Programação Orientada a Objetos – Aula 04

IFPE – Campus Igarassu
2016.1

Ranieri Valença
21/07/2016

Tópicos de hoje

- Revisão
- Métodos com parâmetros
  - Parâmetros X argumentos
  - Passagem por valor vs. Passagem por referência

Lembrando...

Dizemos que um objeto é uma

Instância

de uma classe

Lembrando...

Para instanciar um objeto em Java

(e em algumas outras linguagens – tipo PHP)

Usando o operador **new**

Instanciando objetos

Método **main** – o ponto de entrada para um programa em Java

```
class App {
    public static void main(String[] args) {
        Pessoa p;
        p = new Pessoa();
        p.idade = 30;
        p.nome = "Tom Riddle";
    }
}
```

Declarando variável "p" do tipo Pessoa

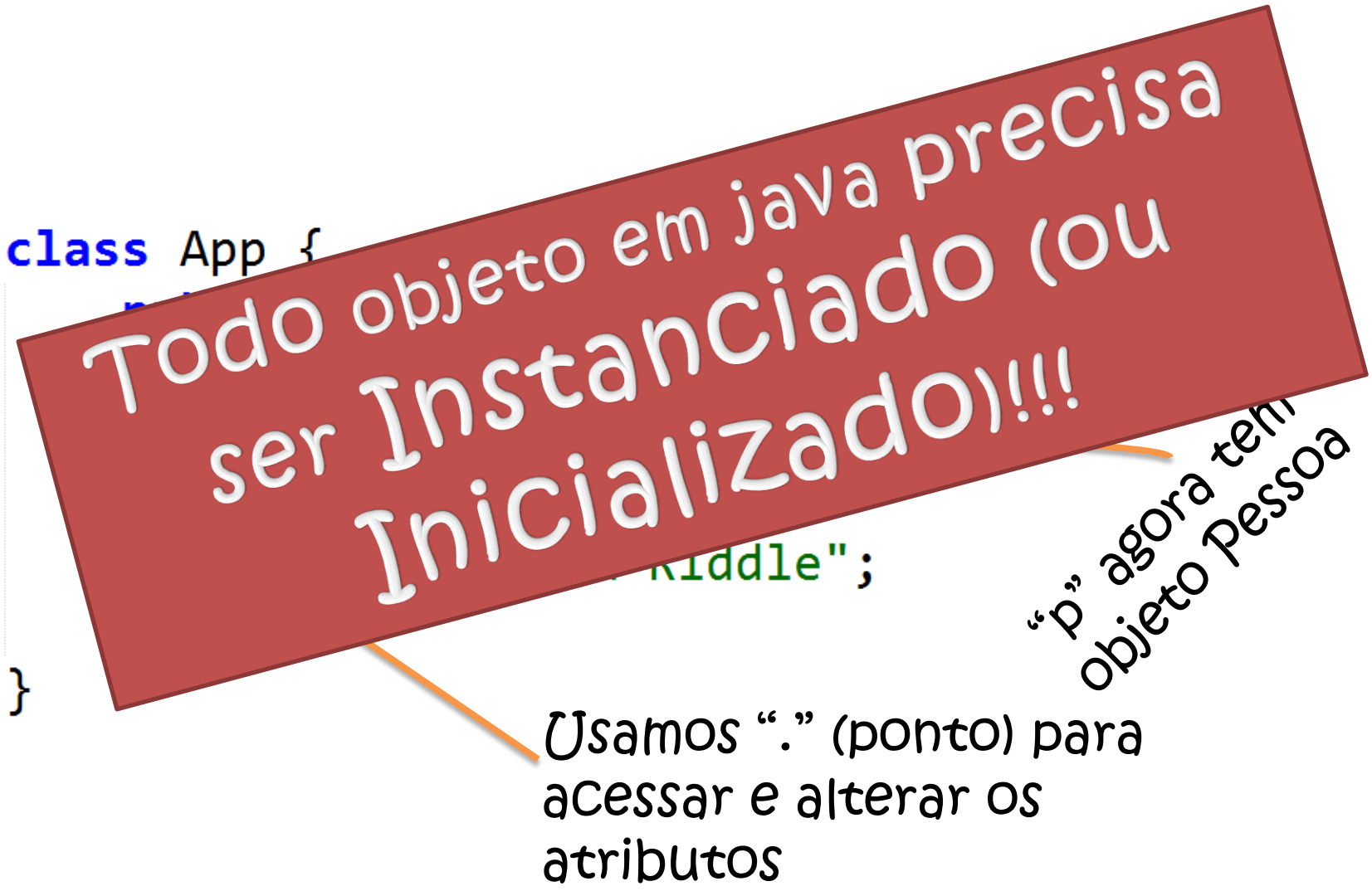
Instanciando objetos

```
class App {
    public static void main(String[] args) {
        Pessoa p;
        p = new Pessoa();
        p.idade = 30;
        p.nome = "Tom Riddle";
    }
}
```

