

# Programação Orientada a Objetos

***Classes, objetos, atributos, métodos e referências***

*Baseados nos slides do prof. Régis Pires Magalhães*

---

**Franciéric Alves – [francieric \[at\] gmail.com](mailto:francieric@gmail.com)**

# Definição de Objetos

- Qualquer entidade (coisa) concreta ou abstrata do mundo real.
  - Transporte, Carro, Carro de Passeio, Produto, Lâmpada Incandescente, Lâmpada Fluorescente, Motor, Conta, Conta Poupança, Conta Corrente, Cartão de Crédito, Processo
- Objetos são formados por suas características (atributos) e seus comportamentos (métodos)

| Objeto   | Característica          | Comportamento           |
|----------|-------------------------|-------------------------|
| Carro    | Velocidade, cor, modelo | Ligar, Acelerar, Frear, |
| Conta    | Saldo, titular          | Creditar, debitar       |
| Cachorro | Raça, cor, temperamento | Latir, correr, morder   |

# Classes

- Modelo, forma ou planta de objetos
- Uma classe é uma descrição do que um ob
- Uma classe está para um objeto, assim como:
  - Uma receita está para uma torta
  - Uma planta está para uma casa
- Diz-se que uma objeto é uma classe em memória (instanciada)

| Classe Carro                       |                 | Objeto Carro A | Objeto Carro B |
|------------------------------------|-----------------|----------------|----------------|
| <i>Atributos<br/>de<br/>Objeto</i> | Número da placa | ABC 111        | XYZ 123        |
|                                    | Cor             | Azul           | Vermelha       |
|                                    | Fabricante      | Mitsubishi     | Toyota         |
|                                    | Velocidade      | 50 km/h        | 100 km/h       |
| <i>Métodos<br/>de<br/>Objeto</i>   | Método Acelerar |                |                |
|                                    | Método Girar    |                |                |
|                                    | Método Frear    |                |                |

## Segundo Alan Key

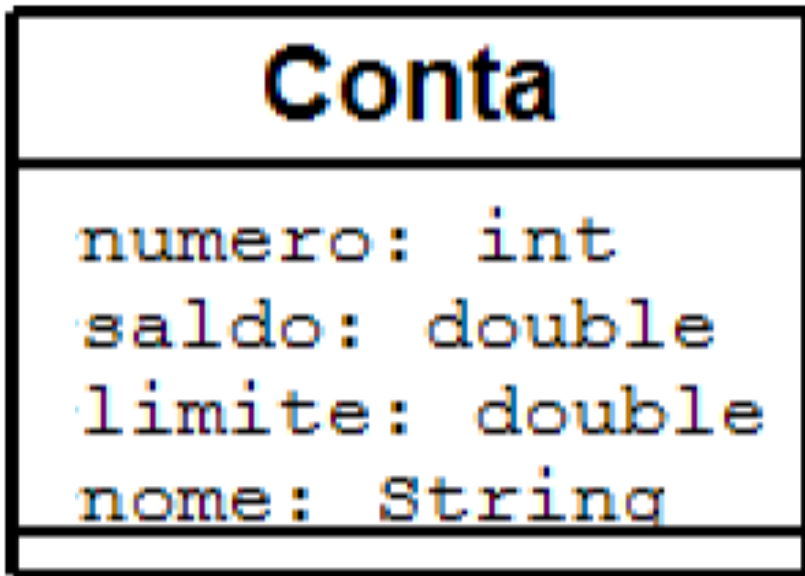
- Qualquer coisa é um objeto;
- Objetos realizam tarefas através da requisição de serviços a outros objetos;
- Cada objeto pertence a uma determinada classe.
- Uma classe agrupa objetos similares;
- A classe é um repositório para comportamento associado ao objeto;
- Classes são organizadas em hierarquias.

