POR CRISTIANE PAGINE

JAVA

INTERFACE

2025

1. Definição de Interface

Em Java, uma interface é um tipo de estrutura que define um conjunto de métodos abstratos (sem implementação) que podem ser implementados por uma classe. Interfaces são um dos pilares da Programação Orientada a Objetos (POO) e são amplamente utilizadas para garantir polimorfismo e abstração.

Características das Interfaces:

✓ Não podem conter implementação de métodos (exceto métodos default e static no Java 8+).

✓ Todos os métodos são públicos (public) e abstratos (abstract) por padrão.

✓ Uma classe pode implementar múltiplas interfaces, superando a limitação da herança única em Java.

✓ Interfaces podem conter constantes (static final), mas não variáveis de instância.

Sintaxe Básica:

```
interface NomeDaInterface {
    void metodo1();
    void metodo2();
}
```

2. Implementando Interfaces em Java

Quando uma classe implementa uma interface, ela deve fornecer a implementação de todos os métodos definidos na interface.

Exemplo de Implementação de Interface:

```
// Definição da interface
interface Animal {
    void fazerSom(); // Método sem implementação
}
// Classe Cachorro implementando a interface Animal
class Cachorro implements Animal {
    public void fazerSom() {
        System.out.println("O cachorro late: Au Au!");
    }
}
// Classe Gato implementando a interface Animal
class Gato implements Animal {
    public void fazerSom() {
        System.out.println("O gato mia: Miau!");
    }
}
// Classe de Teste
public class Teste {
    public static void main(String[] args) {
        Animal meuCachorro = new Cachorro();
        Animal meuGato = new Gato();
        meuCachorro.fazerSom();
        meuGato.fazerSom();
    }
}
// Saída no Console
// O cachorro late: Au Au!
// O gato mia: Miau!
```

Explicação

✓ Criamos a interface Animal com um método abstrato fazerSom().

✓ Cachorro e Gato implementam a interface e fornecem suas próprias versões do método.

✓ No main(), usamos polimorfismo para chamar o método fazerSom() em diferentes classes.

3. Implementando Múltiplas Interfaces

Uma classe pode implementar mais de uma interface. Isso ajuda a contornar a limitação da herança única do Java.

Exemplo com múltiplas interfaces

```
// Primeira interface
interface Animal {
    void fazerSom();
}

// Segunda interface
interface Domesticado {
    void obedecer();
}

// Classe que implementa ambas as interfaces
class Cachorro implements Animal, Domesticado {
    public void fazerSom() {
        System.out.println("O cachorro late: Au Au!");
    }

    public void obedecer() {
        System.out.println("O cachorro obedece
comandos!");
    }
}

// Classe de Teste
public class Teste {
    public static void main(String[] args) {
        Cachorro cachorro = new Cachorro();
        cachorro.fazerSom();
        cachorro.obedecer();
    }
}
```

4. Métodos default e static em Interfaces

Desde o Java 8, as interfaces permitem métodos com implementação usando default e static.

4.1. Métodos default

Os métodos default permitem que as interfaces tenham implementação padrão, sem a necessidade de todas as classes que as implementam reescrevê-los.

Exemplo de Método default:

```
// Definição da interface com método default
interface Veiculo {
    void acelerar(); // Método abstrato

    // Método default com implementação
    default void buzinar() {
        System.out.println("Buzina: Bi Bi!");
    }
}
// Implementação da interface
class Carro implements Veiculo {
    public void acelerar() {
        System.out.println("O carro está
acelerando...");
    }
}
// Classe de Teste
public class Teste {
    public static void main(String[] args) {
        Carro meuCarro = new Carro();
        meuCarro.acelerar();
        meuCarro.buzinar(); // Chamando o método
default
    }
}
```

4.2. Métodos static

Os métodos static pertencem à interface, e não às classes que a implementam. Eles podem ser chamados diretamente pela interface.

Exemplo de Método static

```
interface Utilidade {
    static void exibirMensagem() {
        System.out.println("Este é um método estático
na interface!");
    }
}

// Classe de Teste
public class Teste {
    public static void main(String[] args) {
        Utilidade.exibirMensagem(); // Chamando método
estático da interface
    }
}
```

5. Herança Múltipla com Interfaces

Como Java não suporta herança múltipla entre classes, as interfaces podem ser usadas para simular esse comportamento.

Exemplo de Herança Múltipla

```
// Interface base 1
interface Esportista {
    void treinar();
}

// Interface base 2
interface Trabalhador {
    void trabalhar();
}

// Classe implementando duas interfaces
class Pessoa implements Esportista, Trabalhador {
    public void treinar() {
        System.out.println("A pessoa está
treinando...");
    }

    public void trabalhar() {
        System.out.println("A pessoa está
trabalhando...");
    }
}

// Classe de Teste
public class Teste {
    public static void main(String[] args) {
        Pessoa pessoa = new Pessoa();
        pessoa.treinar();
        pessoa.trabalhar();
    }
}
```

6. Resumo sobre Interfaces em Java

As interfaces são fundamentais na programação em Java, pois permitem definir contratos que as classes devem seguir, promovendo a reutilização de código e a flexibilidade do design.

Principais pontos abordados:

✓ Definição: Interfaces são tipos abstratos que contêm apenas a assinatura dos métodos, sem implementação.

✓ Implementação: Classes devem implementar todos os métodos de uma interface.

✓ Múltiplas interfaces: Uma classe pode implementar mais de uma interface, permitindo simular herança múltipla.

✓ Métodos default e static: Desde o Java 8, interfaces podem conter métodos com implementação, facilitando a manutenção do código.

✓ Herança em interfaces: Interfaces podem estender outras interfaces, organizando melhor os contratos de código.

As interfaces são amplamente utilizadas no desenvolvimento Java, especialmente em APIs, frameworks e aplicações baseadas em padrões como SOLID e POO.