Instituto de computação uff

Introdução a Orientação a Objetos

Leonardo Gresta Paulino Murta

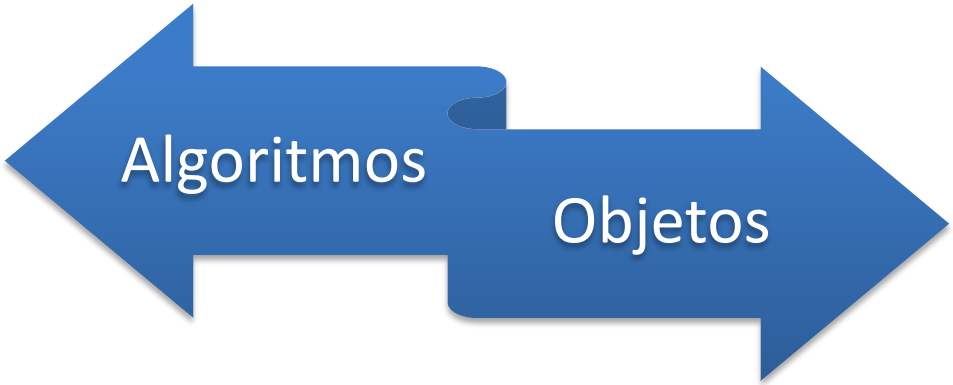
leomurta@ic.uff.br

Aula de hoje

- Estudaremos outras duas estruturas de encapsulamento da Orientação a Objetos
  - Classes
  - Pacotes

Paradigma procedimental versus OO

- O **paradigma procedimental** organiza o programa em termos de **algoritmos**
- O **paradigma OO** organiza o programa em termos de **objetos**



~~Algoritmos~~Objetos

- Podemos criar programa pensando em termos de **objetos ao invés de algoritmos**?
- O mundo é composto de objetos
  - Uma loja tem produtos, pedidos, estoque, etc.
  - Um restaurante tem mesas, garçons, comidas, bebidas, etc.
  - Uma universidade tem professores, alunos, disciplinas, etc.
  - Uma rodoviária tem ônibus, passageiros, bagagens, etc.
- E se **criarmos programas** basicamente **criando objetos** equivalentes ao mundo real, e fazendo com que esses **objetos se comuniquem**?

Objetos

- Definição
  - Um objeto é a **representação computacional de um elemento ou processo do mundo real**
  - Cada objeto possui suas **características** e seu **comportamento**

- Exemplos de Objetos

cadeira	*mesa*	*caneta*	*lápis*
carro	*piloto*	*venda*	*mercadoria*
cliente	*aula*	*programa*	*computador*
aluno	*avião*		

Características de Objetos

- Definição
  - Uma característica descreve uma propriedade de um objeto, ou seja, algum elemento que descreva o objeto.
  - Cada característica é chamada de **atributo** e funciona como uma **variável** pertencente ao objeto

