INTERFACES

Interfaces

"Existem diversas situações na Engenharia de Software em que é importante que grupos muito diferentes de programadores entrem em acordo quanto aos termos de um "contrato" que descreva como seus softwares interagem.

Cada grupo deve ser capaz de escrever seu código sem qualquer conhecimento de como o código do outro grupo é escrito. De um modo geral, esse contrato são as **Interfaces**".

Disponível em: SCHILDT, Herbert. **Java: a referência completa.** 1 ed. Porto Alegre: Bookman, 2014.

Interfaces

- Recurso que define ações que devem ser obrigatoriamente executadas, mas que cada classe pode executar de forma diferente.

- Uma interface não é herdada, mas sim, implementada. Elas contém valores constantes e/ou assinaturas de métodos que devem ser implementados numa classe.

- Isso se deve ao fato que muitas classes podem possuir a mesma ação (método), mas executá-la de maneira diferente.

Interfaces

- Este conceito de Interface não tem nada a ver com interfaces gráficas (parte visual), mas sim, com comportamentos que uma classe pode ter.

- As interfaces são padrões definidos através de especificações, como um “contrato” que define um determinado conjunto de métodos que deverão ser implementados nas classes que “assinam” o contrato.

- Na Orientação a Objetos, as vezes é útil definir o que uma classe deve fazer, mas não como ela o fará.

Interfaces

- Uma interface é sintaticamente semelhante a uma classe abstrata no fato de poder implementar um ou mais métodos sem “corpo”.

- Para que suas ações sejam definidas, esses métodos devem receber um “corpo” em uma classe. Assim, as interfaces não farão suposições sobre como elas são implementadas.

- Não há limites para o número de classes que podem implementar uma interface e não há limites para o número de interfaces que uma classe pode implementar.

Interfaces

- Os métodos são declarados com o uso apenas de seu tipo de retorno e assinatura; assim, sua ocorrência é apenas prevista, mas não programada.

- Para implementar uma interface, a classe deve fornecer obrigatoriamente implementações para os todos métodos descritos nela que deverão ser públicos. Cada classe é livre para determinar os detalhes da interpretação.

- As variáveis declaradas em uma interface não são variáveis de instância, sendo implicitamente public, final e static, devendo ser inicializadas. Ou seja: são basicamente constantes.

Uma interface é definida através da palavra-chave **interface**.

```
public interface Animal {
   public void emitirSom();
   public void dormir();
}
```

O corpo do método da interface é fornecido pela classe o que implementa.

Para implementar uma interface, inclua a cláusula **implements** em uma definição de classe e então crie os métodos requiridos pela interface.

```
public class Cachorro implements Animal {

    public void emitirSom() {
        System.out.println(“Au... Au...");
    }

    public void dormir() {
        System.out.println("Zzz");
    }
}
```

```
public class Gato implements Animal {

    public void emitirSom() {
        System.out.println(“Miau...Miau");
    }

    public void dormir() {
        System.out.println(“Ron! ");
    }
}
```

Adaptado de: https://www.w3schools.com/java/java_interface.asp

```
public interface Conta{
    void depositar(double valor);
    void sacar(double valor);
    double getSaldo();
}
```

A interface Conta tem seus métodos sem corpo, apenas com os parâmetros e o tipo de retorno. os métodos são sobrepostos pelas classes que implementam a interface “Conta”.

```
public class ContaCorrente implements Conta {
    private double saldo;
    private double taxaOperacao = 0.45;

    @Override
    public void deposita(double valor) {
        this.saldo += valor - taxaOperacao;
    }

    @Override
    public double getSaldo() {
        return this.saldo;
    }

    @Override
    public void sacar(double valor) {
        this.saldo -= valor + taxaOperacao;
    }
}
```

```
public class ContaPoupanca implements Conta {
    private double saldo;

    @Override
    public void deposita(double valor) {
        this.saldo += valor;
    }

    @Override
    public double getSaldo() {
        return this.saldo;
    }

    @Override
    public void sacar(double valor) {
        this.saldo -= valor;

    }
}
```

