

Visibilidades e Modificadores

Controle de acesso e modificadores de classes, atributos e métodos em Java

1. Introdução

Em Java, os **modificadores** são palavras-chave usadas para definir características especiais de classes, atributos, métodos e construtores. Eles se dividem, principalmente, em duas categorias: os **modificadores de acesso (visibilidade)**, que controlam quem pode enxergar e utilizar um determinado elemento, e os **modificadores de não-acesso**, que alteram o comportamento do elemento sem interferir diretamente na visibilidade (como `static`, `final` e `abstract`).

O uso correto desses modificadores é fundamental para aplicar o encapsulamento, proteger os dados internos de uma classe e definir claramente a interface pública de um sistema.

2. Modificadores de Acesso (Visibilidade)

Java define quatro níveis de visibilidade, aplicáveis a classes, atributos, métodos e construtores: `public`, `protected`, `default` (também chamado de *package-private*, quando nenhum modificador é escrito) e `private`. A tabela a seguir resume o alcance de cada um.

Modificador	Mesma classe	Mesmo pacote	Subclasse (outro pacote)	Qualquer lugar
<code>private</code>	Sim	Não	Não	Não
<code>default</code> (sem modificador)	Sim	Sim	Não	Não
<code>protected</code>	Sim	Sim	Sim	Não
<code>public</code>	Sim	Sim	Sim	Sim

2.1. `private`

O modificador `private` restringe o acesso apenas à própria classe. É o nível mais restritivo e é amplamente usado para proteger atributos, aplicando o princípio do encapsulamento — o acesso externo é feito somente por meio de métodos públicos (*getters* e *setters*).

```

public class ContaBancaria {
    private double saldo; // só acessível dentro da própria classe

    public double getSaldo() {
        return saldo;
    }

    private void registrarLog(String operacao) {
        // método auxiliar interno, não deve ser exposto
        System.out.println("Log: " + operacao);
    }
}

// Em outra classe:
// conta.saldo;           // ERRO de compilação
// conta.registrarLog(x); // ERRO de compilação

```

2.2. default (package-private)

Quando nenhum modificador é especificado, o elemento tem visibilidade **default** (também chamada de *package-private*): ele pode ser acessado por qualquer classe do **mesmo pacote**, mas não por classes de outros pacotes.

```

// Arquivo: br.com.empresa.Funcionario.java
package br.com.empresa;

class Funcionario {           // classe com visibilidade default
    String cargo;              // atributo com visibilidade default

    void exibirCargo() {       // método com visibilidade default
        System.out.println("Cargo: " + cargo);
    }
}

// Outra classe do MESMO pacote br.com.empresa pode acessar Funcionario normalmente.
// Uma classe de um pacote DIFERENTE não consegue enxergar essa classe.

```

2.3. protected

O modificador **protected** permite o acesso pelas classes do mesmo pacote e também por subclasses localizadas em pacotes diferentes, sendo muito utilizado em hierarquias de herança quando se deseja que apenas as classes filhas tenham acesso a determinado atributo ou método.

```

public class Animal {
    protected String nome;

    protected void emitirSom() {
        System.out.println(nome + " faz um som genérico.");
    }
}

public class Cachorro extends Animal {
    public void latir() {
        // acesso permitido: Cachorro é subclasse de Animal
        System.out.println(nome + " está latindo!");
        emitirSom();
    }
}

```

2.4. public

O modificador `public` é o menos restritivo: o elemento pode ser acessado de qualquer lugar do programa, independentemente do pacote. É normalmente usado para expor a interface pública de uma classe — os métodos que outras partes do sistema devem utilizar.

```

public class Calculadora {
    public int somar(int a, int b) {
        return a + b;
    }
}

// Em qualquer outra classe, de qualquer pacote:
Calculadora calc = new Calculadora();
int resultado = calc.somar(5, 3); // acesso livre, pois somar() é public

```

3. Modificadores de Não-Acesso

Além dos modificadores de acesso, Java oferece outros modificadores que alteram o comportamento de classes, atributos e métodos, sem estarem relacionados à visibilidade. Os mais utilizados são `static`, `final` e `abstract`.

3.1. static

O modificador `static` indica que um atributo ou método pertence à **classe**, e não a uma instância específica. Isso significa que ele é compartilhado por todos os objetos e pode ser acessado sem a necessidade de criar um objeto.

```

public class ContadorDeUsuarios {
    private static int totalUsuarios = 0; // compartilhado por todos os objetos

    public ContadorDeUsuarios() {
        totalUsuarios++;
    }

    public static int getTotalUsuarios() {
        return totalUsuarios;
    }
}

// Uso:
new ContadorDeUsuarios();
new ContadorDeUsuarios();
System.out.println(ContadorDeUsuarios.getTotalUsuarios()); // 2

```

3.2. final

O modificador `final` impede alterações posteriores. Aplicado a um atributo, impede que seu valor seja modificado após a inicialização (constante); aplicado a um método, impede que ele seja sobrescrito por subclasses; aplicado a uma classe, impede que ela seja estendida (herdada).

```

public final class Configuracao {           // classe que não pode ser estendida
    public static final double PI = 3.14159; // constante, não pode mudar

    public final void carregar() {           // método que não pode ser sobrescrito
        System.out.println("Configuração carregada.");
    }
}

// PI = 3.0;                               // ERRO: não é possível alterar uma variável final
// class Outra extends Configuracao {}      // ERRO: classe final não pode ser herdada

```

3.3. abstract

O modificador `abstract` é usado para declarar classes ou métodos que devem ser implementados por subclasses. Uma classe abstrata não pode ser instanciada diretamente e pode conter métodos abstratos (sem corpo), que obrigam as subclasses concretas a fornecerem sua própria implementação.

```
public abstract class Forma {
    public abstract double calcularArea(); // sem implementação

    public void exibirTipo() {
        System.out.println("Esta é uma forma geométrica.");
    }
}

public class Retangulo extends Forma {
    private double largura, altura;

    public Retangulo(double largura, double altura) {
        this.largura = largura;
        this.altura = altura;
    }

    @Override
    public double calcularArea() { // implementação obrigatória
        return largura * altura;
    }
}

// Forma f = new Forma();          // ERRO: não é possível instanciar classe abstrata
Forma f = new Retangulo(4, 5);    // OK: instancia a subclasse concreta
```

4. Boas Práticas

Como regra geral, recomenda-se manter os atributos como `private`, expondo apenas o necessário por meio de métodos públicos bem definidos. Isso reforça o encapsulamento, reduz o acoplamento entre classes e facilita a manutenção do código, pois mudanças internas na classe não afetam quem a utiliza, desde que a interface pública permaneça a mesma.

5. Conclusão

Os modificadores de acesso (`private`, *default*, `protected` e `public`) controlam a visibilidade dos elementos de uma classe, enquanto os modificadores de não-acesso (`static`, `final` e `abstract`) ajustam comportamentos específicos, como compartilhamento entre instâncias, imutabilidade e obrigatoriedade de implementação. Dominar o uso correto desses recursos é essencial para escrever código Java seguro, organizado e alinhado com os princípios da Programação Orientada a Objetos.