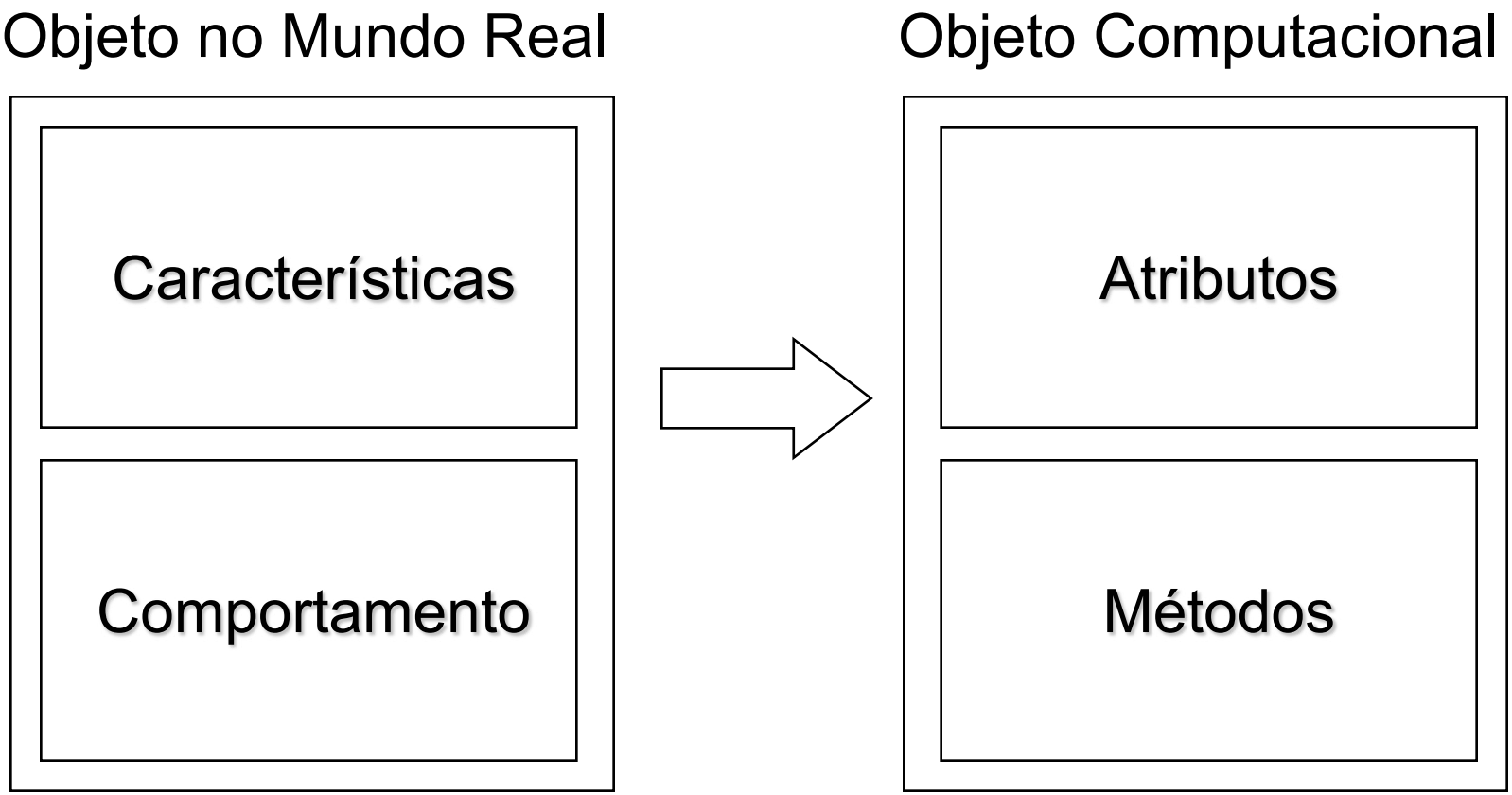
- Exemplo de características do objeto **carro**
  - Cor
  - Marca
  - Número de portas
  - Ano de fabricação
  - Tipo de combustível

Comportamento de Objetos

- Definição
  - Um comportamento representa uma ação ou resposta de um objeto a uma ação do mundo real
  - Cada comportamento é chamado de **método** e funciona como um **procedimento/função** pertencente ao objeto

- Exemplos de comportamento para o objeto **carro**
  - Acelerar
  - Frear
  - Virar para direita
  - Virar para esquerda

Mapeamento de Objetos



Paradigma Procedimental versus OO (exemplo: agenda)

Paradigma Procedimental

- Variáveis
  - Vetor de nomes
  - Vetor de endereços
  - Vetor de telefones
- Procedimentos
  - Listagem de todos os nomes
  - Listagem do endereço dado um nome
  - Listagem do telefone dado um nome
  - Adição de nome, endereço e telefone
  - Remoção de nome, endereço e telefone

Paradigma OO

- Objeto Agenda
  - Atributo
    - Vetor de Contatos
  - Métodos
    - Listagem de Contatos
    - Adição de um Contato
    - Remoção de um Contato
- Objeto Contato
  - Atributos
    - Nome
    - Endereço
    - Telefone
  - Métodos
    - Exibição de nome, endereço e telefone
    - Edição de nome, endereço e telefone

Paradigma OO
(Exemplo: total da compra)

