UniSENAI

Universidade Campus Joinville - SC

Análise e Desenvolvimento de Sistemas

Interface em POO

Profº Diego Madureira e Silvio Souza

Aluno: Nivea Souza

Joinville - SC
2025

Interface em POO

Uma interface, em Programação Orientada a Objetos, define um conjunto de métodos e propriedades que uma classe deve obrigatoriamente implementar. Ela especifica o que deve existir, mas não determina como cada método será executado, pois não contém implementação interna. Quando usamos a palavra-chave *interface*, criamos um contrato. Todas as classes que implementam essa interface devem declarar todos os métodos definidos nela. Se uma classe deixa de implementar algum desses métodos, o sistema gera um erro fatal, porque o contrato não foi cumprido. Como a interface não define conteúdos internos, todos os métodos declarados nela ficam sem corpo. Desse modo, todos esses métodos são obrigatoriamente públicos, porque uma interface representa um conjunto que devem estar acessíveis.

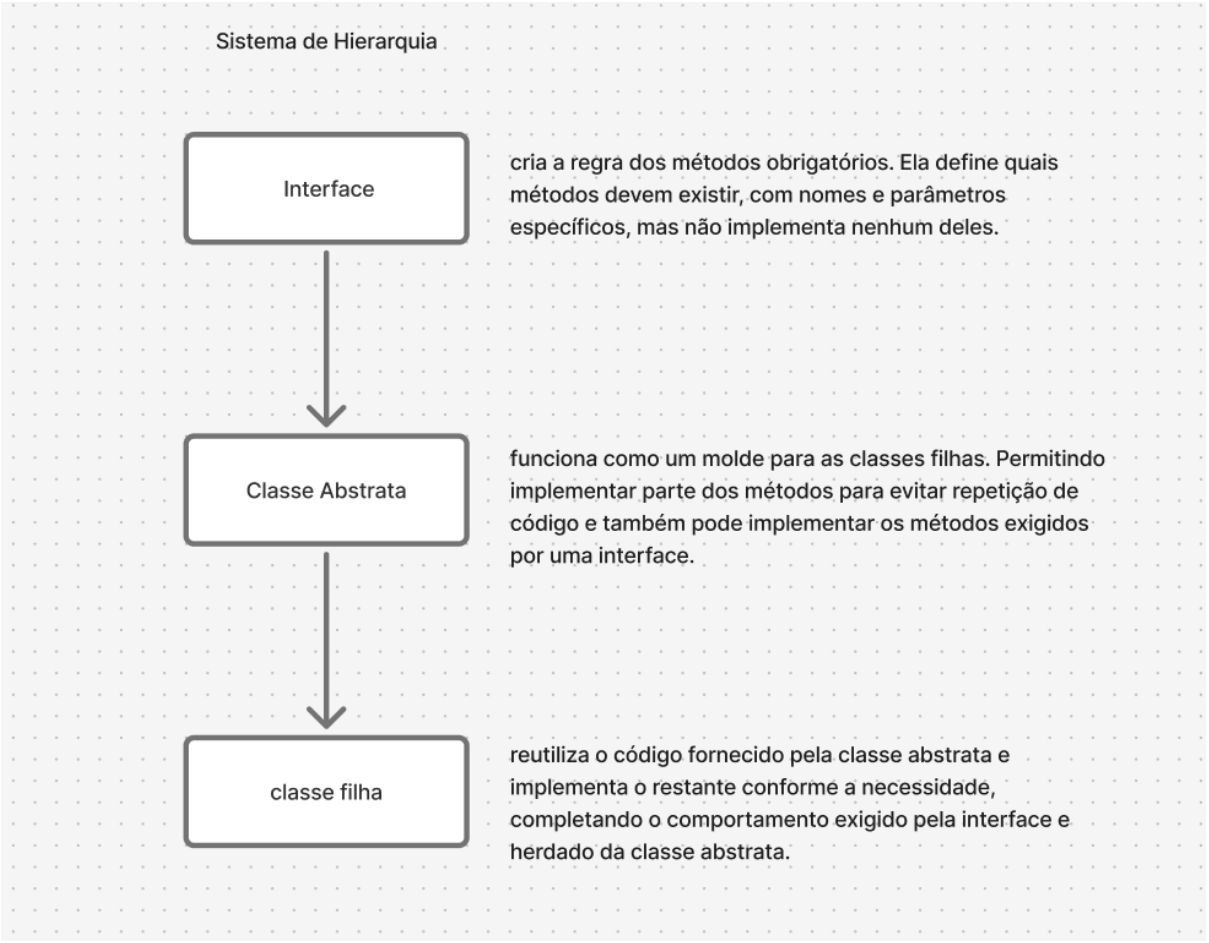

Figura 1 – Representação abstrata de uma imaginária hierarquia entre interface, classe abstrata e classes filhas. Nivea (2025)

