

# LINGUAGEM DE PROGRAMAÇÃO ORIENTADA À OBJETOS – LPOO

## Aula 06 – Introdução a Programação Orientada a Objetos (POO) -

Profa. Thais Rocha – [thais.rocha@docente.unip.br](mailto:thais.rocha@docente.unip.br)

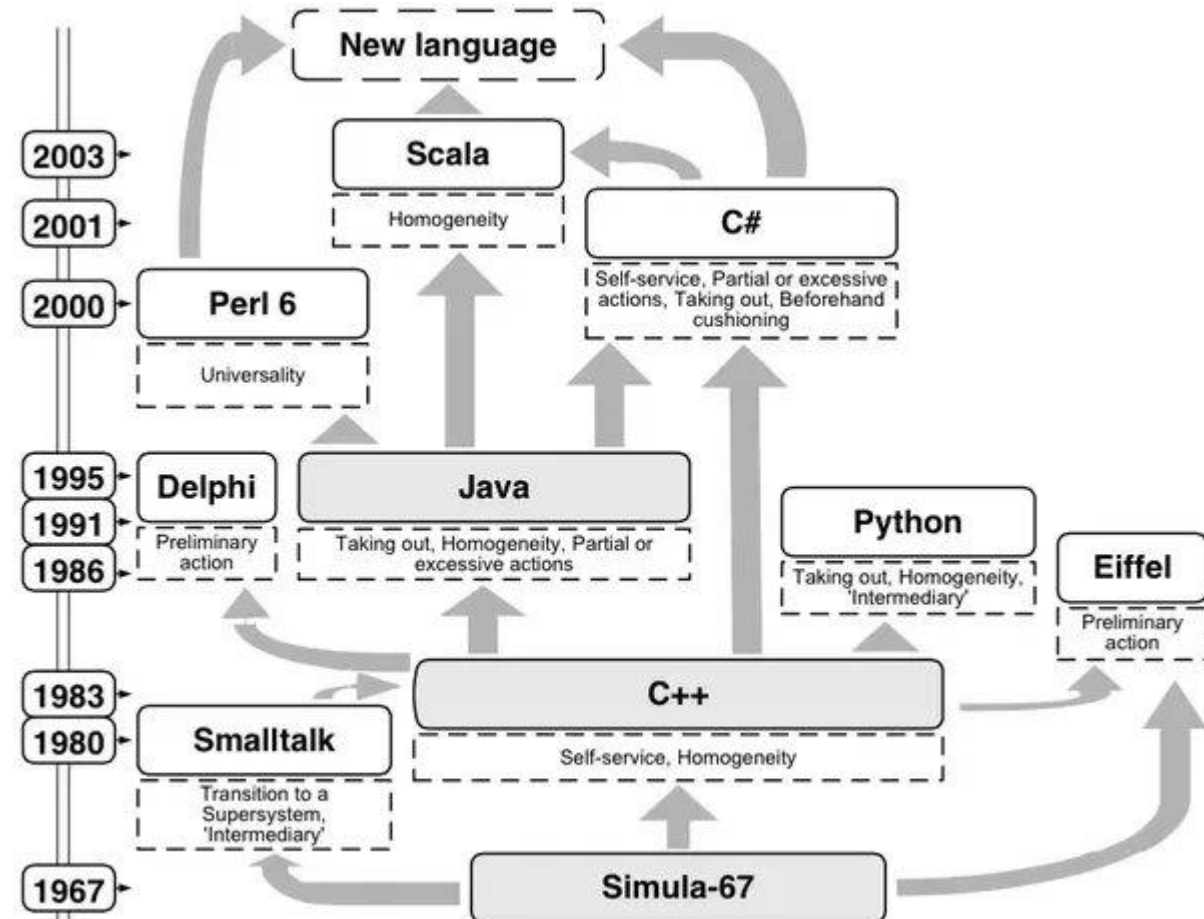


# TÓPICOS DE HOJE

- Historia da POO
- Classes e Métodos

# HISTÓRIA - OO

- A **OO** surgiu no final da década de **60**, quando dois cientistas dinamarqueses criaram a linguagem Simula (Simulation Language)
- **1967** - Linguagem de Programação Simula-67- conceitos de classe e herança



- O termo **Programação Orientada a Objeto (POO)** é introduzido com a linguagem Smalltalk (**1983**)
- FINS DOS ANOS 80 -> **Paradigma de Orientação a Objetos :**
  - abordagem poderosa e prática para o desenvolvimento de software
- Surgiram linguagens híbridas:
  - C++ (1986)**
  - Object-Pascal (1986)**