Pedido: 12345
Cliente: João da Silva
Endereço: Rua dos Bobos, número zero

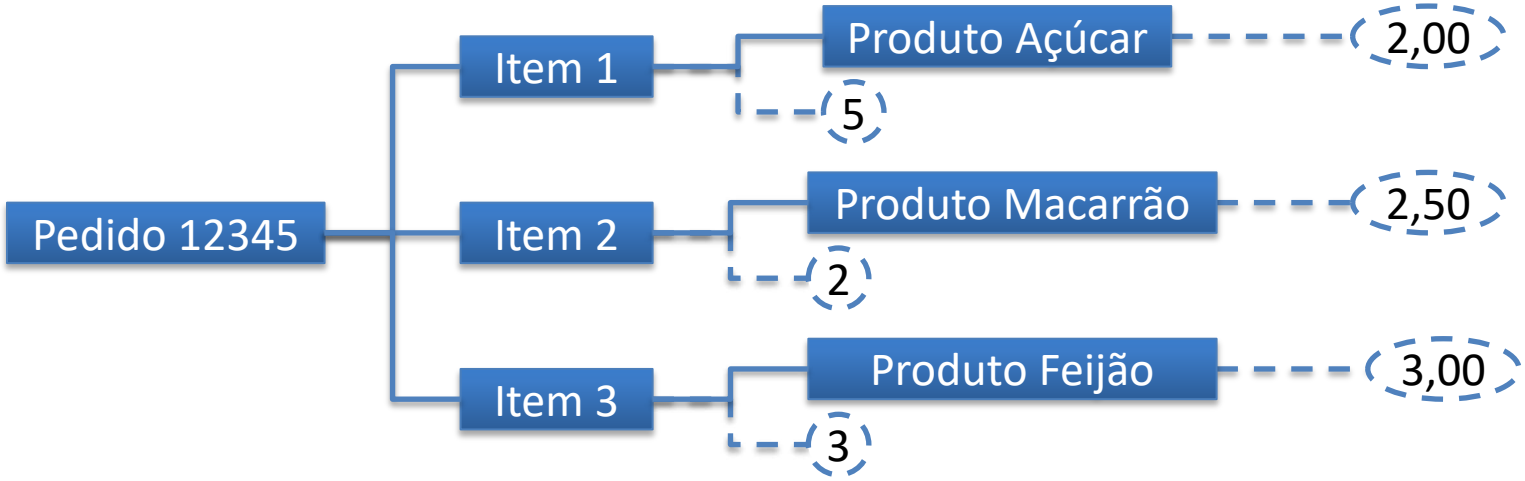
Item	Produto	Preço	Quantidade	Subtotal
1	Açúcar	R$ 2,00	5	R$ 10,00
2	Macarrão	R$ 2,50	2	R$ 5,00
3	Feijão	R$ 3,00	3	R$ 9,00
			TOTAL	R$ 24,00

Quais são os objetos participantes do cálculo do total da compra?

Paradigma OO
(Exemplo: total da compra)

Pedido: 12345
Cliente: João da Silva
Endereço: Rua dos Bobos, número zero

Item	Produto	Preço	Quantidade	Subtotal
1	Açúcar	R$ 2,00	5	R$ 10,00
2	Macarrão	R$ 2,50	2	R$ 5,00
3	Feijão	R$ 3,00	3	R$ 9,00
			TOTAL	R$ 24,00



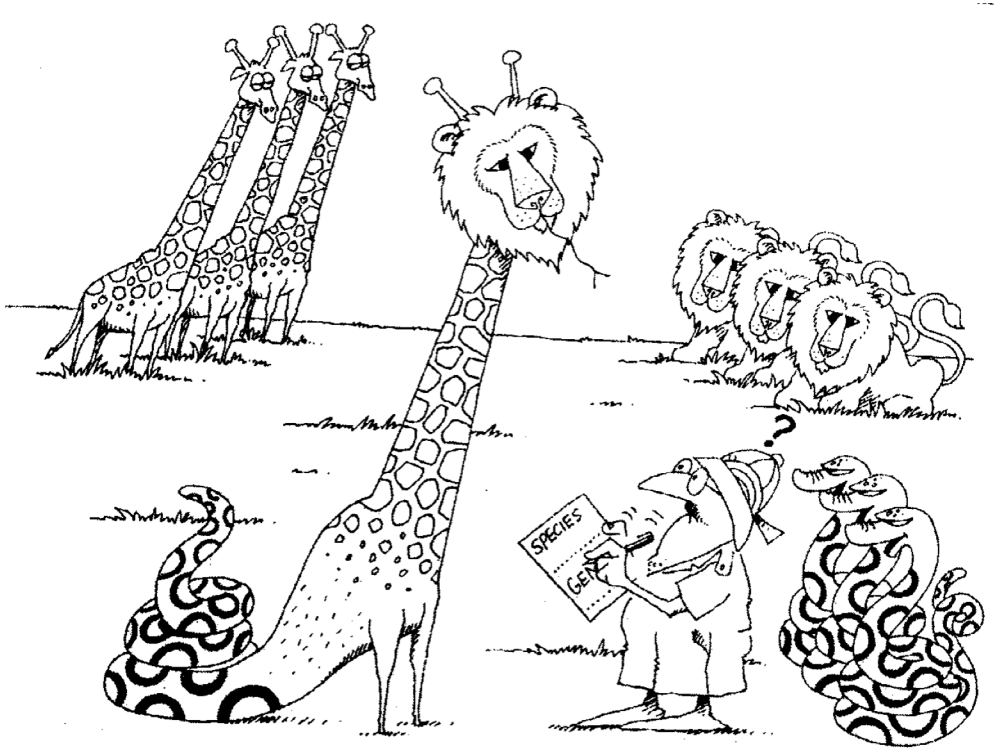
Paradigma OO
(Exemplo: total da compra)

