

Classes e objetos: Utilizando Construtores

Prof. Pedro Pongelupe



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
- 3 Construtores
 - Motivação
 - Usando construtores



Orientação por objetos

- Orientação por objetos abstrai o mundo real utilizando objetos que interagem entre si
- Utilizamos o princípio da decomposibilidade para desenvolver sistemas modulares

Exemplos

- Quais são os componentes de uma gestão de estoque?
- Quais são os componentes de uma rede social?
- Quais são os componentes de um(a) <problema>?



Classe

O quê é?

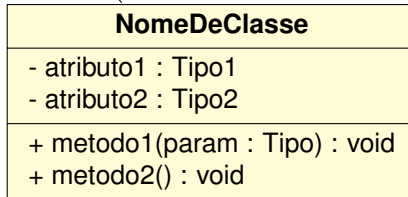
Representação de um Tipo Abstrato de Dados (TAD).

- Atributos → Dados/Características
- Métodos → Ações/Comportamentos



Classe: Representação UML

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.



Classe

Exemplo

Classe Carro

- Atributos: Modelo, ano, velocidade, quilômetros rodados, combustível
- Métodos: Abastecer, alterar velocidade

Carro
+ modelo : String + ano : int + velocidade : int + quilometrosRodados : int + combustivel : double
+ abastecer(quantidade: double) : double + alterarVelocidade(velocidade: int) : void



objeto

O quê é?

Um objeto representa uma entidade referenciável de uma classe. É uma **instância** de uma classe.

- carro1 → ("Chevette Tubarão";1977;120;200000;2.5)
- carro2 → ("Vectra GT";2009;0;57000;50.0)

Carro

+ modelo : String
+ ano : int
+ velocidade : int
+ quilometrosRodados : int
+ combustivel : double



Exemplo: Estoque de Produtos

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

- Atributos:

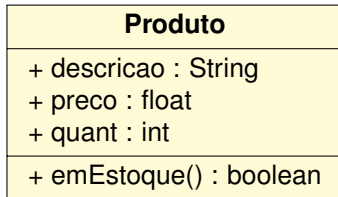
- `descricao` : `String`
- `preco` : **`float`**
- `quant` : **`int`**

- Métodos:

- `emEstoque()` : **`boolean`**



Definindo a classe Produto: UML





Definindo a classe Produto

```
class Produto {  
    String descricao;  
    float preco;  
    int quant;  
  
    boolean emEstoque() {  
        return (quant > 0);  
    }  
}
```



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.");  
        }  
    }  
}
```



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 (*null*, 0.0, 0)

E se a gente quiser criar um objeto com outros valores?



Aumentando a classe Produto: UML

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



Aumentando 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 tunada

```
class Aplicacao {  
    public static void main(String args[]) {  
        Produto p = new Produto();  
  
        p.inicializaProduto("Shulambs", 1.99F, 200);  
        // 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.");  
        }  
    }  
}
```




Hmmmmmmmmmmmmmmmmmm

Code Smell

CODE SMELLS ARE
SYMPTOMS OF POOR
DESIGN OR
IMPLEMENTATION CHOICES

[Martin Fowler]





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.
- O Construtor com parâmetros para nossa classe Produto:

```
Produto(String d, float p, int q) {  
    descricao = d;  
    preco = p;  
    quant = q;  
}
```

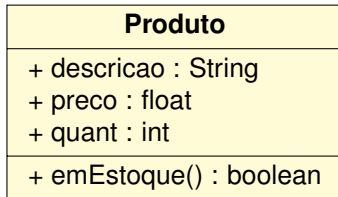


Classe Produto versão final

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



UML final da classe Produto





Usando construtores da classe Produto

```
class Aplicacao {  
    public static void main(String args[]) {  
        Produto p1 = new Produto();  
  
        Produto p2 = new Produto("Shulams", 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);  
    }  
}
```



Classe Produto versão final 2: agora vai!

```
class Produto {  
    String descricao;  
    float preco;  
    int quant;  
  
    Produto() {  
        descricao = "Novo produto";  
        preco = 4.20;  
        quant = 0;  
    }  
  
    Produto(String d, float p, int q) {  
        descricao = d;  
        preco = p;  
        quant = q;  
    }  
  
    boolean emEstoque() {  
        return (quant > 0);  
    }  
}
```



Usando construtores da classe Produto

```
class Aplicacao {  
    public static void main(String args[]) {  
        Produto p1 = new Produto();  
  
        Produto p2 = new Produto("Shulams", 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);  
    }  
}
```



Obrigado!!

Muito obrigado pela atenção! Alguma dúvida? Bora praticar!!!

"De derrota em derrota até a vitória final."