Introdução

Universidade Federal de Uberlândia

Programação Orientada a Objetos

Prof. Fabiano Dorça

Introdução

- Definições iniciais:
  - classe,
  - objeto,
  - métodos,
  - atributos.

Introdução

- Ao escrever um programa OO, cria-se um modelo do mundo real.
- Este modelo é descrito por **partes**, que são os objetos que aparecem no domínio do problema.

Introdução

- Tais objetos possuem **características** próprias, denotadas por **atributos**.

- E também possuem **funções** (comportamentos específicos), dados por **métodos**

Introdução

- Os objetos podem ser **categorizados**, agrupados, e uma classe descreve todos os **objetos de um tipo particular**.

- Desta forma, uma **classe** descreve um **tipo** de objeto, representando na programação OO um tipo.

Introdução

- Normalmente referimos a um objeto particular como uma instância de uma classe.
- Na POO, instância é sinônimo de objeto.

O paradigma orientado a objetos

- Em POO:
  - Um conceito do mundo real é um objeto.
  - Escreve-se código organizado em torno de objetos (conceitos), não de funções.
  - POO → modelagem conceitual
  - Procedimental → modelagem funcional (e.g. DFD)

O paradigma orientado a objetos

- Objetos são formados por:
  - dados que retratem suas características e seu estado;
  - operações que implementam seus métodos.
- A coleção de métodos de um objeto invocáveis por outros objetos é denominada *interface*

O paradigma orientado a objetos

- Objetos de um mesmo **tipo** possuem os mesmos comportamentos.
- Então, dizemos que objetos de mesmo tipo pertencem à mesma **classe.**
- Ou seja, uma classe é um tipo a partir do qual objetos são criados.

O paradigma orientado a objetos

- A classe define:

  - Os elementos de dados que um objeto contém (atributos).

  - O comportamento que o objeto possui (métodos).

  - A maneira como esses elementos de dados e métodos podem ser acessados (interface).

O paradigma orientado a objetos

Mas, afinal, qual é a diferença entre Classe e Objeto?

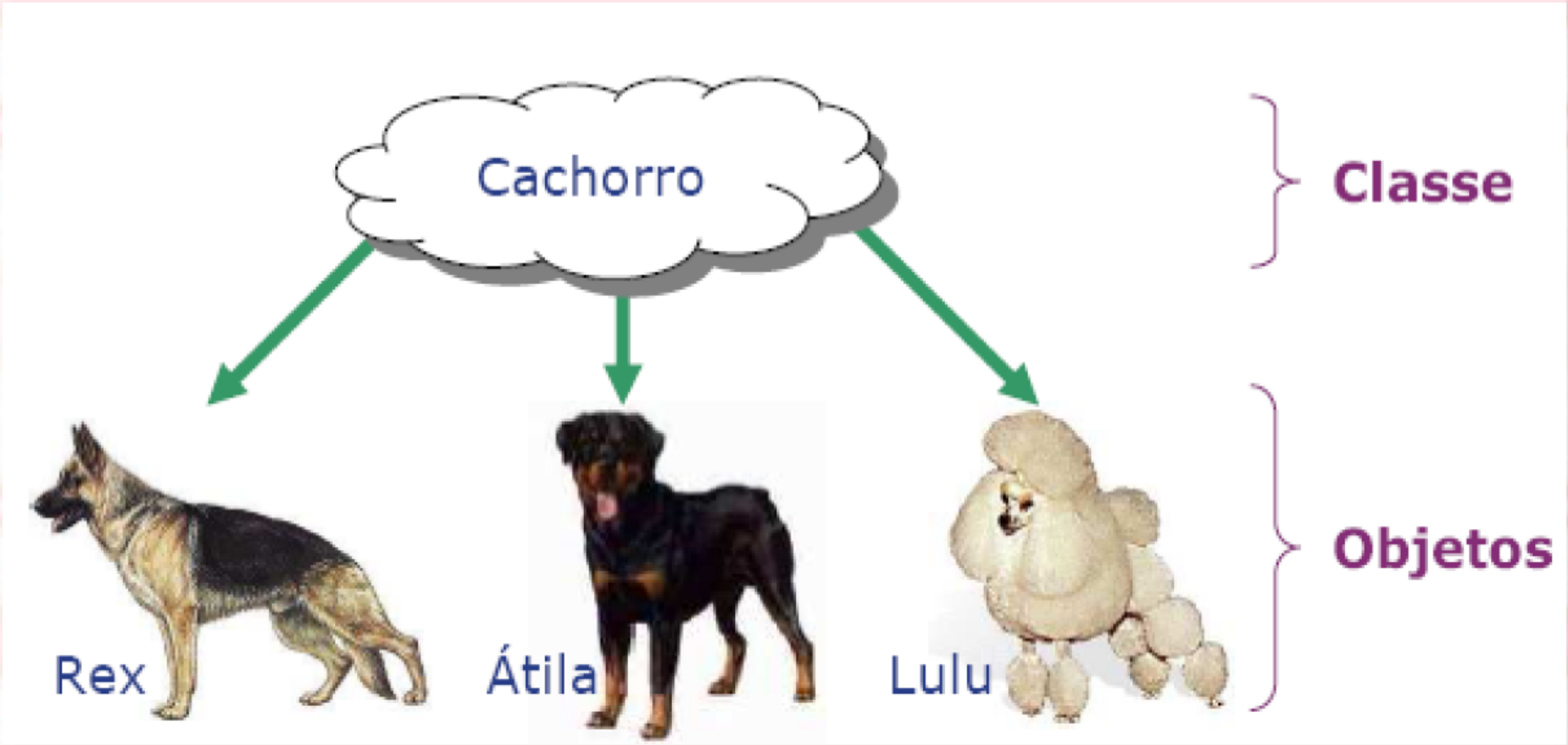
a) Classe:

- *é definição do tipo;*
- *representa um conjunto de objetos de mesmo tipo;*
- *Classe = {obj01, obj02, obj3, … , objN}*

b) Objeto:

- *é cada instância derivada da classe;*
- *é um elemento do conjunto representado pela classe.*

O paradigma orientado a objetos



Modelagem Conceitual

- Análise dos objetos do mundo real e suas inter-relações.
- Descobrir os conceitos (classes) que compõe o domínio do problema.
- Elementos básicos: Conceitos (classes), Atributos, Associações.

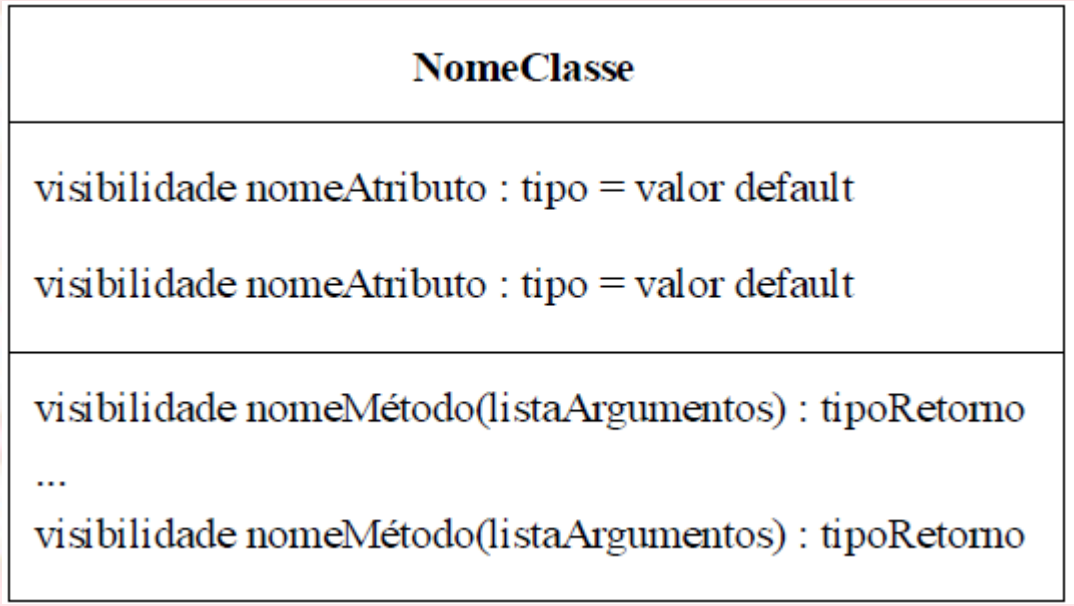
Modelagem Conceitual

- Conceitos
  → são substantivos que representam “coisas” que o sistema manipula
  → são fortes candidatos a classes.

- Diagrama de classes da UML
  → ferramenta para modelagem conceitual e projeto da arquitetura do software.