- Como obter o total da compra?
  1. O objeto **Caixa** pediria ao objeto **Pedido** seu valor total
  2. O objeto **Pedido**, por sua vez, percorreria todos os seus objetos **Item** perguntando o seu valor subtotal e somaria esses valores para responder ao objeto **Caixa**
  3. Cada objeto **Item** perguntaria ao objeto **Produto** o seu preço e multiplicaria esse preço pela quantidade que está sendo comprada, para responder ao objeto **Pedido**

CLASSES

Classes versus Objetos

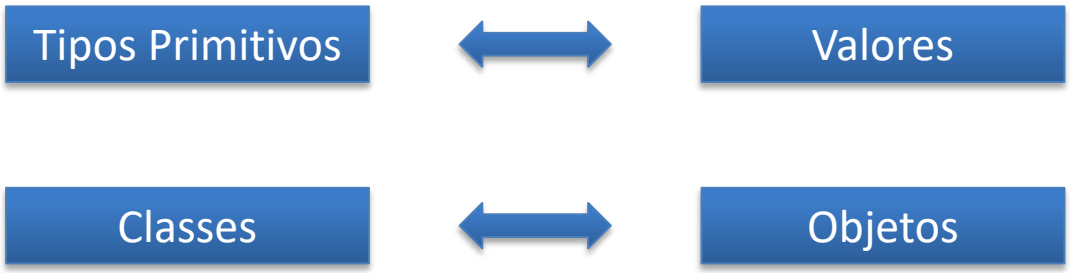
- A **Classe** é o **tipo do Objeto**



Fonte: livro “Object-Oriented Analysis and Design with Applications”

