Modificadores, Enums, Herança e Polimorfismo

Esta apresentação abordará modificadores estáticos, enums, herança e polimorfismo.

Objetivo: fornecer uma compreensão clara e aplicável desses conceitos.

Ideal para estudantes de programação iniciantes.

Modificadores Estáticos

Memória Compartilhada

Atributos estáticos são associados à classe.

Todos os objetos compartilham o mesmo valor.

Acesso Direto

Acesso direto pelo nome da classe.

Não precisa criar uma instância.

Exemplo de Modificador Estático

```
class Contador {
    static int count = 0;

    Contador() {
        count++;
        System.out.println(count);
    }

    public static void main(String args[]) {
        Contador c1 = new Contador();
        Contador c2 = new Contador();
        Contador c3 = new Contador();
    }
}
```

O atributo **count** é estático. Ele armazena o número de objetos criados.

Enums (Enumerações)

1 **Conjunto Fixo**

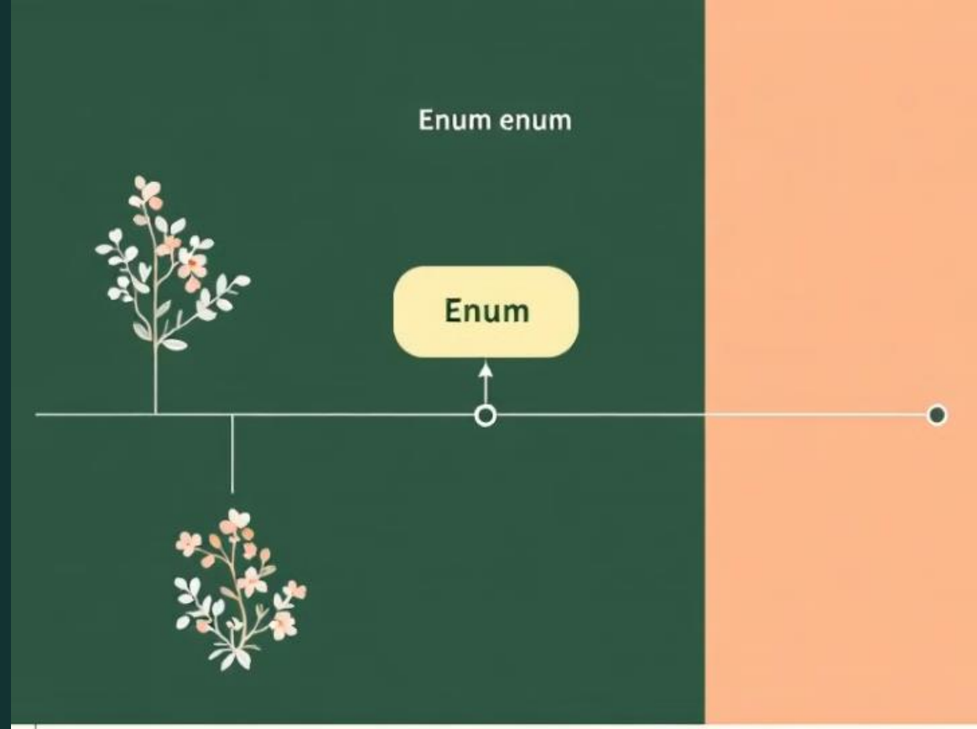
Define um conjunto fixo de constantes.

