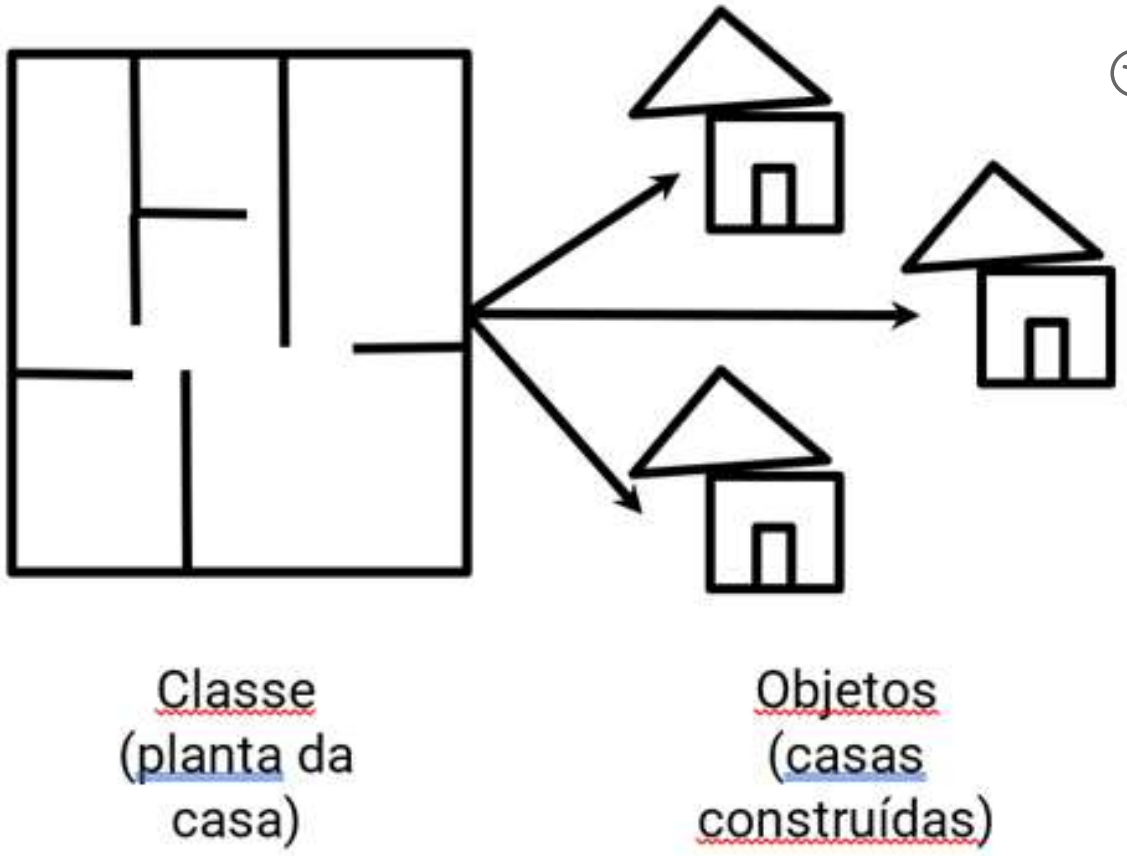

Figura 1 - Analogia da relação entre uma classe e um objeto.

Para criar (instanciar) um novo objeto, é usado a palavra reservada ***new*** acompanhada do método construtor da classe (que por sua vez, pode conter parâmetros). Por exemplo:

```
String exemplo = new String("Exemplo");
```

Neste código, a variável *exemplo* é um novo objeto da classe *String*. A partir deste ponto, é possível usar a variável com todos os métodos disponíveis na implementação da classe. Este exemplo foi com uma classe já implementada e disponibilizada pelo Java, mas é possível criar qualquer classe utilizando a palavra

reservada ***class***. No exemplo abaixo, uma classe Produto é criada com um atributo e um método.

```
public class Produto {

  private String nome;

  public String getNome() {
    return this.nome;
  }

}
```

Além dos conceitos de classe e objeto, a programação orientada a objetos conta com um outro conceito muito importante chamado **encapsulamento.** O conceito de encapsulamento está associado à forma como os atributos e métodos de uma classe, como os criados no exemplo anterior, são visíveis para as demais classes. A ideia deste conceito é fazer da classe uma pequena cápsula, ou seja, encapsular os dados de modo a restringir o acesso ao seu conteúdo.

