

Unidade 1

Seção 3

Acesse este conteúdo pelo
smartphone



O que é isso?
Clique no código e saiba
mais.

Programação Orientada a Objetos

Webaula 3

Construtores e sobrecarga

Nessa webaula, veremos o conceito de alguns tipos especiais de métodos, como os construtores e a sobrecarga.

```

article { font-family: #fontFace; font-size: 14px; color: #000000; line-height: 1.2;
h1, h2, h3, h4, h5, h6 { font-family: #fontFace; line-height: 1.2; margin-bottom: 10px;
h1.unline, h2.unline, h3.unline, h4.unline, h5.unline, h6.unline { border-bottom: 1px solid
h1 { font-size: 24px; margin-bottom: 10px;
h2 { font-size: 20px; margin-bottom: 10px;
h3 { font-size: 16px; margin-bottom: 10px;
h4 { font-size: 14px; margin-bottom: 10px;
h5 { font-size: 14px; margin-bottom: 10px;
h6 { font-size: 14px; margin-bottom: 10px;
strong { font-weight: bold;
ul, ol { list-style-type: none;
margin: 0 0 10px 0;
padding: 0 0 0 20px;
li { list-style-type: disc; color: #000000;
a { color: #000000;
a:hover { text-decoration: underline; color: #000000;
}
ul, ol { margin: 0 0 10px 0;
li {
}
}
li:before { margin-top: 10px;
}
ol li { list-style-type: decimal;
form li:before { margin-top: 10px;
.article-title a { color: #000000;
}

```

Métodos construtores

Os construtores garantem que o código contido neles será executado antes de qualquer outro código em outros métodos, já que uma instância de uma classe só pode ser usada depois de ter sido criada com `new`, o que causa a execução automática do construtor.

- **Construtores** devem ter exatamente o mesmo nome da classe a que pertencem, inclusive considerando maiúsculas e minúsculas.
- **Construtores** não podem retornar valor algum, nem mesmo `void`. Por isso, devem ser declarados sem tipo de retorno.
- **Construtores** não devem receber modificadores (exemplo: `public` ou `private`). Eles serão públicos se a classe for pública.

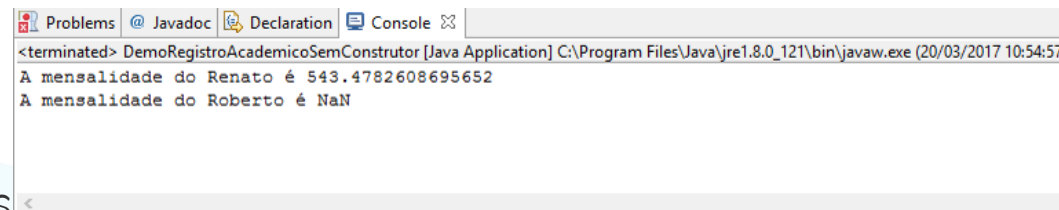
Explore a galeria e observe as classes RegistroAcademicoSemConstrutor e DemoRegistroAcademicoSemConstrutor, adaptadas de Santos (2003).

```
//Esta classe contém campos que representam dados simples de um
registro acadêmico.
public class RegistroAcademicoSemConstrutor {
    //declarando os campos da classe
    private String nomeDoAluno; //nome do aluno
    private int númeroDeMatrícula; //número de matrícula
    private byte códigoDoCurso; //código do curso (1 .. 4)
    private double percentualDeCobrança; //percentual em relação ao
    preço cheio, de 0 a 100%
    /**
     * O método inicializaRegistroAcademicoSemConstrutor recebe
     argumentos para inicializar
     * os campos da classe RegistroAcademicoSemConstrutor
     * @param n o nome do aluno
     * @param m o número de matrícula
     * @param c o código do curso
     * @param p o percentual da bolsa
     */
    public void inicializaRegistroAcademicoSemConstrutor (String n,
    int m, byte c, double p){
        nomeDoAluno = n;
        númeroDeMatrícula = m;
        códigoDoCurso = c;
        percentualDeCobrança = p;
    }
}
```

Fonte: adaptada de Santos (2003).

No exemplo da galeria, as classes fazem, basicamente, o cálculo de uma mensalidade, considerando o curso e o percentual de desconto concedido. A mensalidade do primeiro aluno foi calculada corretamente e a do segundo, os dados não foram informados, e a divisão por zero foi representada pelo Java como NaN. Por isso, é comum que o programador usuário seja forçado a passar dados para as instâncias criadas para que elas tenham sentido, e isso pode ser feito por meio de construtores.