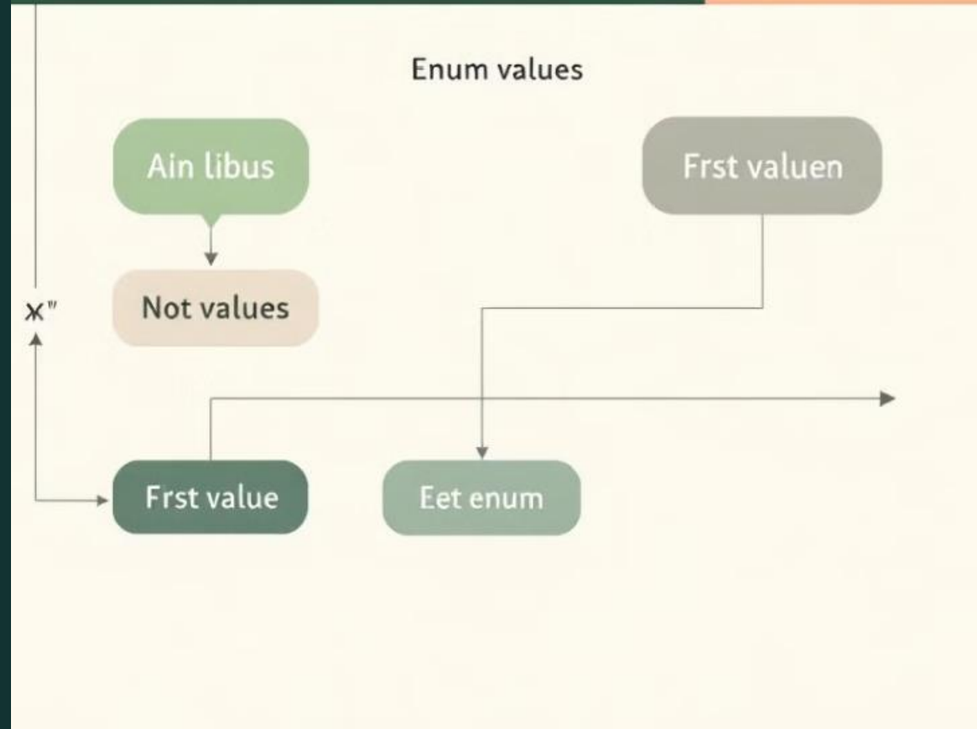
2 **Legibilidade**

Código mais legível e seguro.

3 **Substituição**

Substitui o uso de variáveis constantes.





Exemplo de Enum

```
public enum DiaSemana {
    SEGUNDA,
    TERCA,
    QUARTA,
    QUINTA,
    SEXTA,
    SABADO,
    DOMINGO
}

public class ExemploEnum {
    public static void main(String[] args) {
        DiaSemana hoje = DiaSemana.QUARTA;
        System.out.println("Hoje é: " + hoje);
    }
}
```

Criação de um enum **DiaSemana** para representar os dias da semana.

Herança

Derivação

Uma classe deriva de outra.

Extends

Palavra-chave **extends**.

Reutilização

Reutilização de código.

Exemplo de Herança

```
class Animal {
    String nome;
    void comer() {
        System.out.println("Animal
comendo");
    }
}
```

```
class Cachorro extends Animal {
    void latir() {
        System.out.println("Cachorro
latindo");
    }
}
```

A classe **Cachorro** herda de **Animal**. Reutilização do código e facilita a manutenção.

Polimorfismo

Várias Formas

Classe assume várias formas.

Sobrecarga

Métodos com mesmo nome.

Parâmetros diferentes.

Substituição

Override de métodos.

Em subclasses.

Polimorfismo - @Override

```
class Animal {
    void fazerSom() {
        System.out.println("Som genérico");
    }
}

class Gato extends Animal {
    @Override
    void fazerSom() {
        System.out.println("Miau");
    }
}
```

O método **fazerSom** é sobrescrito na classe **Gato**. A anotação @Override indica a sobrescrita.

Polimorfismo - Sobrecarga

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

    double somar(double a, double b) {
        return a + b;
    }
}
```

Sobrecarga do método **somar** com diferentes tipos de parâmetros.

Referências Bibliográficas

- Java: como programar. 10. ed. São Paulo: Pearson Education do Brasil, 2017. 934 p. ISBN: 978-85-430-0479-2.
- PUGA, Sandra; RISSETTI, Gerson. Lógica de programação e estrutura de dados com aplicações em Java. 2. ed. São Paulo: Pearson Prentice Hall, 2009. 262 p. ISBN: 978-85-7605-207-4.