Modelagem Conceitual

- Representação UML para uma classe:



Tipos não primitivos

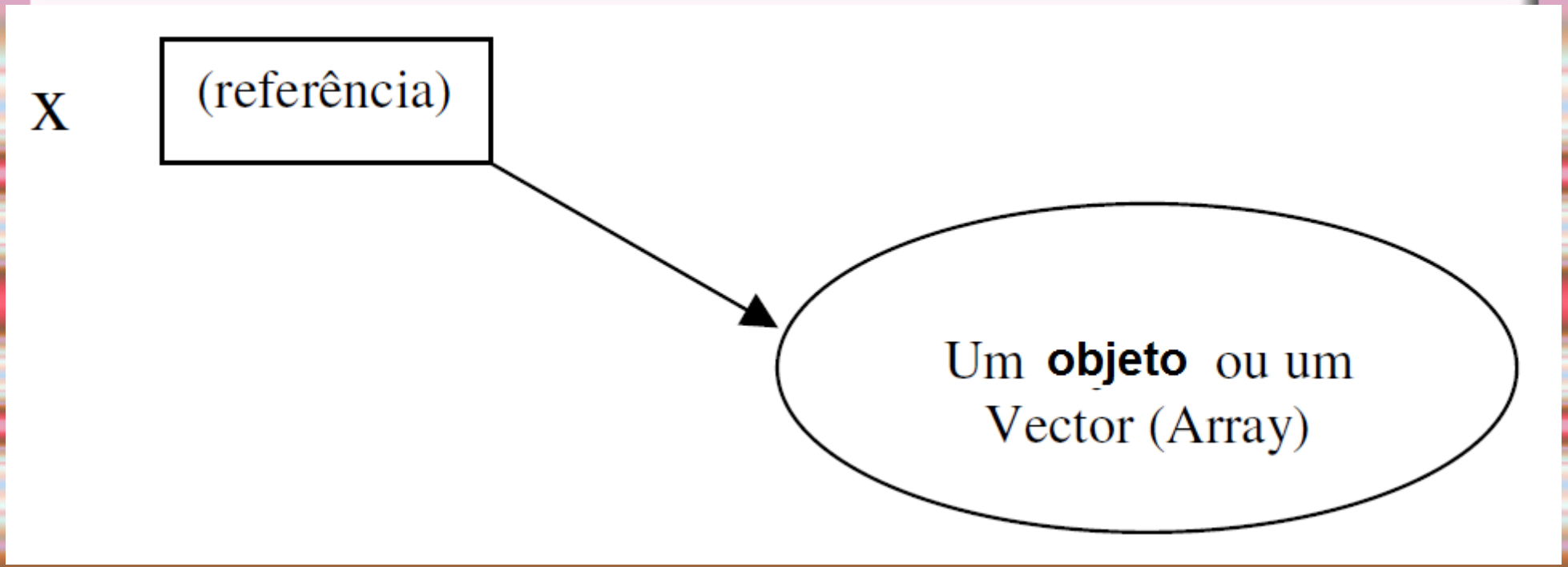
Para cada tipo primitivo existe uma classe que encapsula esse tipo:
boolean **- Boolean**
byte **- Byte**
char **- Character**
short **- Short**
int **- Integer**
long **- Long**
float **- Float**
double **– Double**

Exemplo: instanciação de um objeto da classe Integer
Integer i = new Integer(10);

Tipos referenciados

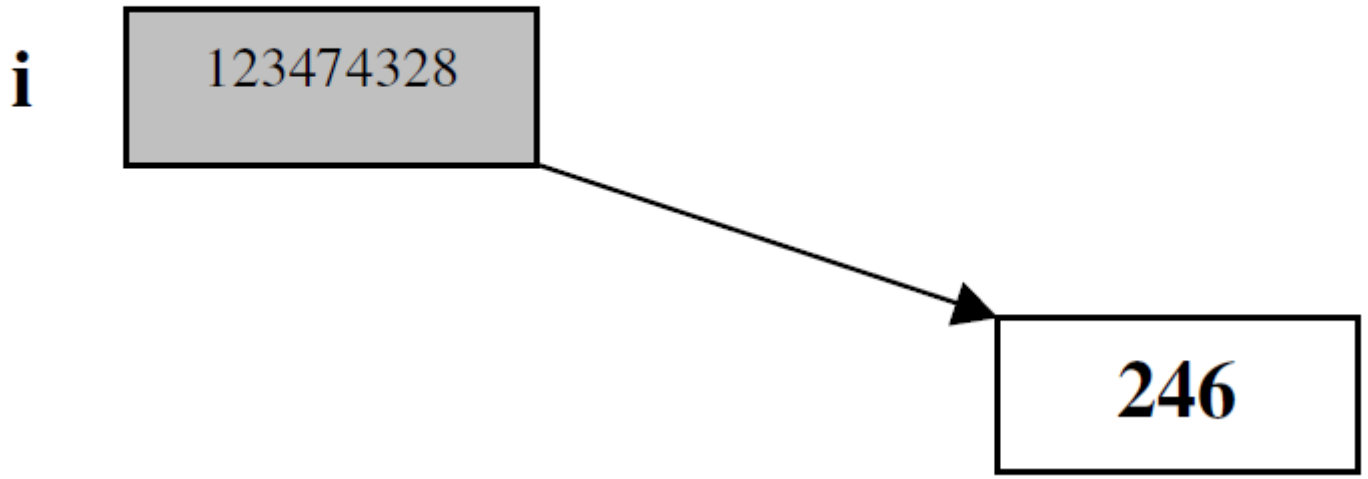
Vetores e classes são tipos referenciados.