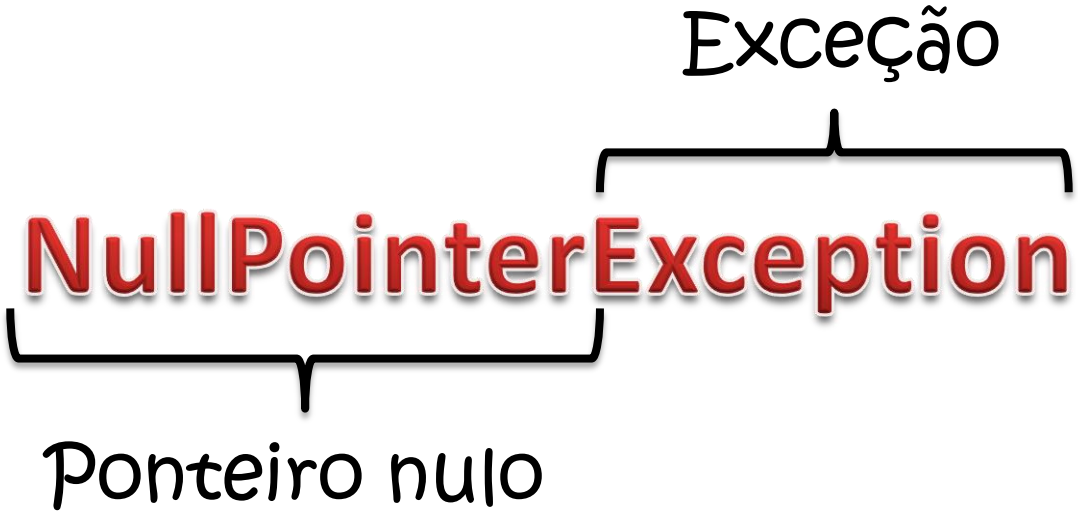
"p" agora tem um objeto Pessoa

Usamos "." (ponto) para acessar e alterar os atributos

Instanciando objetos



Quando tentamos acessar um objeto que não foi inicializado...



Quando tentamos acessar um objeto que não foi inicializado...

Tenha certeza que seus objetos estão inicializados!

Comportamentos dos objetos

No mundo real, os objetos têm

ações e **comportamentos**

Essas ações e comportamentos são comuns a todos os **objetos** similares – ou seja, comuns à **classe**

Traduzindo isso para nosso mundo...

Comportamentos de objetos no mundo real são traduzidos em **Métodos** no paradigma orientado a objetos

Lembrando o que é um método

São **subprogramas** que contém instruções de execução

A ideia é que cada método execute **uma única tarefa**

Esqueleto básico de um método (por enquanto)

```
void imprimeNome() {
    System.out.println("O indivíduo se chama "
        + nome);
}
```

Esqueleto básico de um método (por enquanto)

Tipo de retorno
void significa "sem retorno"

Parênteses

```
void imprimeNome() {
    System.out.println("O indivíduo se chama "
        + nome);
}
```

Nome do método

O corpo do método fica entre chaves

Onde ficam os métodos?? Dentro da classe!

```
class Pessoa {
    int idade;
    float pesoEmQuilogramas;
    float alturaEmMetros;
    char sexo;
    String nome;
    String sobrenome;
    String[] telefone;

    void imprimeNome() {
        System.out.println("O indivíduo se chama "
            + nome);
    }

}
```

Atributos

Métodos

Invocando métodos **em Java**

```
class App {
    public static void main(String[] args) {
        Pessoa p;
        p = new Pessoa();
        p.idade = 30;
        p.nome = "Tom Riddle";

        p.imprimeNomeEIdade();
    }
}
```

Operador "." (ponto)

Invocando métodos **em Java**

```
class App {
    public static
        P
                }
                me = "Tom Riddle";

        p.imprimeNomeEIdade();
    }
}
```

Métodos são invocados a partir de **objetos** (instâncias)

Operador "." (ponto)

Prática 1 (aula passada)

