

Programação Orientada a Objetos

Prof. Dr. Anderson V. de Araujo



Módulo 1 - Fundamentos de Orientação a Objetos

Unidade 3 - Classes e Objetos



Classes


- São modelos (*templates*) usados para criar objetos, definindo:
 - O comportamento de seus objetos através de métodos;
 - As informações destes objetos através de atributos.
- É uma entidade lógica que abstrai um conjunto de objetos com características similares

Classes (2)

- Auxiliam no Reuso
 - Exemplos:
 - Ao desenhar uma planta de uma casa, um arquiteto pode usar a mesma planta para construir várias casas iguais;
 - Um formulário impressão/copiado e distribuído para várias pessoas preencherem, o formulário é um só (molde) mas as cópias preenchidas são diferentes (objetos)
 - Reusar classes existentes, poupa tempo e esforço.

Responsabilidade de uma Classe

- Ao construir uma classe, temos que definir quais são as suas responsabilidades
 - O que ela faz? Quais são suas características?
 - Com quais classes ela se relaciona (colabora)?
 - Ela não pode fazer algo fora das suas responsabilidades
- Exemplo:
 - Classe ContaCorrente
 - Apresentar o saldo
 - Possibilitar um saque
 - Possibilitar um depósito

- 
- Criar nova conta
 - Armazenar todas as contas do banco
 - Gerar relatório de valor total armazenado no banco

Membros Internos

- São partes que compõem a classe;
- **Atributos** (ou Propriedades):
 - São variáveis declaradas dentro da classe, mas fora de qualquer método.
- **Métodos** (ou comportamentos):
 - São métodos internos da classe que correspondem a comportamentos associados à classe.

Objetos

- São criados a partir de classes:
 - São **instâncias** de uma classe;
 - Vários objetos podem ser instanciados a partir de uma classe.
- A classe é o molde e o objeto é o produto moldado pela classe. Exemplos:
 - A classe pode ser imaginada como a planta de uma casa e a casa construída (física) é o objeto criado;
 - Formulário de inscrição é o molde (classe) e as cópias preenchidas são os objetos.
- Pode representar um objeto do mundo real ou uma abstração:
 - Ex.: Casa, Conta bancária

Exemplo 1

- Classe Smartphone:
 - Atributos (propriedades):
 - Número de processadores, memória, tamanho da tela, marca...
 - Comportamentos (métodos):
 - Ligar, mandar mensagem, conectar na internet, ...
- Objetos (instâncias):
 - iPhone 6, Samsung Galaxy 5, Sony Xperia Z3, LG Nexus 5, ...


```
package exemplo;
```

```
class Smartphone{
```

```
    int numeroDeProcs;  
    double memoriaRam;  
    double tamanhoTela;  
    String marca;
```

Atributos
(Propriedades)

```
    void mandarMsg(String numero, String msg){  
        //...  
    }
```

```
    void ligar(String numero){  
        //...  
    }
```