Saída:






```
<terminated> DemoRegistroAcademicoSemConstrutor [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (20/03/2017 10:54:57)
A mensalidade do Renato é 543.4782608695652
A mensalidade do Roberto é NaN
```


Fonte: elaborada pelo autor.

Sobrecarga de métodos

De acordo com Arnold, Gosling e Holmes (2007, p. 86), cada método possui uma assinatura, que consiste de seu nome e número, mais tipos de seus parâmetros. Dois métodos podem ter o mesmo nome se eles tiverem diferentes número e tipos de parâmetros e, portanto, assinaturas diferentes. Essa característica é considerada sobrecarga (ou *overload*, em inglês) porque um único nome de método possui mais de um significado sobrecarregado.



Ao conceituar sobrecarga, Santos (2003, p. 53), assim se expressa: a possibilidade de criar mais de um método com o mesmo nome e assinatura diferente é conhecida como sobrecarga de métodos. A decisão sobre qual método será chamado quando existem dois ou mais métodos será feita pelo compilador, com base na assinatura dos métodos.



Sobreposição de métodos

A declaração de métodos com a mesma assinatura que métodos de classes ancestrais chama-se **sobreposição**.

A sobreposição de métodos, também é chamada de *override* ou superposição. É necessário recordar do conceito de **herança**.

O motivo para usarmos a sobreposição é que métodos de classes herdeiras geralmente executam tarefas adicionais que os mesmos métodos das classes ancestrais não executam (SANTOS, 2003).

Atributos e métodos estáticos em classes

Ao criarmos instâncias de uma classe por meio da palavra reservada `new`, cada instância da classe terá uma cópia de todos os campos declarados na classe. Por padrão, a modificação de um campo de uma instância de uma classe não afeta o valor do mesmo campo em outra instância. Esses campos são conhecidos como campos de instância.

É por meio da declaração e do uso de campos estáticos que conseguimos compartilhar um valor em todas as instâncias de uma mesma classe. Um campo estático é também conhecido como campo de classe, já que ele está associado à classe em que é definido, não à instância dessa classe.

Campos estáticos são declarados com o modificador `static`, , que deve ser posicionado antes do tipo de dado do campo e pode ser combinado com modificadores de acesso (`public` e `private`, por exemplo).

Exemplos de declaração de campos:

```
private int registroAcademico; (campo de instância)
```

```
static private int registroAcademico; (campo estático ou campo de classe).
```



Você viu a definição e utilização de métodos construtores e de sobrecargas de métodos, bem como sobreposição de métodos, atributos e métodos estáticos em classes.

Vídeo de encerramento



Você já conhece o Saber?

Aqui você tem na palma da sua mão a **biblioteca digital** para sua **formação profissional**.

Estude no celular, tablet ou PC em qualquer hora e lugar sem pagar mais nada por isso.

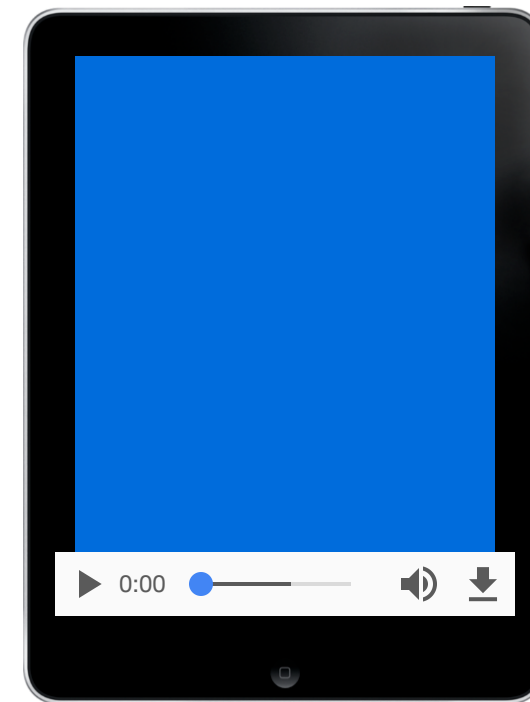
Mais de 450 livros com interatividade, vídeos, animações e jogos para você.



Android:
<https://goo.gl/yAL2Mv>



iPhone e iPad - IOS:
<https://goo.gl/OFWqcq>





Bons estudos!

