

Modificadores e Palavras Reservadas

Orientação a Objetos - DCC025

Prof. Edmar Welington Oliveira
edmar.oliveira@ufjf.edu.br

Universidade Federal de Juiz de Fora - UFJF
Departamento de Ciência da Computação - DCC

Modificadores

<i>Modificador</i>	<i>Descrição</i>
default	Somente classes do mesmo package possuem acesso
public	Todos possuem acesso
protected	Apenas os membros da classe e subclasse
private	Apenas os membros da classe

<i>Modificador</i>	<i>Descrição</i>
static	A variável ou método é comum a todas as instâncias da classe.
final	O valor da variável não pode ser modificado.

Modificadores - Static

- Static

- Um atributo declarado com o modificador static significa que é um **atributo da classe** e **não de instância**. Em outras palavras, existirá apenas um atributo, ocupando uma única posição de memória, em vez de um atributo para cada instância, cada um com sua própria posição de memória.
 - Como resultado, o atributo existirá antes mesmo que qualquer objeto seja criado

Modificadores

- Static
 - Outro efeito é que o atributo será compartilhado por todas as instâncias da classe. Desta forma, **se um objeto alterar o conteúdo do atributo, a alteração afetará todos os outros objetos da classe.**

Modificadores

Classe Pessoa

```
1 public class Pessoa {  
2  
3     private static String nome;  
4  
5     public String getNome() {  
6         return nome;  
7     }  
8  
9     public void setNome(String nome) {  
10         Pessoa.nome = nome;  
11     }  
12 }
```

Alteração no atributo nome do Objeto P1

```
13 public static void main(String args[]){  
14  
15     Pessoa p1 = new Pessoa();  
16     Pessoa p2 = new Pessoa();  
17  
18     System.out.println(p1.getNome());  
19     System.out.println(p2.getNome());  
20  
21     p1.setNome("Edmar");  
22  
23     System.out.println(p2.getNome());  
24 }  
25 }
```

Capturar o valor do atributo do objeto P2

Modificadores

■ Static

- O modificador **static** também pode ser usado na declaração de métodos. Neste caso o método fica ligado à classe e não à instância, sendo denominado de **método de classe**. Como consequência o método pode ser acessado antes de existir um objeto da classe.
 - Exemplo de método static: o método main()

Exemplo - Static

```
3 public class Livro {  
4  
5     private String nome;  
6     private double preco;  
7     private double desconto;  
8  
9     public double getDesconto() {  
10         return desconto;  
11     }  
12  
13     public void setDesconto(double desconto) {  
14         this.desconto = desconto;  
15     }  
16 }
```

Exemplo - Static

```
3 public class Livraria {  
4  
5     ...  
6  
7     public double calculaPrecoFinal(Livro book1) {  
8  
9         double valor = book1.getPreco() * (1.0 - book1.getDesconto() );  
10        return valor;  
11    }  
12  
13    ...  
14  
15 }
```

Se o livreiro tiver a política de descontos únicos para toda a loja, pode ser que o atributo desconto tenha alguma inconsistência, algum valor diferente de um livro para o outro o que seria indesejável.

Como garantir que todos os livros tenham o mesmo desconto ?

Exemplo - Static

```
3 public class Livro {  
4  
5     private String nome;  
6     private double preco;  
7     private static double desconto;  
8  
9     public static double getDesconto() {  
10         return desconto;  
11     }  
12  
13     public static void setDesconto(double d) {  
14         desconto = d;  
15     }  
16 }
```

Exemplo - Static

```
3 public class Livraria {  
4  
5     ...  
6  
7     public double calculaPrecoFinal(Livro book1) {  
8  
9         double valor = book1.getPreco() * (1.0 - Livro.getDesconto());  
10        return valor;  
11    }  
12  
13    ...  
14  
15 }
```

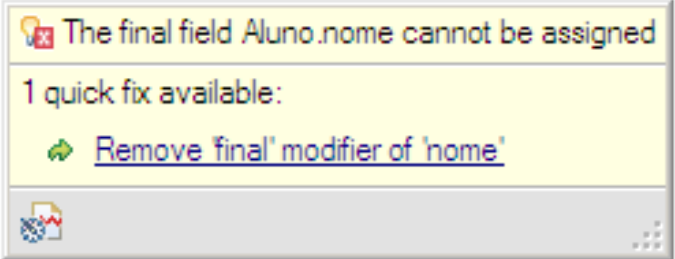
Modificadores - Final

■ Final

- O modificador **final** pode ser usado em atributos, métodos e, também, classes. Impede se modifique o que está sendo prefixado.
- Final aplicado sobre:
 - Atributos: torná-los constantes
 - Métodos: impedir sobrescrita de métodos
 - Classes: impedir criar classes filhas (impedir herança)
- OBS: Sobrescrita (sobrescrever): Métodos de assinaturas iguais (mesmo método em termos de assinatura, mas podem possuir implementações diferentes