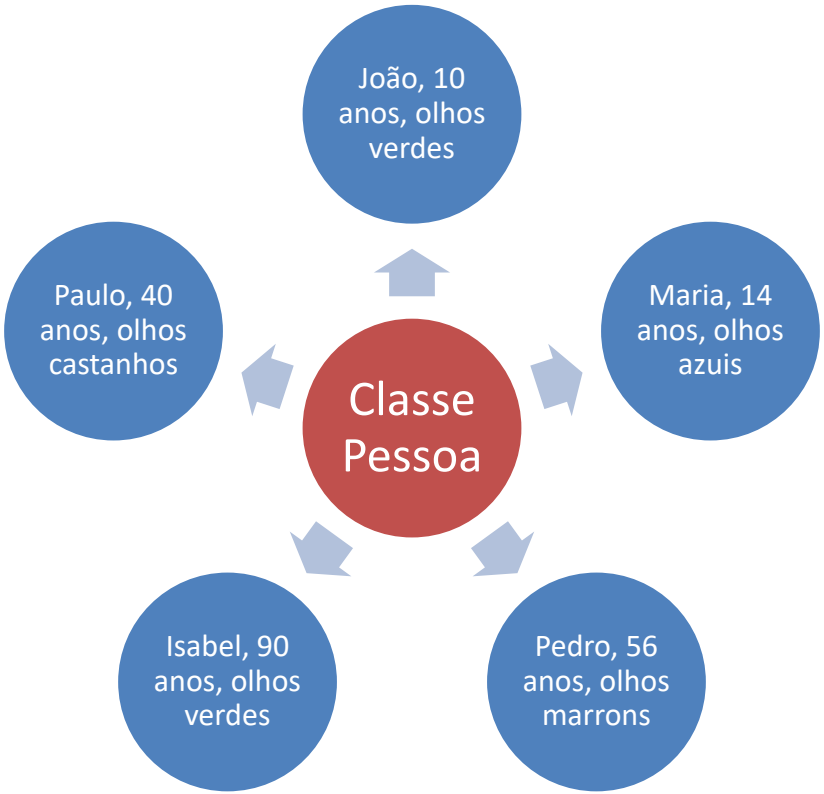
Classes versus Objetos

- Valores têm tipos primitivos
  - 123 é um valor inteiro
  - True é um valor booleano
  - 12,3 é um valor real
- Objetos pertencem a classes
  - João, Pedro e Paulo são da classe Pessoa
  - Fusca e Ferrari são da classe Carro
  - Flamengo e Fluminense são da classe Time



Classes versus Objetos

- Uma **classe é uma fôrma**, capaz de **produzir objetos**
- Os **programadores criam classes**, as **classes instanciam objetos**



Classes

- A classe descreve as características e comportamento de um conjunto de objetos
  - Em Java, **cada objeto pertence a uma única classe**
  - O objeto possuirá os atributos e métodos definidos na classe
  - O objeto é chamado de instância de sua classe
  - A classe é o bloco básico para a construção de programas OO

Exemplo de Classe

```
public class Carro {
  private int velocidade;

  public void acelera() {
    velocidade++;
  }

  public void freia() {
    velocidade--;
  }
}
```

Atributos (características) são variáveis globais acessíveis por todos os métodos da classe

Métodos (comportamentos)

Criação de objetos

- A classe é responsável pela criação de seus objetos via método construtor
  - Mesmo nome da classe
  - Sem tipo de retorno

```
public Carro(int velocidadeInicial) {
  velocidade = velocidadeInicial;
}
```

Criação de objetos

- Objetos devem ser instanciados antes de utilizados
  - O comando ***new*** instancia um objeto, chamando o seu construtor
- Exemplo:

```
Carro fusca = new Carro(10);
Carro bmw = new Carro(15);
fusca.freia();
bmw.acelera();
fusca = bmw;
```

Qual a velocidade de cada carro em cada momento?

O que acontece aqui?

Criação de objetos

- Valor *null*:
  - Utilizado para representar um objeto não inicializado
  - Quando um método retorna um objeto, ele pode retornar *null* para indicar, por exemplo, que o objeto não foi encontrado
  - É possível atribuir *null* para descartar um objeto previamente instanciado
- Exemplo:

```
Carro fusca = new Carro(10);
fusca.acelera();
fusca = null;
```

PACOTES

Pacotes

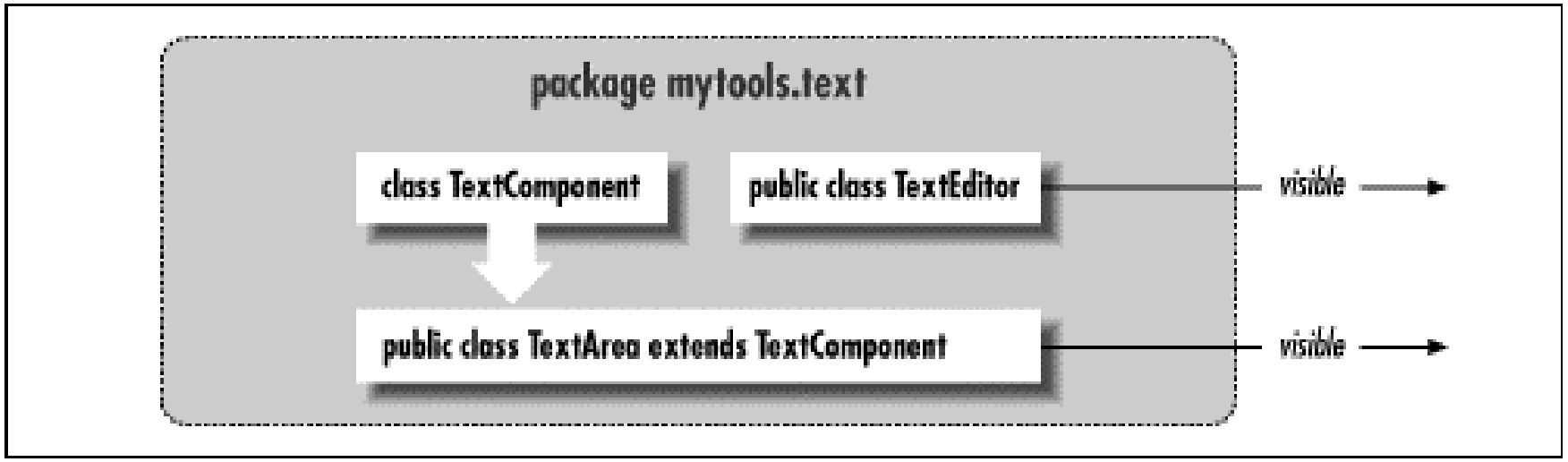
- Utilizados para agregar classes relacionadas
- O pacote de uma classe é indicado na primeira linha da classe
  - Declaração *package*
- Se uma classe não declara seu pacote, o interpretador assume que a classe pertence a um pacote *default*

```
package br.uff.ic;

public class Pessoa {
   ...
}
```

