

Classes e Construtores

Prof. Hugo de Paula



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Departamento de Ciência da Computação

Sumário

- 1 Módulos e classes
 - Classes e objetos
- 2 Exemplo: Estoque de Produtos
 - Criando a classe Produto
 - Definindo a classe Produto
 - Semântica de referência
- 3 Construtores
 - Construtores
 - Usando construtores



Modularidade

Ocultação de informação

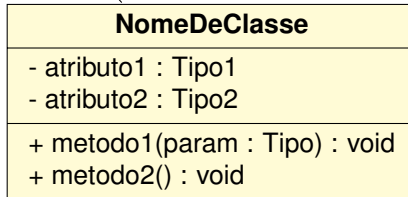
Toda informação a respeito de um módulo deve ser privativa do módulo, exceto se for explicitamente declarada como pública.

- Modularidade produz encapsulamento.
- Entidades semelhantes formam uma classe de objetos.
 - Um objeto representa uma entidade referenciável (identificada).



Classes e objetos

Nome de classe é obrigatório.



Compartimentos de atributos ou métodos são opcionais.

UML (*Unified Modeling Language*) permite representar classes e objetos para fins de modelagem de dados.



Math Object

Math
+E : double = 2.718 +LN2 : double = 0.693 +LN10 : double = 2.302 +LOG2E : double = 1.442 +LOG10E : double = 0.434 +PI : double = 3.141592653589793 +SQRT1_2 : double = 0.707 +SQRT2 : double = 1.141
+abs(entrada x : double) : double +acos(entrada x : double) : double +asin(entrada x : double) : double +atan(entrada x : double) : double +ceil(entrada x : double) : double +cos(entrada x : double) : double +exp(entrada x : double) : double +floor(entrada x : double) : double +log(entrada x : double) : double +max(entrada x : double, entrada y : double) : double +min(entrada x : double, entrada y : double) : double +pow(entrada x : double, entrada y : double) : double +random() : double +round(entrada x : double) : double +sin(entrada x : double) : double +sqrt(entrada x : double) : double +tan(entrada x : double) : double +toSource() : double +valueOf() : double

<html>

<body>

<script type="text/javascript">

```
document.write(Math.round(0.60) + "<br />");  
document.write(Math.round(0.50) + "<br />");  
document.write(Math.round(0.49) + "<br />");  
document.write(Math.round(-4.40) + "<br />");  
document.write(Math.round(-4.60));
```

</script>

</body>

</html>



Exemplo: Estoque de Produtos

Criar uma classe `Produto` para um sistema de gerenciamento de estoque.

- Atributos:

- `descricao` : `String`
- `preco` : **`float`**
- `quant` : **`int`**

- Métodos:

- `emEstoque()` : `bool`
- `inicializaProduto(String, float, int)`



Definindo a classe Produto: UML

Produto
+ descricao : String + preco : float + quant : int
+ inicializaProduto(descricao : String, preco : float, quant : int) : void + emEstoque() : boolean



Definindo a classe Produto

```
class Produto {  
    String descricao;  
    float preco;  
    int quant;  
  
    boolean emEstoque()  
    {  
        return (quant > 0);  
    }  
  
    void inicializaProduto(String d, float p, int q)  
    {  
        descricao = d;  
        preco = p;  
        quant = q;  
    }  
}
```




Usando a classe Produto

```
class Aplicacao {  
    public static void main(String args[])  
    {  
        Produto p = new Produto();  
  
        p.descricao = "Shulambs";  
        p.preco = 1.99F;  
        p.quant = 200;  
  
        System.out.println("Produto: " + p.descricao);  
        System.out.println("Preço: " + p.preco);  
        System.out.println("Estoque: " + p.quant);  
  
        if (p.emEstoque())  
            System.out.println("Produto em estoque.");  
    }  
}
```



Semântica de referência

- Conceito de referência:
 - Uma referência é um ponteiro (apontador) constante.
 - Referências são chamadas de alias (sinônimos).
- Em Java, uma instância de uma classe é interpretada como uma referência para um objeto e não o objeto propriamente dito.



Criando Objetos: C++ vs Java

C++:

- Declaração de um Objeto:
 - `nomeClasse nomeObjeto;`
 - Objeto é inicializado no momento da declaração e um estado lhe é atribuído.
- Declaração de apontadores:
 - `nomeClasse *nomeObjeto;`
 - Cria-se o apontador mas não se cria o objeto.
 - Criação do objeto através da cláusula **new**.
 - Apontador pode apontar para qualquer objeto.



Criando Objetos: C++ vs Java

Java:

- Declaração de um Objeto:
 - nomeClasse nomeObjeto;
 - Semelhante ao apontador no C++.
 - Cria-se a referência mas não se cria o objeto.
 - Criação do objeto através da cláusula **new**.
 - Uma vez criado o objeto, a referência não pode ser manipulada numericamente.



Criando Objetos

Produto p;

- Cria-se uma referência para um objeto do tipo `Produto`, mas não se aloca a memória para armazenar o objeto.
- Variável `p` aponta para NADA (`null`)

`p = new Produto();`

- Cria-se efetivamente o objeto `Produto`.
- Faz com que a referência `p` aponte para `Produto`.



Construindo um objeto

- Objetos são instâncias de uma classe:
 - Lê-se instância como sendo um elemento com o tipo da classe e um estado corrente individual.
- Exemplo:
 - Classe \rightarrow Produto (tipo com descricao, preco e quantidade)
 - Objeto de Produto $\rightarrow p = (\textit{Shulambs}; R\$1,99; 200)$
- Ao criar um objeto sua memória é inicializada.
- Se não for definido um modo de inicialização o compilador usa valores padrão. Ex:
 - `p = new Produto();` cria $(\textit{null}, 0.0, 0)$



Construtores

- Construtores são usados para inicializar objetos com valores diferentes do padrão.
- Construtores:
 - Possuem o mesmo nome da classe.
 - Não possuem valores de retorno.
- Uma classe pode ter de 0 a muitos construtores.



Construtores

- Razões para se usar construtores especializados:
 - Algumas classes não possuem estado inicial aceitável sem parâmetros.
 - Fornecer um estado inicial é conveniente e aceitável quando da construção de alguns tipos de objetos.
 - Construir um objeto aos poucos pode ser desgastante de forma que pode ser conveniente que se tenha um estado inicial correto quando forem criados.
 - Um construtor que não é público restringe quem irá criar objetos utilizando-o. Pode-se assim restringir o uso de sua classe.



Classe Produto: usando construtores

```
class Produto {  
    ...  
  
    Produto(String d, float p, int q)  
    {  
        if (d.length() >= 3)  
            descricao = d;  
        if (p > 0)  
            preco = p;  
        if (q >= 0)  
            quant = q;  
    }  
  
    Produto() {  
        descricao = "Novo Produto";  
        preco = 0.01F;  
        quant = 0;  
    }  
}
```



Usando construtores da classe Produto

```
class Aplicacao {  
    public static void main(String args[])  
    {  
        Produto p1 = new Produto();  
  
        Produto p2 = new Produto("Shulambs", 1.99F, 200);  
  
        System.out.println("Produto: " + p1.descricao);  
        System.out.println("Preço: " + p1.preco);  
        System.out.println("Estoque: " + p1.quant);  
  
        System.out.println("Produto: " + p2.descricao);  
        System.out.println("Preço: " + p2.preco);  
        System.out.println("Estoque: " + p2.quant);  
    }  
}
```