Fonte: https://www.devmedia.com.br/polimorfismo-classes-abstratas-e-interfaces-fundamentos-da-poo-em-java/26387

```
public interface Cidadao {
  public void votar();
  public void casar();
  public void estudar();
}

public interface Contribuinte {
   public void declararIR();
   public void receberRestituicao();
}

public interface Pai {
   public void educar();
}
```

```
public class Brasileiro implements Cidadao, Contribuinte, Pai {

    @Override
    public void votar(){
        // Código
    }

    @Override
    public void casar() {
        // Código
    }

    @Override
    public void estudar() {
        // Código
    }

    @Override
    public void declararIR() {
        // Código
    }

    @Override
    public void receberRestituicao() {
        // Código
    }

    @Override
    public void educar() {
        // Código
    }
}
```

Para implementar várias interfaces, separe-as com uma vírgula

Interfaces

- Como as classes abstratas, as interfaces não podem ser usadas para criar objetos.
- É permitido e comum que classes que implementam interfaces podem definir métodos adicionais.
- Uma interface não pode conter um construtor (pois não pode ser usada para criar objetos)
- A partir do Java 8 podemos definir métodos dentro de interfaces fornecendo uma implementação default.

Por que usar Interfaces?

- **Segurança:** Oculte certos detalhes e mostre apenas o que é importante de um objeto (interface).

- **Java não suporta "herança múltipla“:** Uma classe só pode herdar de uma superclasse). No entanto, isso pode ser alcançado com interfaces, já que uma classe pode implementar várias interfaces.

- Quando a evolução for mais importante que a flexibilidade das interfaces, deve-se utilizar classes abstratas.

Métodos Default - Java 8

- Na versão 8 do Java é possível definir implementações em métodos de interface, chamados de método default.
- A maior motivação para a criação de métodos default foi a necessidade de adicionar novas funcionalidades às interfaces existentes sem quebrar o código que a utilizam
- Nas versões anteriores do Java, ao adicionar um novo método numa interface, somos obrigados a alterar todas as classes que herdam dessa interface.
- O recurso permite que interfaces possam evoluir sem risco de comprometer o legado (versões antigas).

Referências Bibliográficas

PALMEIRA, Thiago Vinícius Varallo. Polimorfismo, Classes abstratas e Interfaces: Fundamentos da POO em Java. **Dev Media.** Disponível em: <https://www.devmedia.com.br/polimorfismo-classes-abstratas-e-interfaces-fundamentos-da-poo-em-java/26387>. Acesso em 29 de Junho de 2020.

SENAGA, Marcelo. Métodos Default no Java. **Dev Media.** Disponível em: <https://www.devmedia.com.br/metodos-default-no-java/33012>. Acesso em 29 de Junho de 2020.

INTERFACE Java. **W3Schools**. Disponível em<https://www.w3schools.com/java/java_interface.asp>. Acesso em 29 de Junho de 2020.

JUNIOR. Antonio B. C. Sampaio. Curso de Java Completo Para Iniciantes. **Udemy.** 2019. Disponível em: <https://www.udemy.com/course/java-8-fundamentos-teoricos-e-orientacao-a-objetos/learn/lecture/4366560#overview>. Acesso em 29 de Junho de 2020.

Referências Bibliográficas

GUANABARA. Gustavo. Curso POO Teoria #06a - Pilares da POO: Encapsulamento. **Youtube.** 2016. Disponível em: <https://www.youtube.com/watch?v=1wYRGFXpVlg>. Acesso em 29 de Junho de 2020.

JUNIOR, Peter Jandl. **Java: guia do programador.** 3 ed. São Paulo: Novatec, 2015.

SCHILDT, Hebert. **Java para iniciantes: crie, compile e execute programas Java rapidamente.** 6 ed. Porto Alegre: Bookman, 2015.

GALLARDO, R.; KANNAN, S.; ZACKHOUR, S. B. **Tutorial Java.** 5 ed. Rio de Janeiro: Alta Books, 2015.

SCHILDT, Herbert. **Java: a referência completa.** 1 ed. Porto Alegre: Bookman, 2014.

Material desenvolvido pelo
Prof. Rafael da Silva Polato
rafael.polato@etec.sp.gov.br

Divisão de Turma
Prof. Leandro Bordignon
leandro.bordignon01@etec.sp.gov.br