1. Crie um método chamado “**autonomia**” na classe “Veiculo”
2. Na hora de imprimir a autonomia, invoque o método “autonomia()”

Métodos que retornam algo

Algumas ações de certos objetos têm um retorno

"Por exemplo, se você pergunta a idade de uma pessoa, ela **retorna** a idade para você, que armazena aquela informação"

Métodos que retornam algo

Utilizamos a palavra especial “return” para que um método retorne algo

Obviamente quando um método retorna algum valor, seu **tipo de retorno** precisa ser especificado

Métodos que retornam algo

```
int alturaEmCentimetros() {
    return alturaEmMetros * 100;
}
```

```
int altura = p.alturaEmCentimetros();
System.out.println("A criatura tem " + altura + "cm");
```

Prática 2

1. Faça o método “autonomia” da classe “Veiculo” **retornar** a autonomia do veículo ao invés de imprimi-la
2. Altere a chamada ao método e imprima a autonomia dentro do método **main** da classe “App”

Métodos parametrizados

Esses métodos que fizemos até agora apenas fazem operações com os atributos do objeto

E se quisermos calcular quantos litros seriam necessários para uma certa quantidade de quilômetros?

Métodos parametrizados

A quantidade de combustível **depende** de quantos quilômetros eu quero saber

Métodos parametrizados

```
String nomeComApelido(String apelido) {
    String s = nome + " (" + apelido + ")";
    return s;
}
```

Métodos parametrizados

```
// ...
int altura = p.alturaEmCentimetros();
System.out.println("A criatura tem " + altura + "cm");
System.out.println("A criatura se chama " +
    p.nomeComApelido("Serumaninho do mal"));
```

Métodos parametrizados

Um método também pode receber múltiplos parâmetros:

Métodos parametrizados

```
class Numero {
    boolean ehDivisivel(int a, int b) {
        if (a % b == 0) {
            return true;
        }
        return false;
    }
}
```

Vírgula para separar os parâmetros

Métodos parametrizados

```
class App {
    public static void main(String[] args) {
        Numero n = new Numero();
        System.out.println(n.ehDivisivel(7, 3));
        System.out.println(n.ehDivisivel(28, 4));
    }
}
```

Parâmetro vs Argumento – A batalha épica

Em orientação a objetos, há uma **sutileza** em relação aos conceitos de **parâmetro** e **argumento**

Parâmetro vs Argumento – A batalha épica

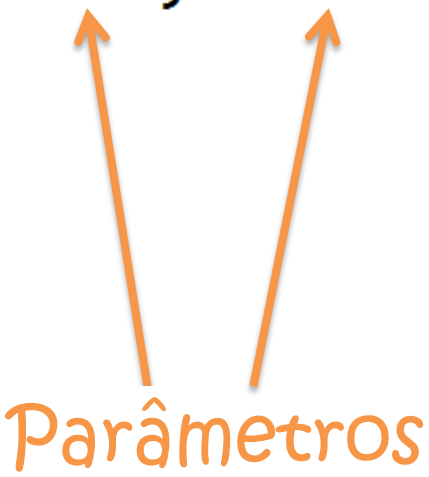
Argumentos são os valores passados para um método durante sua **invocação**

Parâmetro vs Argumento – A batalha épica

Parâmetros são as variáveis declaradas no método, que servem apenas no **escopo** daquele método e que recebem os valores dos argumentos