# Classificação

- Enquadramento dos objetos em categorias (classes), conforme suas características e suas funções.
  - Um mamífero que vôa é morcego, um que late é cachorro;
  - Um barco e um automóvel são meios de transporte, tendo em vista sua funcionalidade;
  - Um local lógico onde é feito um depósito em um banco é uma conta
  - Um recolhimento compulsório feito pelo governo é um tributo

# Exemplo

- Classe em UML e Classe em Java



```
class Conta {  
    int numero;  
    String nome;  
    double saldo;  
    double limite;  
    // ..  
}
```

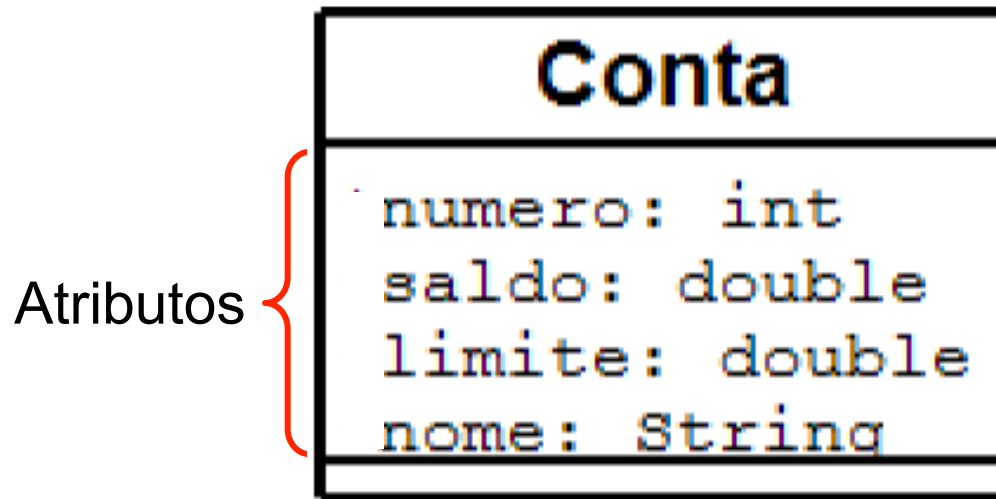
# Criação de objetos

- Instanciação
  - Para instanciar um objeto a palavra reservada **new** é utilizada;
    - Retorna uma referência para o objeto criado.
  - Em Java, todos os objetos são criados dinamicamente.

```
Conta conta1 = new Conta();
```

# Elementos de uma classe

- Atributos:
  - São características de cada objeto de uma determinada classe.
  - São declarados fora de um bloco e ficam “solto” dentro da classe.





# Exemplo

- Usando uma classe e atributos:

```
public class Programa {  
    public static void main(String[] args) {  
        Conta conta1 = new Conta();  
        conta1.nome = "Franciéric";  
        conta1.saldo = 1000.0;  
        System.out.println("Saldo: "  
            + conta1.saldo);  
    }  
}
```

# Métodos

- Definem os comportamentos de uma classe, ou seja, o que ela faz.
- Todo método deve ter um retorno, mesmo que seja void

| Conta   |  |
|---------|--|
|         | <code>numero: int</code><br><code>saldo: double</code><br><code>limite: double</code><br><code>nome: String</code> |
| Métodos | <code>saca(valor:double)</code><br><code>deposita(valor:double)</code>   |

# Exemplo de métodos

```
public class Conta {  
    int numero;  
    String nome;  
    double saldo;  
    double limite;  
  
    void saca(double valor) {  
        saldo = saldo - valor;  
    }  
  
    void deposita(double valor) {  
        saldo = saldo + valor;  
    }  
  
    double consultaSaldo() {  
        return saldo;  
    }  
}
```

# Usando a classe Conta

```
public class Programa {  
    public static void main(String[] args) {  
        Conta conta1 = new Conta();  
        conta1.deposita(1000);  
        conta1.saca(2000);  
        System.out.println("Novo saldo:" +  
                           conta1.consultaSaldo());  
    }  
}
```