Modificadores - Final

```
3 public class Aluno {  
4  
5     private final String nome = "NomeTeste";  
6  
7     public void alteraNome(String valor){  
8         nome = valor;  
9     }  
10  
11 }  
12  
13  
14
```



Como o atributo "nome" está definido como FINAL, é impossível alterar seu valor. Qualquer tentativa neste sentido gera um erro de compilação

Palavra Reservada “this”

- Referência “this”
 - Em algumas situações é necessário referenciar o próprio objeto corrente. Por essa razão, todo objeto possui um atributo especial identificado por **this**, que é uma referência para o próprio objeto.
 - O **this** é um apelido para o endereço de memória do objeto

Palavra Reservada "this"

```
public class objetoGeo
{
    protected Color cor;
    protected int x, y;

    public objetoGeo(Color cor, int x, int y)
    {
        this.cor = cor; this.x=x; this.y = y;
    }
    public Color retCor() {return cor;}
}
```

Atributo

Variável


Este exemplo mostra um uso típico do atributo this. Ele mostra um método cujo parâmetro formal possui o mesmo nome de um atributo de instância. Para distinguí-los, é necessário qualificar o atributo da instância com o atributo this.

Teste



- Teste do “this”
 - Crie uma classe qualquer (escolha um nome. Ex: Aluno)
 - Crie um atributo privado nome (String)
 - Crie um método
 - Receba por parâmetro um nome (String)
 - Atribua o valor recebido como parâmetro ao atributo nome
 - Não faça uso do “this”
 - Verifique o resultado
 - Depois, altere o nome do parâmetro
 - Verifique o resultado
 - Depois, use o this no caso anterior e no primeiro caso

Resultado do Teste

```
3 public class Aluno {  
4  
5     private String nome;  
6  
7     public void alteraNome(String nome) {  
8         nome = nome;  
9     }  
10  
11 }  
12  
13
```

 The assignment to variable nome has no effect

2 quick fixes available:

-  [Qualify left hand side](#)
-  [Qualify right hand side](#)

Demais Resultados

```
3 public class Aluno {  
4  
5     private String nome;  
6  
7     public void alteraNome(String valor){  
8         nome = valor;  
9     }  
10  
11 }
```

```
3 public class Aluno {  
4  
5     private String nome;  
6  
7     public void alteraNome(String valor){  
8         this.nome = valor;  
9     }  
10
```

```
3 public class Aluno {  
4  
5     private String nome;  
6  
7     public void alteraNome(String nome){  
8         this.nome = nome;  
9     }  
10  
11 }
```

Diferença

- Diferença entre “this” e “this(...)”
 - O primeiro destina-se a **resolver ambigüidades entre nomes** de variáveis e atributos. Já o segundo, destina-se a chamar construtor em sobrecarga (visto na aula de construtores).

Uso Combinado

Não se usa o "this" quando se trata de atributo Static

```
1 public class Pessoa {
2
3     private static String nome;
4
5     public String getNome() {
6         return nome;
7     }
8
9     public void setNome(String nome) {
10        Pessoa.nome = nome;
11    }
12
13    public static void main(String args[]) {
14
15        Pessoa p1 = new Pessoa();
16        Pessoa p2 = new Pessoa();
17
18        System.out.println(p1.getNome());
19        System.out.println(p2.getNome());
20
21        p1.setNome("Edmar");
22
23        System.out.println(p2.getNome());
24    }
25 }
```

Uso do modificador Static em um atributo de classe

Testes - Static e this

```
3 public class Aluno {  
4  
5     private static String nome;  
6  
7     public void alteraNome(String nome) {  
8         this.nome = nome;  
9     }  
10  
11 }
```

💡 The static field `Aluno.nome` should be accessed in a static way

3 quick fixes available:

- ➡ [Change access to static using 'Aluno' \(declaring type\)](#)
- ➡ [Remove 'static' modifier of 'nome'](#)
- @ [Add @SuppressWarnings 'static-access' to 'alteraNome\(\)'](#)

```
3 public class Aluno {  
4  
5     private static String nome;  
6  
7     public void alteraNome(String nome) {  
8         nome = nome;  
9     }  
10  
11 }
```

`Aluno.nome = nome;`

```
3 public class Aluno {  
4  
5     private static String nome;  
6  
7     public void alteraNome(String valor) {  
8         nome = valor;  
9     }  
10  
11 }
```

Modificadores e Palavras Reservadas

Orientação a Objetos - DCC025

Prof. Edmar Welington Oliveira
edmar.oliveira@ufjf.edu.br

Universidade Federal de Juiz de Fora - UFJF
Departamento de Ciência da Computação - DCC