Pacotes

- Modificadores permitem que determinadas classes sejam visíveis apenas para outras classes do mesmo pacote



Pacotes

- Sempre que for usar uma classe de outro pacote, é necessário importar
- A importação se realiza através da palavra-chave *import*, seguida do nome da classe desejada
  - As importações são apresentadas antes da declaração da classe mas depois da declaração do pacote
  - A importação de um pacote **não importa** os subpacotes recursivamente

```
package br.uff.ic.prog1;

import java.util.Scanner;

public class Fisica {
   ...
}
```

Regra de ouro para classes e pacotes

- Classes devem ser mapeadas em arquivos com o mesmo nome
  - Classe **Pessoa**
  - Arquivo **Pessoa.java**
- Pacotes devem ser mapeados em diretórios
  - Pacote **br.uff.ic**
  - Diretório **br\uff\ic**
- Se o nome completo da classe é **br.uff.ic.Pessoa**
  - Deve haver **br\uff\ic\Pessoa.java**

ATRIBUTOS E MÉTODOS

Modificadores

- Atributos e métodos podem ter diferentes visibilidades e escopos
  - Estamos até agora usando somente os modificadores *public static*
  - O que significam esses modificadores?
  - Quais outros modificadores existem?

Modificador de visibilidade

- Indica quem pode acessar o atributo ou método:
  - O modificador ***private*** indica que o atributo ou método pode ser chamado apenas por outros métodos da própria classe
  - A ausência de modificador é conhecida como ***package***, e indica que o atributo ou método pode ser chamado somente por classes do mesmo pacote
  - O modificador ***protected*** indica que o atributo ou método pode ser chamado somente por classes do mesmo pacote ou subclasses;
  - O modificador ***public*** indica que o atributo ou método pode ser chamado por qualquer outra classe

Modificador de escopo

- Indica a quem pertence o método (ou atributo)
  - Ao objeto (instância)
  - À classe como um todo (compartilhado para todas as instâncias)
- Atributos ou métodos estáticos (*static*) pertencem à classe como um todo
  - Podem ser chamados diretamente na classe, sem a necessidade de instanciar objetos
  - Métodos estáticos só podem manipular atributos estáticos
- A ausência do modificador *static* indica que o atributo ou método é de instância

Agora já sabemos ler!!!

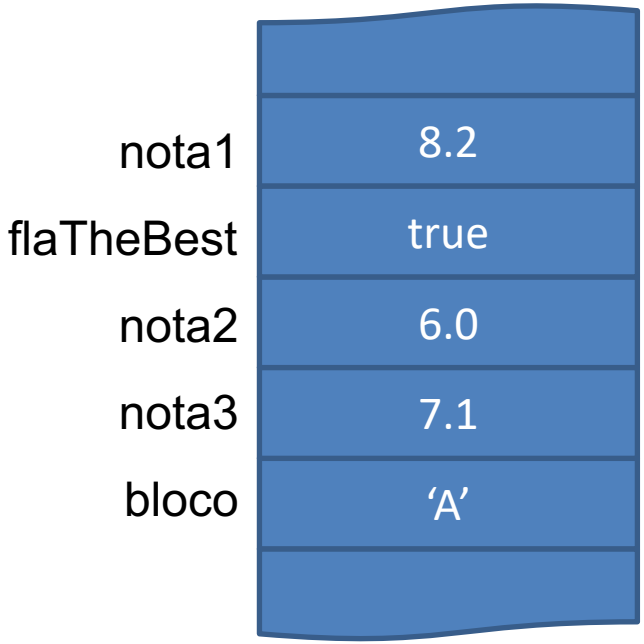
public static void main(String[] args)