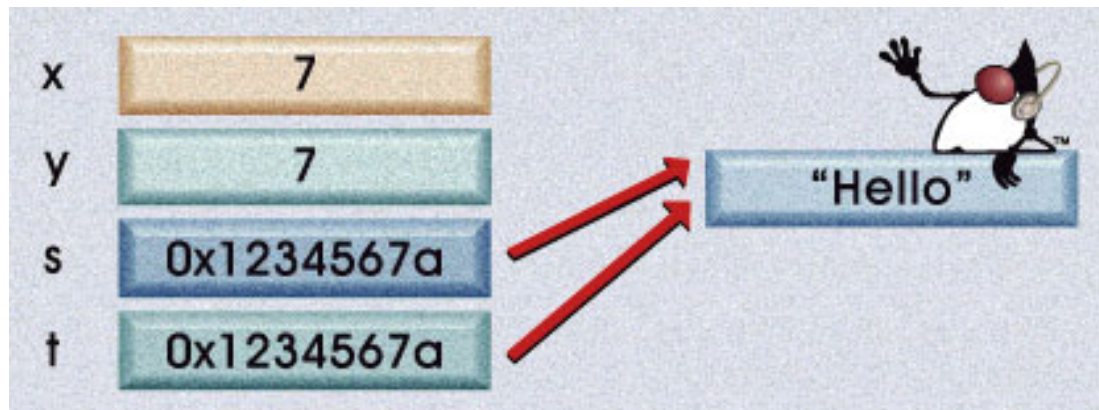
# Várias instâncias

```
public class TestaDuasContas {  
    public static void main(String[] args) {  
        Conta minhaConta = new Conta();  
        minhaConta.deposita(1000);  
  
        Conta meuSonho = new Conta();  
        meuSonho.deposita(1500000);  
    }  
}
```

# Referências

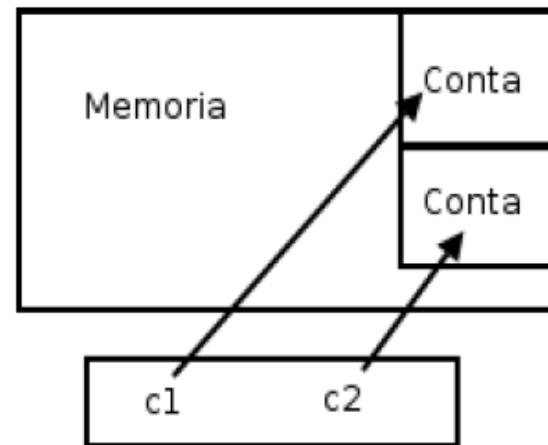
- Variáveis associadas a um objeto não guardam o objeto, mas uma referência para o objeto.
- Uma variável nunca é um objeto, mas uma referência para um objeto.

```
...  
int x = 7;  
int y = x;  
String s = "Hello";  
String t = s;  
...
```



# Referências

- Não é correto dizer: “**c1 é um objeto**”.
- Correto seria: “**Tenho uma referência c1 a um objeto do tipo Conta**”.
- No entanto, por praticidade diz-se: “**Tenho um objeto c1 do tipo Conta**”.
- c1 e c2 guardam um número que identifica a posição de memória do objeto Conta referenciado por cada uma delas.



# Qual o saldo?

```
public class TestaReferencias {  
    public static void main(String args[]) {  
        Conta c1 = new Conta();  
        c1.deposita(100);  
  
        Conta c2 = c1;  
        c2.deposita(200);  
  
        System.out.println(c1.consultaSaldo());  
        System.out.println(c2.consultaSaldo());  
    }  
}
```



# Referência this

- Todo objeto possui uma referência implícita para si mesmo, chamada **this**.
- A referência **this** é válida dentro de todo o corpo de uma classe, até mesmo no construtor.
- Quando um parâmetro tiver o mesmo nome de um atributo, **this** é utilizada para referenciar o atributo.

...


```
public void definirSaldoInicial(double saldo) {  
    this.saldo = saldo;  
}
```

...

# Objetos como parâmetros

- Para parâmetros que são objetos:
  - há a passagem apenas do endereço de memória.
  - não há cópia de objetos.
  - alterando-se um objeto passado como parâmetro, é o mesmo que alterar o objeto original

| Conta  |
|--|
| <code>numero: int</code><br><code>saldo: double</code><br><code>limite: double</code><br><code>nome: String</code>                             |
| <code>saca(valor:double): boolean</code><br><code>deposita(valor:double)</code><br><code>transfere(destino:Conta,valor:double): boolean</code> |



# Exemplo

```
public class Conta {  
    // atributos e metodos...  
    void transfere(Conta destino, double valor) {  
        this.saldo = this.saldo - valor;  
        destino.saldo = destino.saldo + valor;  
    }  
}
```

| Conta   |
|---|
| numero: int<br>saldo: double<br>limite: double<br>nome: String                                  |
| saca(valor:double): boolean<br>deposita(valor:double)<br>transfere(destino:Conta, valor:double) |