O **valor de uma variável de um tipo referenciado é uma referência** para o valor ou conjunto de valores representados pela variável.



Tipos referenciados

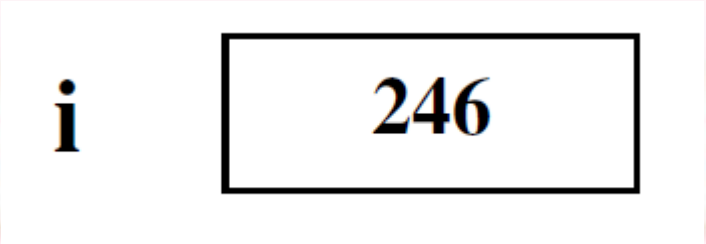
Exemplo: *Integer i = new Integer(246);*



Tipos não referenciados

Em contraste, o **valor de uma variável de um tipo primitivo é o próprio valor**.

Exemplo: *int i = 246;*



Operadores Aritméticos

op1 + op2, op1 - op2, op1 * op2, op1 / op2

op1 % op2 (resto da divisão inteira)

op++ e ++op incrementar (de uma unidade)

op-- e --op decrementar (de uma unidade)

Operadores relacionais e condicionais

op1 > op2 op1 >= op2

op1 < op2 op1 <= op2

op1 == op2 op1 != op2

op1 && op2 conjunção

op1 || op2 disjunção

!op negação

op1 ^ op2 disjunção exclusiva (XOR)

Java

- Sintaxe básica:
  - Criação de classes
  - Criação de atributos
  - Criação de métodos
  - Vetores
  - Instanciação de Objetos

Java

Definindo Classes

- Para definir uma classe use a palavra chave *class* e o nome da classe.
- Exemplo:

```
class Minhaclasse{
...
}
```

Java

Declarando um Array:

```
String difficult[];
Point hits[];
int temp[];
```

Java

Criando Objetos Arrays:

O operador *new* para cria uma nova instância de um array, por exemplo:

```
int[] temps = new int[99];
```

Java

Acessando os Elementos do Array

Os arrays em Java sempre iniciam-se na posição 0 como no C++. Por exemplo:

```
String[] vet = new String[10];

vet[10]=”erro...”;
```

Descobrir tamanho do array vet:

```
vet.length;
```

Java

Exemplo:

```
class Fornecedor
{
    String nome;

    void verNome(){
        System.out.println(nome);
    }
}
```

Java

```
class Produto
{
   int[] codigos;
   Fornecedor[] fornecedores;

   void criaVetor(int n){
      codigos = new int[n];
      fornecedores = new Fornecedor[n];
   }
}
```

Java

```
class Principal
{
    public static void main(String args[]){
        Produto p = new Produto();
        p.criaVetor(10);
        p.codigos[0] =1;
        System.out.println(p.codigos[0]);
        Fornecedor f = new Fornecedor();
        f.nome = "Martins";
        p.fornecedores[0] = f;
        f.verNome();
    }
}
```

Java

Arrays Multidimensionais

É possível declarar e criar um array de arrays e acessá-los como no estilo-C.

```
int coords[][]= new int[12][12];
coords[0][0] = 1;
coords[0][1] = 2;
```

Java

Desvio condicional

```
if ( x < y)
     System.out.println(" x e menor do que y");
else
     System.out.println(" y e maior);
```

Nota técnica: A diferença entre o if em Java e C ou C++ é que o teste deve retornar um valor booleano(true ou false).

Java

Blocos

```
if (x> w)
{ // inicio do bloco
    int y=50;
    System.out.println("dentro do bloco");
    System.out.println("x:" + x);
    System.out.println("y:" + y);
} // final do bloco
```

Java

Looping For

O loop em Java tem esta sintaxe:

```
for(inicialização; teste; incremento){
   bloco de comandos;
}
```

Java

Loop While

O *while* é usado para repetir um comando, ou um conjunto de comando enquanto a condição é verdadeira.

```
While (condição){
    bloco de comandos;
}
```

- Ok!