Passagem por valor vs. passagem por referência

- Algumas linguagens permitem passagem de argumentos por referência
  - Não é o caso de Java, que sempre faz passagem por valor
- Diferenças
  - Passagem por valor = cópia dos valores para outra posição de memória
  - Passagem por referência = reuso da posição de memória

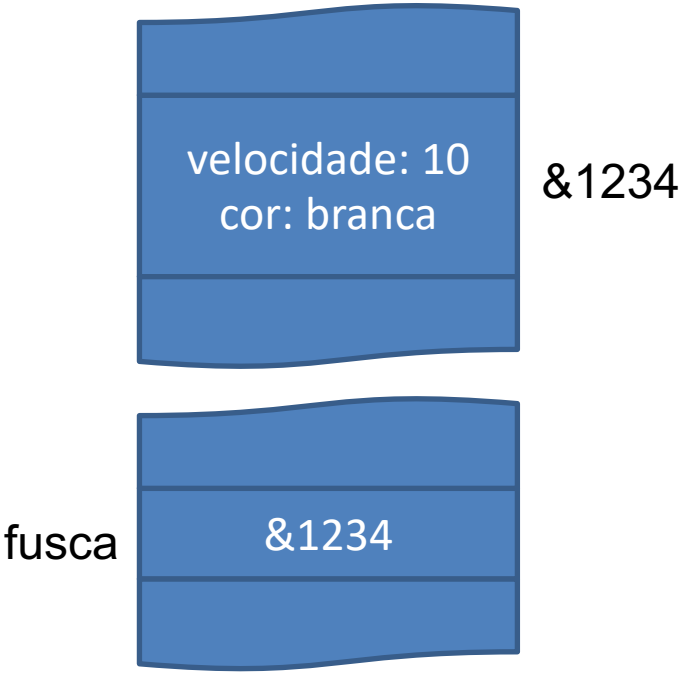
Passagem por valor vs. passagem por referência

- Variáveis que contêm tipos primitivos (byte, short, int, long, float, double, char, boolean) sempre ocupam diretamente uma posição na memória



Passagem por valor vs. passagem por referência

- Variáveis que contêm objetos na verdade guardam a posição de memória dos objetos



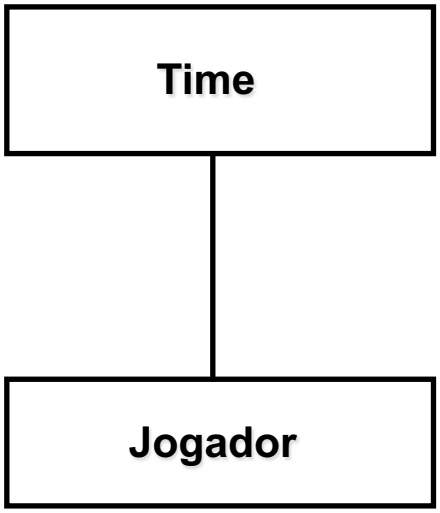
Passagem por valor vs. passagem por referência

- Quando é passado um objeto por valor a referência é copiada
  - Mudanças nos atributos dos objetos são vistas de fora
  - Instanciações de novos objetos nas variáveis não são vistas de fora

Classes são tipos!

- Classes podem ser utilizadas como
  - Tipos dos atributos de uma outra classe
  - Parâmetros de um método
  - Tipo de retorno de um método

Classes são tipos!




```
class Time
{
        private Jogador[] jogadores;
        …
        public Jogador getJogador(int numero) {
        ...
        public void escala(Jogador jogador) {
        ...
}


class Jogador
{
        private Time time;
        …
}
```

Exercício 1

- Identifique as classes para a seguinte especificação:

"O supermercado vende diferentes tipos de produtos. Cada produto tem um preço e uma quantidade em estoque. Um pedido de um cliente é composto de itens, onde cada item especifica o produto que o cliente deseja e a respectiva quantidade. Esse pedido pode ser pago em dinheiro, cheque ou cartão."

Exercício 2

- Refaça o exercício da aula de variáveis compostas usando somente um vetor (e classe) no lugar de dois vetores (um de nomes e outro de idades)
  - Lembrete: o exercício era para listar 10 pessoas ordenado por nome e depois por idade

Exercício 3

- Refaça o exercício da aula de subprogramação usando uma classe para representar a calculadora

Instituto de computação uff

Introdução a Orientação a Objetos

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br