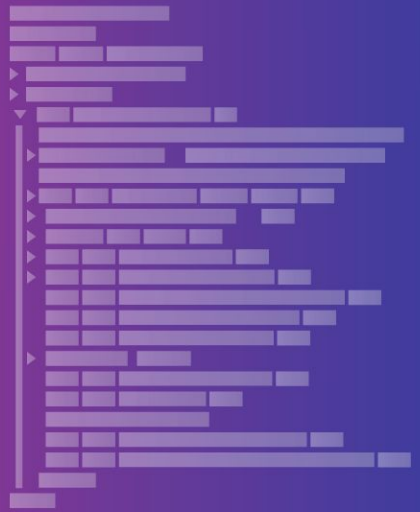
Java: sobrescrita e extends



DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA

- Sobrescrita
- Extends vs. Implements
- Interface

Sobrescrita

Assim como a sobrecarga (que vocês já estudaram) a **sobrescrita** é presente em Java desde seu início. Na maioria das linguagens de programação, métodos com o mesmo nome tendem a gerar um erro, o que não é o caso em Java, por conta dos dois métodos mencionados.

A **sobrescrita** em resumo é a criação, na subclasse, de um método com a mesma assinatura, ou seja, mesmo nome, mesma quantidade de parâmetros, e o mesmo tipo de parâmetros do método sobrescrito da superclasse.

Tá, mas qual é a diferença entre ambos?

A sobrecarga tem assinatura diferente apesar do mesmo nome, a sobrescrita tem a mesma assinatura.

Extends vs. Implements

As palavras-chave *extends* e *implements* realizam operações parecidas em Java: elas permitem que uma classe herde as características (atributos e métodos) de outra classe.

Extends permite que uma classe de fato estenda a funcionalidade de uma classe “mãe”. Contudo, apenas uma classe pode ser estendida. Isso serve para evitar ambiguidade.

A implementação, executada a partir da palavra-chave "implements", permite a implantação de uma *interface* sem limites de quantidade.

```
class One {
    public void methodOne()
    {

        // Some Functionality
    }
}

class Two extends One {

    public static void main(String args[])
    {
        Two t = new Two();

        // Calls the method one
        // of the above class
        t.methodOne();
    }
}
```

Interface

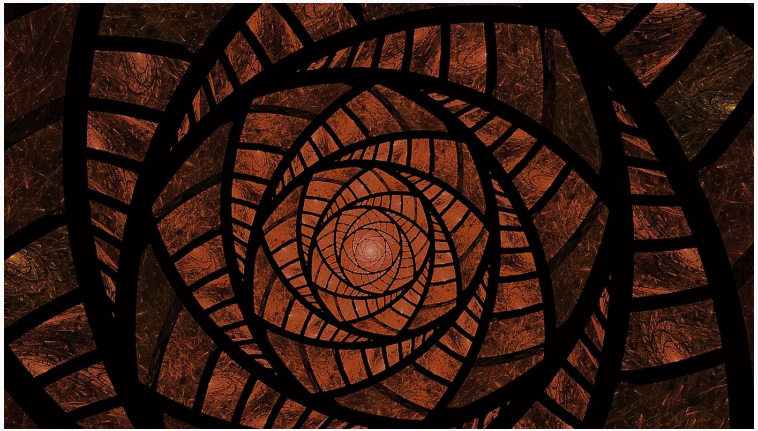
As interfaces são um tipo “especial” de classe que permite a criação de abstrações.
Para acessarmos os métodos da interface, que também são abstratos, a interface precisa ser implementada por outra classe

```
// Defining an interface
interface One {
    public void methodOne();
}

// Defining the second interface
interface Two {
    public void methodTwo();
}

// Implementing the two interfaces
class Three implements One, Two {
    public void methodOne()
    {

        // Implementation of the method
    }

    public void methodTwo()
    {

        // Implementation of the method
    }
}
```

Interface

Para que serve a interface?

- Abstração Total
- Herança múltipla
- Loose-coupling (acoplamento solto) - construção de sistemas com associações fracas entre componentes - evitando assim a afetação de componentes a partir da alteração



Material Complementar

- https://howtodoinjava.com/java/oops/extends-vs-implements/
- https://www.digitalocean.com/community/tutorials/interface-in-java

AVALIAÇÃO DOCENTE

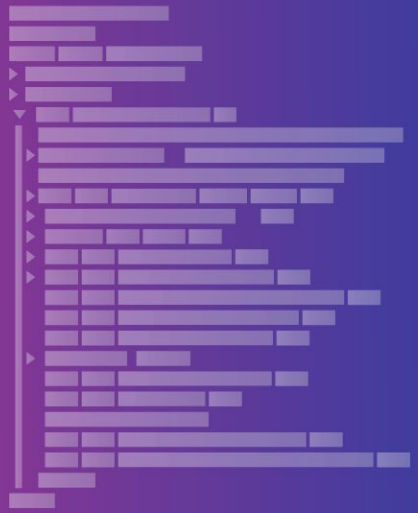
O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

DEVinHouse

Parcerias para desenvolver a sua carreira



OBRIGADO!

SENAI

<LAB365>