# Sobrescrevendo equals(Object o)

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

# Temas

1 equals(Object o)

2 Objeto pode ser qualquer coisa

3 Casting

# 1 | equals(Object o)

# .equals(Object o)

Quando criamos um objeto ou instância, o que temos é uma **referência à memória** (RAM), ou seja, os dados não são armazenados diretamente na variável de tipo de objeto, apenas a referência ao local onde estão os valores dos atributos do objeto.

É por isso que não podemos usar o operador **"=="** para comparar a igualdade entre dois objetos porque estaríamos comparando referências.

# .equals(Object o)

Para comparar corretamente os objetos, devemos usar o **método equals()**, que herdamos de **Object**, mas o equals que obtemos por padrão nem sempre funciona corretamente, portanto, é necessário sobrescrevê-lo.

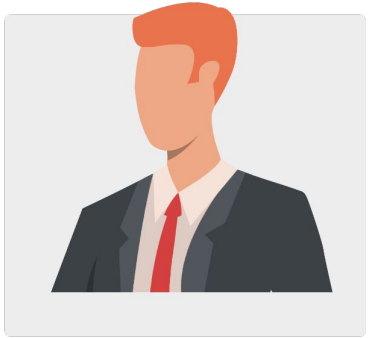
O método equals() recebe um objeto **Object** como parâmetro, o que nos dará dificuldade adicional quando se trata de sobrescrevê-lo.

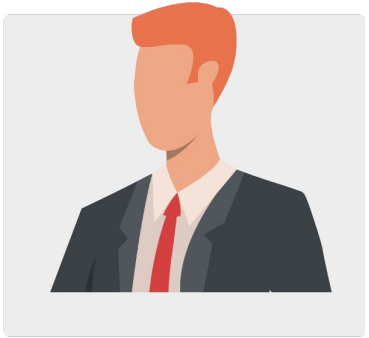
Lembre-se de que não podemos alterar a assinatura do método se quisermos sobrescrevê-lo.

# O que significa que dois objetos são iguais?

Antes de começar a trabalhar com o método equals, devemos pensar no que significa que dois objetos são iguais, por exemplo, se compararmos dois funcionários, poderíamos definir que eles são iguais se seus arquivos forem iguais.

Dessa forma, quando escrevemos uma classe, uma das coisas que devemos determinar é como vamos verificar a igualdade de duas instâncias dessa classe.





# 2 | Objeto pode ser qualquer coisa

# Classe Empregado

Como exemplo, vamos trabalhar com a classe de empregados e, como mencionamos, dois funcionários são iguais se seus arquivos forem iguais.

```
public class Empregado{
      private   String nome;
      private   String arquivo;
      protected double salario;
      protected double descontos;


}
```

# .equals(Object o)

Vamos sobrescrever o método .equals (Object o). Nosso primeiro passo é lembrar como é a assinatura deste método, devemos respeitar a assinatura do método herdado de Object.

```
public class Empregado{
      private   String nome;
      private   String arquivo;
      protected double salario;
      protected double descontos;

      @Override
      public boolean equals(Object o){
      }
}
```

# .equals(Object o)

Para começar a escrever nosso método equals, devemos considerar que o parâmetro que está pedindo é um Objeto, não diz que é um Empregado. Logo, a primeira coisa a verificar é se realmente é um Empregado, caso contrário, já podemos dizer que eles não são iguais.

Vamos ver duas maneiras de verificar isso:

- **instanceof**
- **getClass()**

```
@Override
public boolean equals(Object o){

}
```

# instanceof

O **instanceof** é uma forma de comparar duas instâncias.

```
@Override
public boolean equals(Object o){
      if (o==null)
            return false;
      if (!(o instanceof Empregado))
            return false;
      else{

      }
}
```

# instanceof

```
@Override
public boolean equals(Object o)
      if (o==null)
            return false;
      if (!(o instanceof Empregado))
            return false;
      else{

      }
}
```

Por ser um objeto, ele pode ter um valor nulo, é a primeira coisa que nós verificamos.

# instanceof

```
@Override
public boolean equals(Object o)
     if (o  null)
          return false;
     if (!(o instanceof Empregado))
          return false;
     else{

     }
}
```

instanceof é um operador que nos permite verificar se é ou não uma instância de Empregado. Nesse caso, se não for, retornamos falso, os objetos não podem ser os mesmos.

# instanceof

```
@Override
public boolean equals(Object o)
     if (o  null)
          return false;
     if (!(o instanceof Empleado))
          return false;
     else{
     }
}
```

Se for uma instância de Empregado, siga a verificação.

# .getClass()

Com getClass(), também podemos comparar a classe à qual os objetos pertencem.

```
@Override
public boolean equals(Object o){
      if (o==null)
            return false;
      if (this.getClass()==o.getClass())
            return false;
      else{

      }
}
```

# .getClass()

```
@Override
public boolean equals(Object o)
     if (o==null)
          return false;
     if (this.getClass()==o.getClass())
          return false;
     else{

     }
}
```

Verificamos se a classe da instância é a mesma classe do objeto recebido como parâmetro.

# 3 | Casting

# Casting

Agora teríamos que verificar a igualdade (ter o mesmo arquivo). Para fazer essa verificação, precisaremos pedir ao objeto "o", o arquivo para compará-lo com a instância. Mas "o" é um objeto, isto é, não "sabe" o que tem.

```
@Override
public boolean equals(Object o){
      if (o==null)
            return false;
      if (!(o instanceof Empregado))
            return false;
      else{
      }
}
```

# Casting

Assim, o método estaria concluído, mas por conveniência usamos um molde. Embora não seja necessário criar um novo objeto, é mais confortável para uma leitura posterior.

```
@Override
public boolean equals(Object o){
      if (o==null)
            return false;
      if (!(o instanceof Empleado))
            return false;
      else{
            Empregado empregadoAux=(Empregado)o;
            return
this.getArquivo().equals(empregadoAux.getArquivo())
;
            }
      }
}
```

# .getClass()

```
@Override
public boolean equals(Object o)
     if (o==null)
          return false;
     if (this.getClass()==o.getClass())
          return false;
     else{
          Empregado empregadoAux=(Empregado)o;

          if(this.getArquivo().equals(empregadoAux.getArquivo()));
               return true;
          }
     return false;
     }
}
```

Projetamos "o" e o atribuímos a um objeto do tipo Empregado. Com o casting, o que conseguimos é transformar, a fim de atribuí-lo a um objeto do tipo Empregado, e assim, usar seus métodos.

# .getClass()

```
@Override
public boolean equals(Object o)
      if (o==null)
            return false;
      if (this.getClass()==o.getClass())
            return false;
      else{
            Empregado empregadoAux=(Empregado)o;
            if (this.getArquivo().equals(empregadoAux.getArquivo()));
                  return true;
            }
      return false;
      }
}
```

Verificamos se tem o mesmo arquivo, utilizando o equals. Arquivo é um atributo do tipo String, portanto, não podemos usar o operador “==”.


DigitalHouse>
Coding School