**Definição conceitual de interfaces**

Em programação orientada a objetos, uma interface é uma estrutura que define um conjunto de métodos e constantes. Uma interface não pode ser instanciada, mas pode ser implementada por classes.

As interfaces são usadas para especificar o comportamento de uma classe. Elas podem ser usadas para:

* Fornecer uma definição comum para classes que possuem comportamentos semelhantes.
* Facilitar a reutilização do código.
* Tornar o código mais flexível e adaptável.

**Exemplo simples de interface**

Considere a seguinte interface:

Java

public interface Animal {  
 void emitirSom();  
 void seMover();  
}

Esta interface define dois métodos: emitirSom() e seMover(). Qualquer classe que implemente esta interface deve fornecer uma implementação para esses métodos.

Por exemplo, a seguinte classe implementa a interface Animal:

Java

public class Cachorro implements Animal {  
  
 @Override  
 public void emitirSom() {  
 System.out.println("Au au!");  
 }  
  
 @Override  
 public void seMover() {  
 System.out.println("O cachorro está correndo.");  
 }  
}

A classe Cachorro implementa os métodos emitirSom() e seMover() da interface Animal. Portanto, um objeto da classe Cachorro pode ser usado em qualquer lugar onde seja esperado um objeto da interface Animal.

**Implementação programática de interfaces em Java**

Para implementar uma interface em Java, use a palavra-chave implements na declaração da classe. Por exemplo:

Java

public class Cachorro implements Animal {  
  
 @Override  
 public void emitirSom() {  
 System.out.println("Au au!");  
 }  
  
 @Override  
 public void seMover() {  
 System.out.println("O cachorro está correndo.");  
 }  
}

A palavra-chave implements indica que a classe Cachorro está implementando a interface Animal.

Os métodos da interface devem ser implementados na classe que a implementa. Se um método da interface não for implementado, a classe será considerada abstrata.

**Exemplo de implementação de métodos de interface**

Considere a seguinte interface:

Java

public interface Calculadora {  
  
 int somar(int a, int b);  
 int subtrair(int a, int b);  
 int multiplicar(int a, int b);  
 int dividir(int a, int b);  
}

Esta interface define quatro métodos: somar(), subtrair(), multiplicar() e dividir().

A seguinte classe implementa a interface Calculadora:

Java

public class CalculadoraSimples implements Calculadora {  
  
 @Override  
 public int somar(int a, int b) {  
 return a + b;  
 }  
  
 @Override  
 public int subtrair(int a, int b) {  
 return a - b;  
 }  
  
 @Override  
 public int multiplicar(int a, int b) {  
 return a \* b;  
 }  
  
 @Override  
 public int dividir(int a, int b) {  
 return a / b;  
 }  
}

Esta classe implementa todos os métodos da interface Calculadora. Portanto, um objeto da classe CalculadoraSimples pode ser usado em qualquer lugar onde seja esperado um objeto da interface Calculadora.

**Outras características de interfaces**

Além de métodos, as interfaces também podem conter constantes. As constantes de uma interface são automaticamente públicas e estáticas.

As interfaces podem ser herdadas por outras interfaces. Isso permite que uma interface combine os comportamentos de várias interfaces.

As interfaces também podem ser implementadas por classes anônimas. As classes anônimas são classes que não têm nome. Elas são usadas para fornecer uma implementação temporária para uma interface.

**Conclusão**

As interfaces são uma ferramenta poderosa que pode ser usada para melhorar a qualidade do código em Java. Elas podem ser usadas para:

* Fornecer uma definição comum para classes que possuem comportamentos semelhantes.
* Facilitar a reutilização do código.
* Tornar o código mais flexível e adaptável.

Entender como implementar interfaces em Java é essencial para qualquer programador que trabalhe com a linguagem.

**Fontes**

1. <https://github.com/brunobarretofreitas/RedesSD2016>

2. <https://github.com/AdrielGraminho/CursosSplenda>

**Exemplo de interface para um sistema contendo classe cliente, fornecedor e login**

Java

public interface Login {  
  
 void autenticar(String login, String senha);  
  
}

Esta interface define um método chamado autenticar(). Este método recebe dois parâmetros: o login e a senha do usuário. O método deve retornar true se o login e a senha forem válidos, ou false caso contrário.

As classes Cliente e Fornecedor implementam a interface Login. A implementação do método autenticar() nessas classes pode variar, dependendo do sistema específico.

**Implementação da classe Cliente**

Java

public class Cliente implements Login {  
  
 private String login;  
 private String senha;  
  
 public Cliente(String login, String senha) {  
 this.login = login;  
 this.senha = senha;  
 }  
  
 @Override  
 public void autenticar(String login, String senha) {  
 if (this.login.equals(login) && this.senha.equals(senha)) {  
 System.out.println("Login efetuado com sucesso!");  
 } else {  
 System.out.println("Login ou senha inválidos.");  
 }  
 }  
  
}

Esta implementação do método autenticar() verifica se o login e a senha recebidos são iguais aos login e senha do cliente. Se forem, o método imprime uma mensagem de sucesso. Caso contrário, o método imprime uma mensagem de erro.

**Implementação da classe Fornecedor**

Java

public class Fornecedor implements Login {  
  
 private String login;  
 private String senha;  
  
 public Fornecedor(String login, String senha) {  
 this.login = login;  
 this.senha = senha;  
 }  
  
 @Override  
 public void autenticar(String login, String senha) {  
 if (this.login.equals(login) && this.senha.equals(senha)) {  
 System.out.println("Login efetuado com sucesso!");  
 } else {  
 System.out.println("Login ou senha inválidos.");  
 }  
 }  
  
}

Esta implementação do método autenticar() é semelhante à implementação da classe Cliente. A única diferença é que ela verifica se o login e a senha recebidos são iguais aos login e senha do fornecedor.

**Exemplo de uso**

Java

public class Main {  
  
 public static void main(String[] args) {  
 Cliente cliente = new Cliente("cliente1", "senha1");  
 Fornecedor fornecedor = new Fornecedor("fornecedor1", "senha1");  
  
 cliente.autenticar("cliente1", "senha1");  
 fornecedor.autenticar("fornecedor1", "senha1");  
 }  
  
}

Este código imprime as seguintes mensagens:

Login efetuado com sucesso!  
Login efetuado com sucesso!

**Conclusão**

Este exemplo ilustra como uma interface pode ser usada para especificar o comportamento de classes que possuem comportamentos semelhantes. No caso deste exemplo, a interface Login especifica que as classes Cliente e Fornecedor devem implementar um método chamado autenticar(). A implementação deste método pode variar de acordo com o sistema específico.

**Fontes**

1. <https://github.com/Ryuukuh/PARADIGMAS-B>