O encapsulamento é importante pois se dentro de uma classe todos os seus métodos e atributos fossem acessíveis por todas as outras classes, seria praticamente impossível garantir a segurança de uma aplicação e, além disso, bastante improvável de estruturar o código de forma organizada, afinal, todas as pessoas envolvidas em um mesmo projeto poderiam usar todos os métodos de todas classes livremente, mesmo que não tenham sido pensadas para serem usadas desta forma. Em resumo, a literatura aponta três grandes benefícios do uso do encapsulamento:

- Proteção dos atributos da classe de acessos indevidos ou acidentais;

- Possibilidade de definir regras para alteração dos valores mantidos pelos atributos;

- Possibilidade de limitar as operações e alterações realizadas pelos elementos que estão acessando a classe.

Para fazer uso destes benefícios, o Java oferece alguns níveis de visibilidade para seus elementos. Esses níveis são chamados de **modificadores de acesso.** Os modificadores disponíveis são:

Nível de visibilidade	O que faz?
public	Torna acessível a outros objetos de qualquer classe.
private	• Não se aplica a classes. • Atributos só podem ser acessados por objetos da mesma classe. • Métodos só podem ser chamados por métodos da própria classe.
protected	• Não se aplica a classes. • Atributos e métodos são acessíveis dentro da própria classe, das subclasses e outros que façam parte do mesmo pacote.
nada especificado	• Classe é visível somente por classes do mesmo pacote • Atributos e métodos são acessíveis somente dentro das • classes que pertencem ao mesmo pacote. Este modo de acesso é também chamado de *default*.

Independente do tipo de visibilidade, uma vez que a classe é instanciada, seus métodos e atributos são acessíveis através do operador de ponto (ponto final). No exemplo abaixo, a classe Pessoa invoca o seu próprio método e seus atributos no método construtor, ou seja, o método que é invocado quando a classe vai criar(instanciar) um novo objeto:

```
public class Pessoa {

  private String nome;
  private Integer idade;

  public Pessoa(String nome, Integer idade) {
    this.setNome(nome);
    this.setIdade(idade);
  }

  public String getNome() {
    return nome;
  }

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

  public Integer getIdade() {
    return idade;
  }

  public void setIdade(Integer idade) {
    this.idade = idade;
  }
}
```

Nesta classe Pessoa, há dois atributos marcados como privados (*private*). Eles são: nome e idade. Sendo privados, eles são acessíveis somente dentro da própria classe. Por outro lado, dentro do construtor da classe há dois métodos sendo chamados: *setNome()* e *setIdade()*. Esses métodos evitam que seja feito o acesso direto aos atributos da classe e os torna disponíveis para que outras classes possam

modificá-las. Ou seja, desta forma é possível ter atributos que são privados e jamais vistos por outras entidades.

Atividade Extra

- Artigo: Getters and Setters in Java Explained
- Artigo: Java Modifiers

Referência Bibliográfica

BARNES, D. J. KOLLING, M. **Programação orientada a objetos com java: uma introdução prática usando o bluej.** 4.ed. Pearson: 2009.

FELIX, R. (Org.). **Programação orientada a objetos.** Pearson: 2017.

MEDEIROS, L. F. de. **Banco de dados: princípios e prática.** Intersaberes: 2013;

ORACLE. Java Documentation, 2021. **Documentação oficial da plataforma Java.** Disponível em: < https://docs.oracle.com/en/java/ >.

Ir para exercício