Classes e objetos

Prof. Pedro Pongelupe

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Departamento de Ciência da Computação

Sumário

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>?

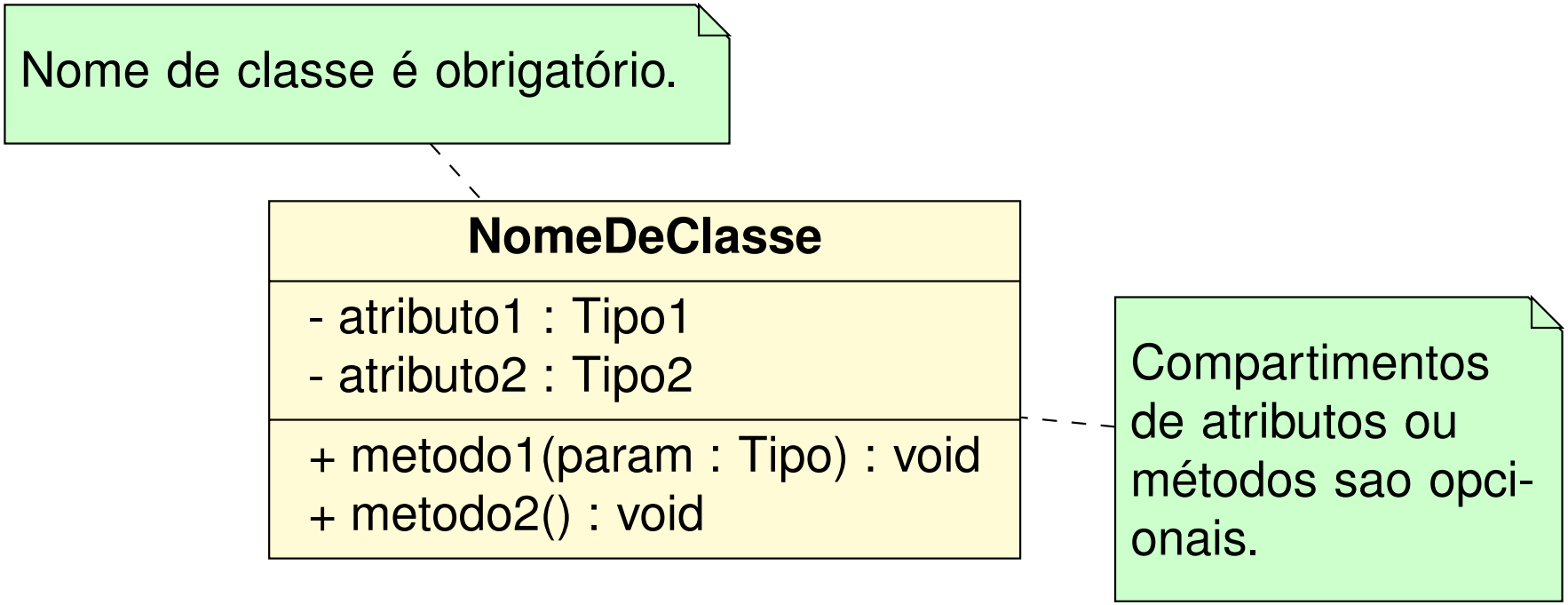
Classe

O quê é?

Representação de um Tipo Abstrato de Dados (TAD).

- Atributos $\rightarrow$ Dados/Características
- Métodos $\rightarrow$ Ações/Comportamentos

Classe: Representação UML



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

Classe

Exemplo

Classe Carro

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

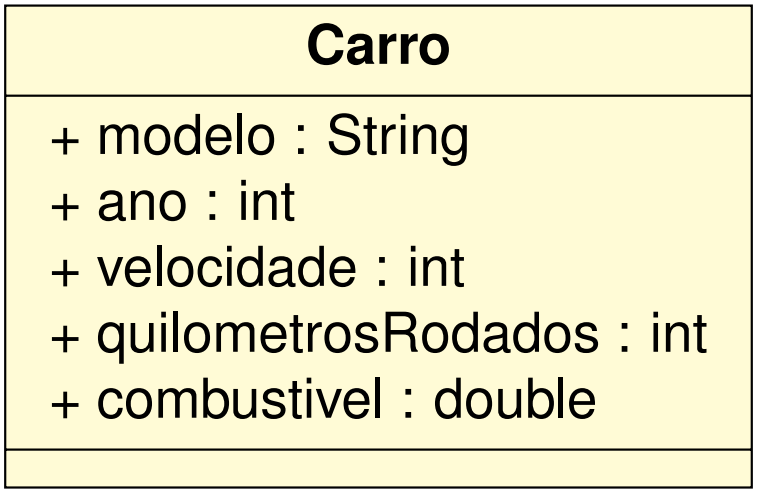
Carro
+ modelo : String + ano : int + velocidade : int + quilometrosRodados : int + combustivel : double
+ abastecer(quantidade: double) : double + alterarVelocidade(velocidade: int) : void

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)



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
  - incializaProduto(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 $\longrightarrow$ Produto (tipo com descricao, preco e quantidade)
  - Objeto de Produto $\longrightarrow$ $p = (Shulambs; R\$1,99; 200)$
- 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 $(null, 0.0, 0)$

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 publico 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);

    }
}
```