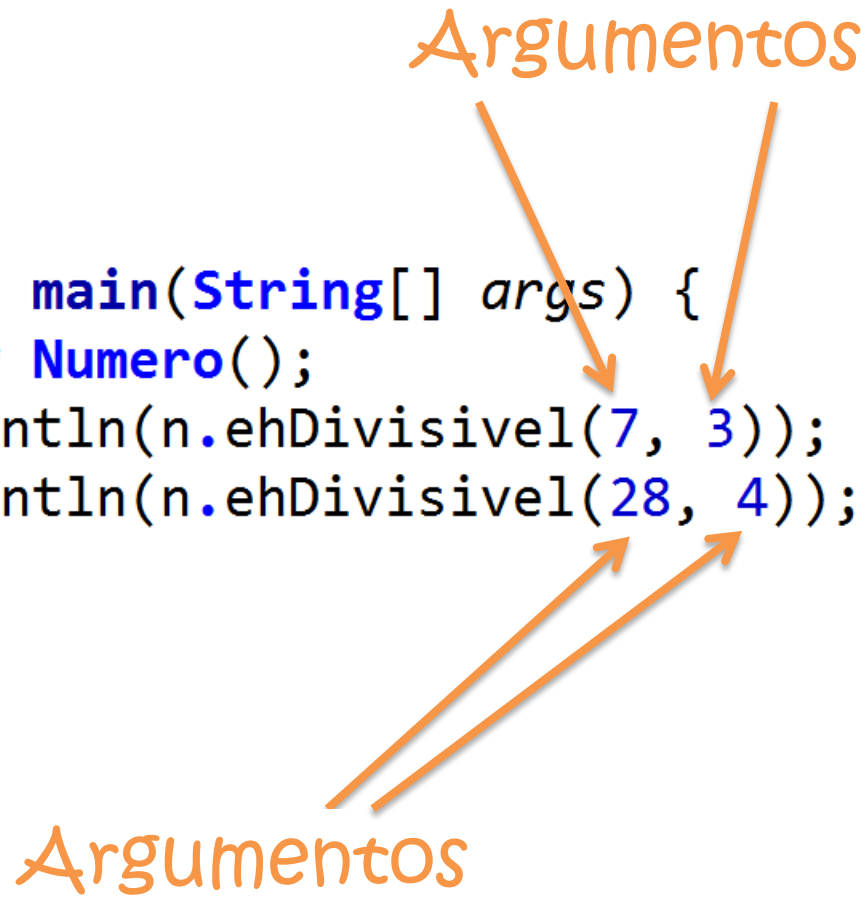
Parâmetro vs Argumento – A batalha épica

```
class Numero {
    boolean ehDivisivel(int a, int b) {
        if (a % b == 0) {
            return true;
        }
        return false;
    }
}
```



Parâmetro vs Argumento – A batalha épica

```
class App {
    public static void main(String[] args) {
        Numero n = new Numero();
        System.out.println(n.ehDivisivel(7, 3));
        System.out.println(n.ehDivisivel(28, 4));
    }
}
```



Prática 3

1. Crie um método chamado "**combustivelNecessario**" na classe "Veiculo", que recebe um argumento que corresponde ao numero de km para o qual se deseja calcular a quantidade de combustível necessária
2. Crie um método chamado "**dinheiroNecessario**" na classe "Veiculo" que recebe como argumentos a quantidade de km e o preço do litro de combustível

