

## Material de Apoio Aula 4

### Método Construtor

Um objeto, ao ser instanciado, tem toda sua representação interna criada. Em particular, é criado um espaço de memória para armazenar seu *estado interno*, ou seja, seus atributos. Este espaço de memória necessita ser *inicializado*, atribuindo os valores iniciais que devem ser assumidos em cada um dos atributos do objeto. A responsabilidade de inicialização destes atributos é do método construtor. O método construtor é um método especial dentro da classe. Dizemos que este método é especial por três razões:

- É invocado implicitamente no momento da criação de um novo objeto;
- Não retorna nenhum valor;
- Possui o nome da própria classe.

Assim, a nossa classe Cofrinho, trabalhada em aulas anteriores, pode possuir o seguinte construtor:

<table border="1"><tr><td>Cofrinho</td></tr><tr><td>saldo</td></tr><tr><td>Cofrinho() deposita(int) retira(int) mostraSaldo : int</td></tr></table>	Cofrinho	saldo	Cofrinho() deposita(int) retira(int) mostraSaldo : int	<pre>class Cofrinho {     private int saldo;      public Cofrinho() { // Quando o objeto for         saldo = 0;     // criado, o cofrinho está     }                 // vazio...     public void deposita( int v ) {         saldo = saldo + v;     }     public void saca( int v ) {         saldo = saldo - v;     } }</pre>
Cofrinho				
saldo				
Cofrinho() deposita(int) retira(int) mostraSaldo : int				

1. Implemente a classe Cofrinho no BlueJ.
  - Crie um objeto da classe Cofrinho.
  - Deposite 5 pilas. Inspeção o valor dos atributos.
  - Saque 4 pilas. Inspeção o valor dos atributos.
  - Crie um segundo objeto da classe Cofrinho. Inspeção os atributos dos dois objetos cofrinhos.
  - Deposite 10 pilas neste segundo cofrinho. Inspeção o valor dos atributos dos dois objetos cofrinhos – observe que apenas o atributo saldo de um dos objetos, no caso o segundo, foi alterado.
2. Implemente uma classe para objetos capazes de calcular o desempenho de uma turma. Este desempenho deve ser apresentado, segundo a necessidade do usuário, como média aritmética  $ma = (nota_1 + nota_2 + nota_3 + \dots + nota_n)/n$ , ou harmônica global  $mh = n / (1/nota_1 + 1/nota_2 + 1/nota_3 + \dots + 1/nota_n)$ , onde  $n$  é o número de alunos da turma. Utilize o BlueJ para desenvolver e testar sua classe. Para implementar esta classe, identifique: os atributos necessários, o valor inicial para os atributos e os métodos que devem ser implementados. Utilize ponto-flutuante (`double`) para representar notas dos alunos e as médias.
3. Tema de casa: faça um programa teste (que contenha o método `main`) para testar sua classe Cofrinho com comandos de linha (`javac` e `java`).

## Sobrecarga de método

A assinatura de um método é determinada pelo nome do próprio método e pelo *número e tipo dos parâmetros* que o método correspondente recebe. No exemplo da classe `Cofrinho`, o método `deposita` tem como assinatura `deposita(int)`, assim como o método construtor possui a assinatura `Cofrinho()`. Um recurso bastante poderoso em linguagem orientadas a objetos é a possibilidade de *sobrecarregar métodos*. A sobrecarga de método significa atribuir diferentes *assinaturas* aos métodos. Assim, um método pode receber diferentes conjuntos de parâmetros. Por exemplo, poderíamos propor uma nova implementação para classe `Cofrinho` como segue.

<table border="1"><tr><td>Cofrinho</td></tr><tr><td>saldo</td></tr><tr><td>Cofrinho() Cofrinho(int) deposita(int) deposita(int, int) retira(int) mostraSaldo : int</td></tr></table>	Cofrinho	saldo	Cofrinho() Cofrinho(int) deposita(int) deposita(int, int) retira(int) mostraSaldo : int	<pre>class Cofrinho {     private int saldo;      public Cofrinho() { // Quando o objeto for         saldo = 0;    // criado, o cofrinho está     }                // vazio...     public Cofrinho( int v ) { // Quando o objeto for         saldo = 0;    // criado, o cofrinho recebe     }                // um deposito inicial     public void deposita( int v ) {         saldo = saldo + v;     }     public void deposita( int v1, int v2 ) {         saldo = saldo + v1 + v2;     }     public void saca( int v ) {         saldo = saldo - v;     } }</pre>
Cofrinho				
saldo				
Cofrinho() Cofrinho(int) deposita(int) deposita(int, int) retira(int) mostraSaldo : int				

4. Complete a classe `Cofrinho` no BlueJ utilizando a possibilidade de sobrecarregar métodos.
5. Com auxílio do BlueJ, implemente uma classe para conter informações sobre um funcionário de uma empresa (classe `Funcionario`).
  - Quais são os atributos desta classe? Inclua entre eles, o salário que o funcionário deve receber por hora trabalhada.
  - Implemente, para esta classe, pelo menos três métodos construtores: um que receba apenas o nome do funcionário e assuma valores *default* para os demais atributos (assuma que o funcionário deve receber 2 pilas por hora trabalhada); o segundo construtor deve receber, além do nome, o valor que o referido trabalhador deve receber por hora trabalhada.
  - Identifique e implemente demais métodos que achar conveniente para um objeto da classe `Funcionario`.
6. Tema de casa: faça um programa teste (que contenha o método `main`) para testar sua classe `Funcionario` com comandos de linha (`javac` e `java`). Inclua um novo construtor, que receba, como parâmetro, um objeto da classe `Entrada` (material de apoio 1) e leia, do teclado, as informações necessárias para inicializar o estado interno do objeto.
7. Implemente uma classe `Carro`. Objetos desta classe possuem alguns atributos, tais como: matrícula, consumo por litro, contador de quilometragem e quantidade de litros de combustível no tanque (defina demais atributos conforme sua necessidade específica). Esta classe deve conter, no mínimo, métodos para realizar os seguintes serviços:
  - Abastecer o carro;
  - Fazer o carro andar um número determinado de quilômetros;
  - Fazer o carro andar uma quantidade de quilômetros definida aleatoriamente (`random` da classe `Math`)
  - Mostrar quantidade de combustível no tanque;
  - Mostrar o status do veículo: quantidade de combustível, quilômetros percorridos e quantidade de quilômetros que podem ser percorridos com a quantidade de combustível disponível.