Classe Abstrata vs Interface

Uma classe abstrata funciona como um molde para outras classes. Ela não pode ser instanciada, portanto não cria objetos diretamente. Ela serve como base e evita duplicações de código, permitindo que métodos e propriedades comuns sejam compartilhados pelas classes filhas.

```php
abstract class Animal {
    public $nome;

    public function set($nome) {
        $this->nome = $nome;
    }
    public function dormir() {
        echo $this->nome . " está dormindo.<br>";
    }

    // Método abstrato: cada animal fará isso de um jeito diferente
    abstract public function fazerSom();
}
```

Figura 2 – Classe Abstrata sendo criada. Nivea (2025)

```php
class Cachorro extends Animal {
    public function fazerSom() {
        echo $this->nome . " está latindo: Au au!<br>";
    }
}
class Gato extends Animal {
    public function fazerSom() {
        echo $this->nome . " está miando: Miau!<br>";
    }
}
class Passaro extends Animal {
    public function fazerSom() {
        echo $this->nome . " está cantando: Piui piui!<br>";
    }
}
```

Figura 3 – Classes filhas recebendo essa herança da classe abstrata sendo criada. Nivea (2025)

A diferença central entre uma classe abstrata e uma interface ocorre porque a classe abstrata pode fornecer parte da implementação, enquanto a interface apenas define um contrato. Uma classe abstrata pode implementar uma interface parcialmente, e as classes que a estendem completam o restante das implementações exigidas.

```php
interface Pagamento {
    public function pagar($valor);
}
```

```php
abstract class FormaPagamento implements Pagamento {

    public function validarValor($valor) {
        echo "Validando o valor: R$ $valor<br>";
    }

    // A classe abstrata ainda não implementa pagar(),
    // então as filhas terão que implementar.
}
```

```php
class CartaoCredito extends FormaPagamento {

    public function pagar($valor) {
        $this->validarValor($valor);
        echo "Pagamento de R$ $valor realizado no cartão.<br>";
    }
}
```

Figura 4 – Class abstrata recebendo a interface e as classes filhas tendo que afirmar a interface.
Nivea (2025)

A interface determina que certos métodos existirão com nomes e parâmetros específicos, mas deixa para a classe que a implementa a responsabilidade de definir o comportamento desses métodos. A classe abstrata, por sua vez, pode declarar métodos obrigatórios e também implementar outros métodos complementares.

Importância da Interface na manutenção do código

O uso de interfaces melhora a manutenção do código porque estabelece contratos claros dentro do sistema. Quando definimos uma interface, determinamos exatamente quais métodos uma classe deve possuir, com seus nomes e parâmetros específicos. Isso garante que diferentes classes possam oferecer comportamentos distintos, mas sempre seguindo a mesma estrutura, reduzindo o acoplamento do código, porque passo a depender do contrato e não da implementação concreta. Isso me permite trocar uma classe por outra sem alterar o código que utiliza esse contrato. Por exemplo, se eu tiver um método que recebe uma interface como parâmetro, eu posso substituir facilmente uma classe antiga por uma nova implementação sem quebrar o sistema. As interfaces também facilitam a organização do projeto, porque deixam claro o papel de cada componente. Desse modo, elas permitem que eu adicione novas funcionalidades apenas criando novas classes que implementam o mesmo contrato, sem modificar o código existente. Por esses motivos, o uso de interfaces torna o código mais flexível, mais organizado, mais fácil de testar e muito mais preparado para crescer sem se tornar confuso.

Referências:

PHP NET. Object interfaces. Disponível em www.php.net. 2025. Link: https://www.php.net/manual/en/language.oop5.interfaces.php . Acesso em: 18/11/2025.