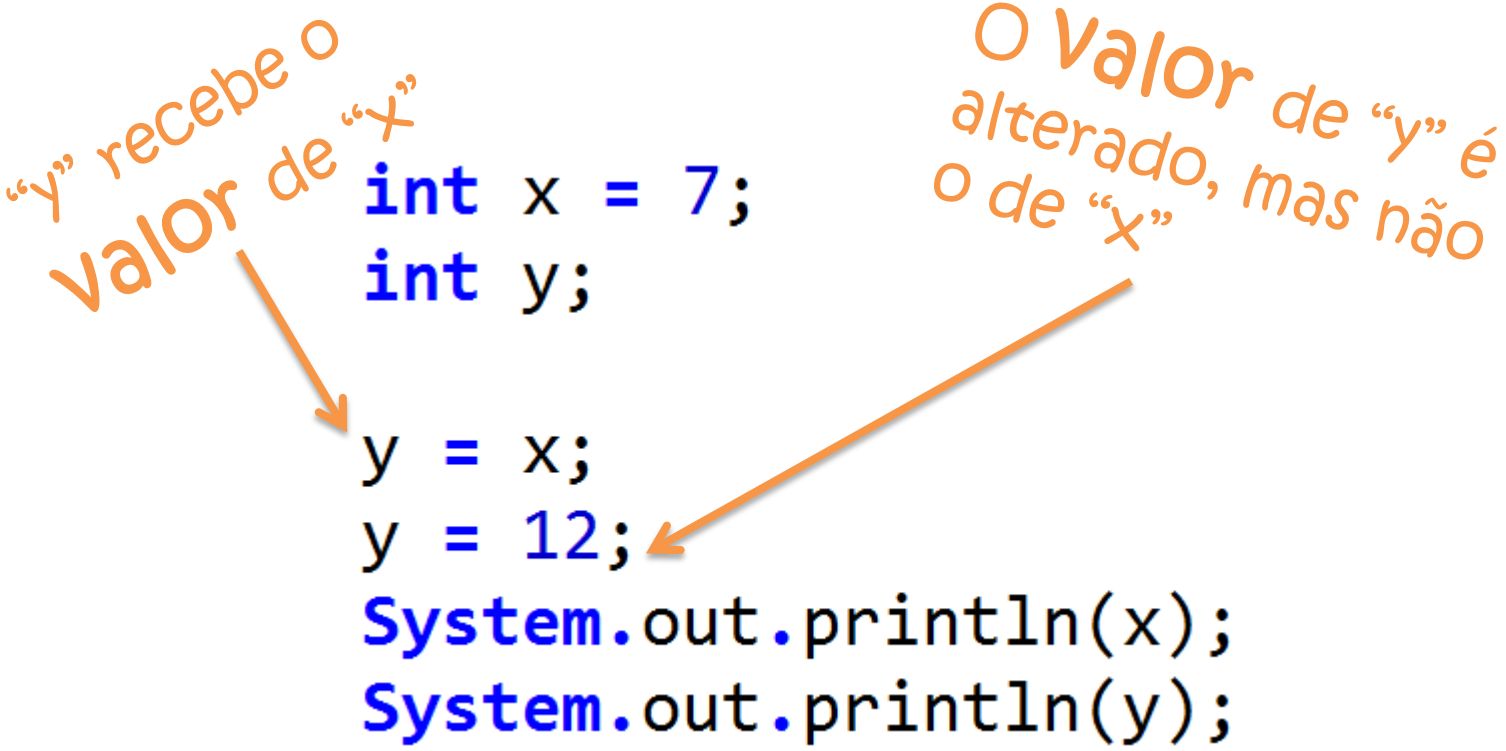
Referências e valores

As linguagens orientadas a objetos possuem o conceito de **referência**

Uma atribuição não necessariamente cria uma **cópia**

~~Como é pae?~~

Referências e valores



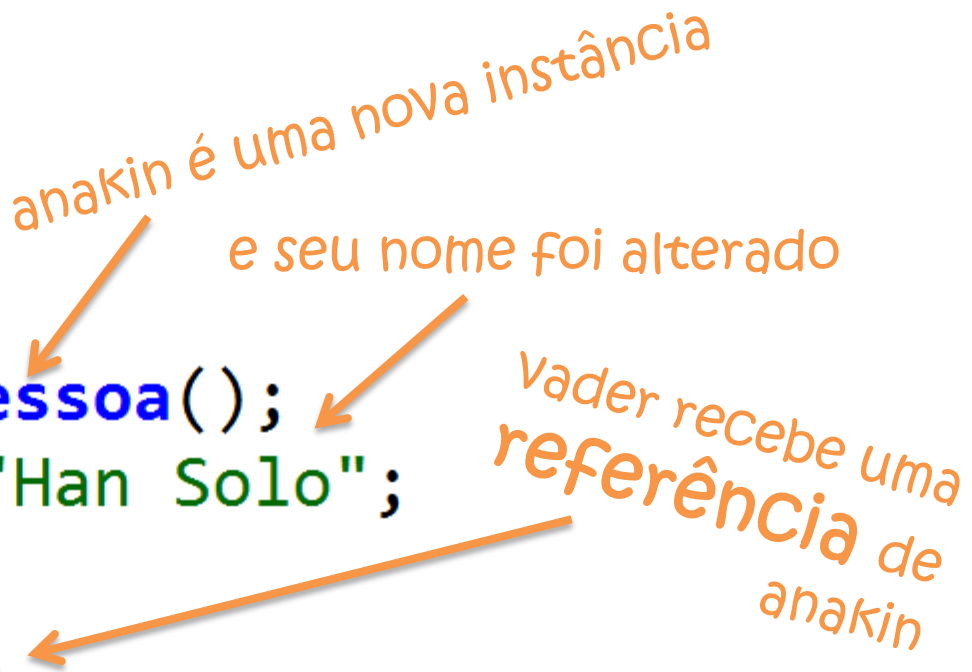
Referências e valores

```
Pessoa anakin;
Pessoa vader;

anakin = new Pessoa();
anakin.nome = "Han Solo";

vader = anakin;
vader.nome = "Lord Vader";

System.out.println(anakin.nome);
System.out.println(vader.nome);
```



Prática 4

1. Declare duas variáveis do tipo “Veiculo”
   - **`veiculo1`** e **`veiculo2`**
2. Inicialize `veiculo1`
3. Faça `veiculo2 = veiculo1;`
4. Altere os valores dos atributos de `veiculo1`
5. Imprima (`System.out.println`) os atributos de `veiculo2`

O que acontece?