UNIP
UNIVERSIDADE PAULISTA

Linguagem de Programação Orientada a Objeto

Construtores e Sobrecarga

Professora Sheila Cáceres

Sobrecarga (overloading)

- Na mesma classe podemos definir vários métodos com o mesmo nome → Sobrecarga
- O tipo de retorno tem que ser o mesmo
- A quantidade ou o tipo dos parâmetros tem que ser diferentes

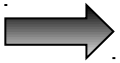
```
public class Triangulo {
	private int lado1;
	private int lado2;
	private int lado3;

	public int calculaPerimetro(){
		return lado1+lado2+lado3;
	}

	public int calculaPerimetro(int l1, int l2, int l3) {
		return l1+l2+l3;
	}
}
```

Construtores

- Servem para inicializar objetos.
- Determina que ações devem ser executadas na criação de um objeto.
- Cuida da alocação de todos os recursos necessários para o objeto e retorna uma instância do objeto
- Frequentemente recebem valores de parâmetros externos para inicializar os atributos.
- Uma classe pode ter um ou mais construtores (overloading - sobrecarga)

Construtores - Exemplo

- Sempre têm o mesmo nome da suas classes.
- Não pode especificar um valor de retorno (nem void).

Por Exemplo: 

```
public class Triangulo {
    private int lado1;
    private int lado2;
    private int lado3;

    public Triangulo(int la1, int la2, int la3){
        lado1 = la1;
        lado2 = la2;
        lado3 = la3;
    }
}
```

- Por padrão, o compilador fornece um construtor padrão sem parâmetros em qualquer classe que não inclua explicitamente um construtor

Referência this

- Indica que um dado atributo é próprio da classe.

```
public class Triangulo {
    private int lado1;
    private int lado2;
    private int lado3;

    public Triangulo(int la1, int la2, int la3){
        lado1 = la1;
        lado2 = la2;
        lado3 = la3;
    }
}
```

Construtor anterior

```
public class Triangulo {
    private int lado1;
    private int lado2;
    private int lado3;

    public Triangulo(int lado1, int lado2, int lado3) {
        this.lado1 = lado1;
        this.lado2 = lado2;
        this.lado3 = lado3;
    }
}
```

Usando this

Construtores - Exemplo

```
public class Programa {
    public static void main(String args[]) {
        Triangulo t = new Triangulo();
    }
}
```

ERRO! A classe Triangulo possui
apenas um construtor que requer
3 argumentos inteiros.
O construtor padrão não foi criado porque
Triangulo já tem construtor

Construtores - Exemplo

```
public class Programa {
    public static void main(String args[]) {
        Triangulo t1 = new Triangulo(5, 2, 6);
        Triangulo t2 = new Triangulo(3, 4, 5);
        Triangulo t3 = new Triangulo(5, 3, 3);
    }
}
```

Agora SIM!

Construtores

- O construtor pode garantir que os dados com os quais inicializaremos as variáveis são adequados. Essa prática também é aconselhável para os métodos (principalmente set e get).

```
public class Triangulo {
    private int lado1;
    private int lado2;
    private int lado3;

    public Triangulo(int la1, int la2, int la3){
        if (la1>0 && la1<(la2+la3)){
            lado1 = la1;
        }
        if (la2>0 && la2<(la1+la3)){
            lado2 = la2;
        }
        if (la3>0 && la3<(la1+la2)){
            lado3 = la3;
        }
    }
}
```

Perceba que o construtor Triangulo se assegura que os valores iniciais sejam válidos para inicializar a variável.
Caso o valor digitado não satisfaça a condição, a variável continua com seu valor padrão 0.0.

Sobrecarga de construtores

- Os métodos de uma classe podem ser **sobrecarregados**.
- Para sobrecarregar um **método construtor**, simplesmente forneça uma definição separada de cada um dos métodos sobrecarregados.
- Tais métodos **devem** possuir **listas de parâme-tros diferentes**.

Sobrecarga de construtores

```
class Empregado {
    private String nome;
    private double salario;
    public Empregado(String n, double s) {
        nome = n;
        salario = s;
    }
    public Empregado(String n) {
        nome = n;
        salario = 500;
     }
}
```

```
public class TestEmpregado {
    public static void main(String args[]) {
        Empregado j = new Empregado("Joao",1500);
        Empregado m = new Empregado("Maria");
    }
}
```

Destrutores

- Muitas linguagens OO têm métodos denominados Destrutores, que desalocam todos os recursos alocados pelo construtor.
- São chamados quando um objeto deixa de existir
- Java cuida de liberar todos os recursos alocados para um objeto que não é mais referenciado através do *garbage collector (GC)*
- O GC de Java possui uma gerência automática de memória, isto é, quando um objeto não é mais referenciado pelo programa, ele é automaticamente coletado (destruído).
- Existem algumas situações de limpeza que o GC não efetua. Por exemplo, arquivos abertos ou conexão com Banco de Dados. Neste caso pode-se definir um método chamado de *finalizer*. Este método é chamado imediatamente pelo GC antes do objeto ser destruído.
- O Garbage collector pode ser chamado explicitamente através do método System.gc();

Exercício

- Crie uma classe chamada Funcionario.
- Atributos: Nome, Sobrenome, Salario
- Métodos
  - Implemente os métodos get e set para os atributos.
  - Implemente dois métodos construtores
    - O primeiro recebe os parâmetros Nome, Sobrenome, Salario
    - O segundo recebe os parâmetros nome e sobrenome (o salario será inicializado com 0)
- Testando:
  - Escreva uma classe de teste chamada FuncionarioTeste.
  - Crie dois objetos Funcionario e exiba o salário anual de cada objeto. Então dê a cada um aumento de 10% e exiba novamente o salário anual de cada Empregado (no main da classe FuncionarioTeste).

Referências

- Java: Como programar.
  Autores: H. M. Deitel e P. J. Deitel
  Editora: Pearson – 6a Edição
  Capítulo 3: Introdução a Classes e Objetos

- Programação Orientada a Objetos com Java, David J. Barnes and Michael Kolling. Pearson 2004.

- Nota: O material da apresentação foi extraido de algumas das fontes aqui apresentadas