Objeto Iphone 6

- numeroDeProcs = 2
- memoriaRam = 1
- tamanhoTela = 4.7
- marca = Apple

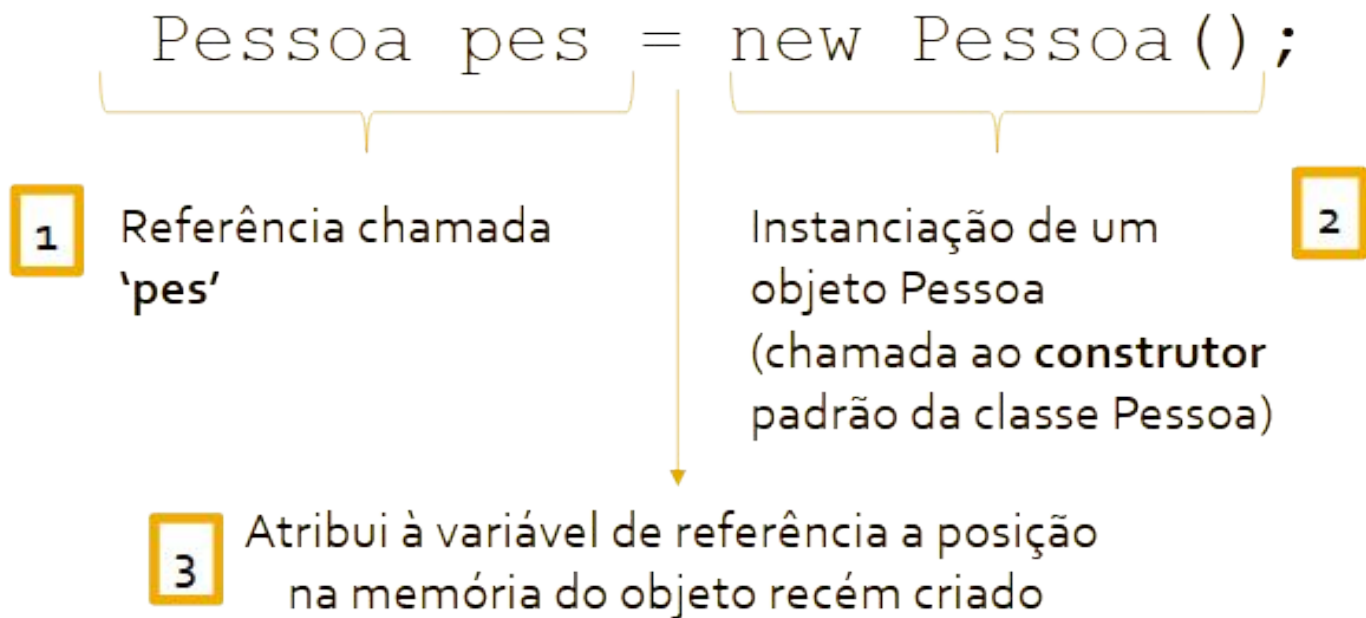
Objeto Nexus 5

- numeroDeProcs = 4
- memoriaRam = 2
- tamanhoTela = 4.95
- marca = LG

Ciclo de Vida de um Objeto

- Um objeto é criado usando a palavra reservada `new` (instanciação):
 - **`Pessoa pes = new Pessoa();`**
 - A JVM busca o arquivo `Pessoa.class` no disco e o carrega para a memória;
 - A JVM cria um espaço na memória do tamanho do objeto e aponta uma referência (chamada de **`pes`**) para o objeto criado;
- Não é possível destruir objetos diretamente da memória em Java:
 - Eles são removidos automaticamente da memória (pelo *Garbage Collector*):
 - Como por exemplo, quando nenhuma referência apontar para o objeto.

Resumindo...



```
class Pessoa {  
    String nome;  
    int idade;  
}
```

Memória



Operador . (Ponto)

- Possibilita o acesso a membros internos de uma classe
 - Mas nem sempre isso é possível, depende do modificador de acesso* de cada membro
- Exemplo:

```
Pessoa p = new Pessoa();  
p.nome = "João";  
p.idade = 35;  
p.falar();
```

*mais tarde...

Memória



Variáveis

- Variáveis servem para armazenar valores;
- Existem dois tipos de variáveis:
 - Tipos Primitivos:
 - `char`, `boolean`, `byte`, `short`, `int`, `long`,
`double` ou `float`.
 - Variáveis de Referência:
 - Usadas para se referir (apontar para) e acessar um objeto.

Atributos

- São variáveis declaradas dentro da classe, mas fora de qualquer método;
- Possuem valores padrão de inicialização;
- Pode ser da Instância ou da Classe:
- Atributo da Instância (do Objeto):
 - Não possui o modificador static;
 - Um valor para cada objeto.
- Atributo da Classe:
 - Possui o modificador static;
 - Um valor único para todos os objetos/instâncias.

Atributos - Exemplo

- Classe **Pessoa**, com os atributos **nome** e **idade**;

```
class Pessoa {  
    String nome;  
    int idade;  
    public static void main(String[] args) {  
        Pessoa p = new Pessoa();  
        p.nome = "João";  
        p.idade = 35;  
        System.out.println(p.nome + p.idade);  
    }  
}
```

Métodos

- São blocos de código separados que pertencem a um programa e tem por finalidade realizar uma tarefa específica
- Geralmente correspondem à uma ação:
 - somar, correr, imprimir, ler Arquivo, ...
- Podem retornar um valor ao código chamador;
- O tipo do valor de retorno é definido antes do nome do método:
 - A palavra-chave **void** indica a ausência de retorno. Serve para definir um método que não retorna dado algum.

Métodos (2)

- Evitar reescrever código para uma mesma ação toda vez que se desejar executá-la;
- Podem ser declaradas em qualquer posição no código:
 - Diferentemente de outras linguagens que obrigam que a declaração ocorra antes da sua chamada/utilização.
- Sinônimos:
 - Procedimento, Função, Sub-rotina, Subprograma, ...

Declaração de um Método

- Definir as características, tipo de retorno, nome e parâmetros;
- Sintaxe geral:

```
[características] tipo_retorno nome(parâmetros) {  
    // código do método  
}
```

- Exemplo:

Assinatura do método

```
public static void main(String[] args) {  
}
```

Retorno de um Método

- O retorno dentro do método é realizado através da palavra reservada **return**:
 - O **return** pára a execução do método corrente e devolve o valor de retorno ao método chamador.

```
public boolean ehPar(int a) {  
    return a % 2 == 0;  
}  
public String imprimeAltura(double altura) {  
    return "Minha altura é: " + altura;  
}
```

Parâmetros

- É a sequência de definições de variáveis separadas por vírgulas entre os parênteses na assinatura do método;
- O código interno ao método pode usar as variáveis com valores passados no código chamador;
- Um método pode ter nenhum, um ou mais parâmetros.

Exemplo - Classe Pessoa

```
class Pessoa {  
    String nome;  
    int idade;  
  
    public void imprimirData(int dia) {  
        System.out.println("Eu tenho " + idade + " anos.  
        Nasci no dia: " + dia);  
    }  
  
    public void imprimirNome() {  
        System.out.println(nome);  
    }  
}
```

Exemplo (2) - método main

```
public static void main(String[] args) {  
    Pessoa p = new Pessoa();  
    p.nome = "João";  
    p.idade = 35;  
    p.imprimirNome();  
    p.imprimirData(25);  
}  
}
```

Licenciamento



Respeitadas as formas de citação formal de autores de acordo com as normas da ABNT NBR 6023 (2018), a não ser que esteja indicado de outra forma, todo material desta apresentação está licenciado sob uma [Licença Creative Commons - Atribuição 4.0 Internacional](https://creativecommons.org/licenses/by/4.0/).