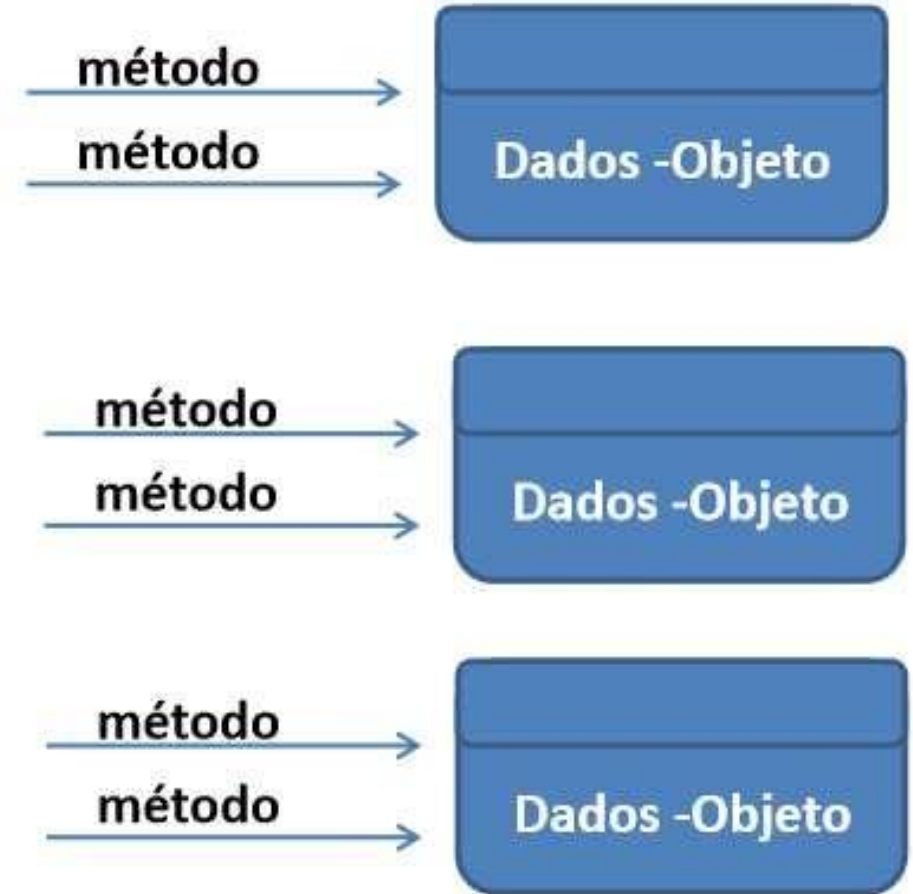
Surgiram diversos Métodos/Técnicas de Análise e Projeto OO :

- CRC (Class Responsibility Collaborator, Beecke e Cunningham, **1989**)
- OOA (Object Oriented Analysis, Coad e Yourdon, **1990**)
- Booch (**1991**)
- OMT (Object Modeling Technique, Rumbaugh, **1991**)
- Objectory (Jacobson, **1992**)
- Fusion (Coleman, **1994**)
- UML (Unified Modeling Language, **1997**)

Existem diversas linguagens OO, tais como:

- Smalltalk (**1972**)
- Ada (**1983**)
- Eiffel (~**1985**)
- Object Pascal (**1986**)
- Common Lisp (**1986**)
- C++ (~**1989**)
- Java (**1995**)
- C# (**2000**)

# PARADIGMA – PROGR. ESTRUTURAL X PROG. ORIENTADA A OBJETO



# PARADIGMA – PROGR. ESTRUTURAL X PROG. ORIENTADA A OBJETO

- Consiste no mapeamento do problema no mundo real a ser resolvido num modelo computacional.
- **Programação Estrutura:**
  - Consiste na criação de um conjunto de procedimentos (algoritmos) para resolver o problema
  - Encontrar modos apropriados de armazenar os dados
- **Programação Orientada a Objetos:**
  - Consistem em identificar os objetos e as operações relevantes no mundo real
  - O mapeamento desses em representações abstratas no espaço de soluções



## POO:

- Paradigma de Programação
  - Dominante nos dias atuais
- Substituiu as técnicas de programação procedimental (estruturada)
- “Fornece um mapeamento direto entre o mundo real e as unidades de organização utilizadas no projeto”
- Diversas unidades de software, chamadas de objetos, que interagem entre si
- Separa claramente a noção de o que é feito de como é feito

# PROGR. ESTRUTURAL X PROG. ORIENTADA A OBJETO

Programação estruturada	Programação orientada a objetos
• Variáveis	• Variáveis • Atributos
• Procedimentos	• Métodos
• Funções	• Métodos
• Operações matemáticas e lógicas	• Operações matemáticas e lógicas
• Laços e desvio condicionais	• Laços e desvio condicionais

# O QUE É POO ?

## **Paradigma de Programação :**

Tudo é um OBJETO

Objetos são abstrações do mundo real ou entidades do sistema que se auto gerenciam

Objetos são independentes e encapsulam representações de informação e estado.

**Exemplos de linguagens:** JAVA, C#, C++, Python

## **Evolução do ambiente de Programação**

Tamanho do código  
Configuração das equipes  
Requisitos do cliente  
Prazos de projetos

## **Qualidade do Código**

código limpo  
estilo consistente  
ser fácil de entender

## Produtividade:

Provê uma melhor organização do código.  
Contribui para o reaproveitamento de código.

Facilidade de estender o software devido a duas razões:

- herança: novas classes são construídas a partir das que já existem
- as classes formam uma estrutura fracamente acoplada o que facilita alterações

## Produtividade:

- melhora de comunicação entre desenvolvedores e clientes;
- redução da quantidade de erros no sistema, diminuindo o tempo nas etapas de codificação e teste;
- maior dedicação à fase de análise, preocupando-se com a essência do sistema;
- mesma notação é utilizada desde a fase de análise até a implementação.
- Manutenção do código: a modularização natural em classes facilita a realização de alterações no software

## Reuso do códigos

Os objetos são componentes potencialmente reutilizáveis.

○ encapsulamento dos métodos e representação dos dados para a construção de classes facilitam o desenvolvimento de software reutilizável, auxiliando na produtividade de sistemas.


# PRINCIPAIS VANTAGENS - POO

**PE (Programação Estruturada):** É possível reutilizar códigos na programação estruturada, porém em muitos casos você será obrigado a utilizar o famoso "CTRL C + CTRL V".

**POO:** Com a POO você é capaz de elaborar um relacionamento entre diversos componentes, estabelecendo comunicação entre eles e facilitando assim, e muito a reutilização de código, além da facilidade de se herdar atributos e comportamentos de outros objetos.



# OVERVIEW - POO



**Classes  
(Objetos)**

**Atributos**

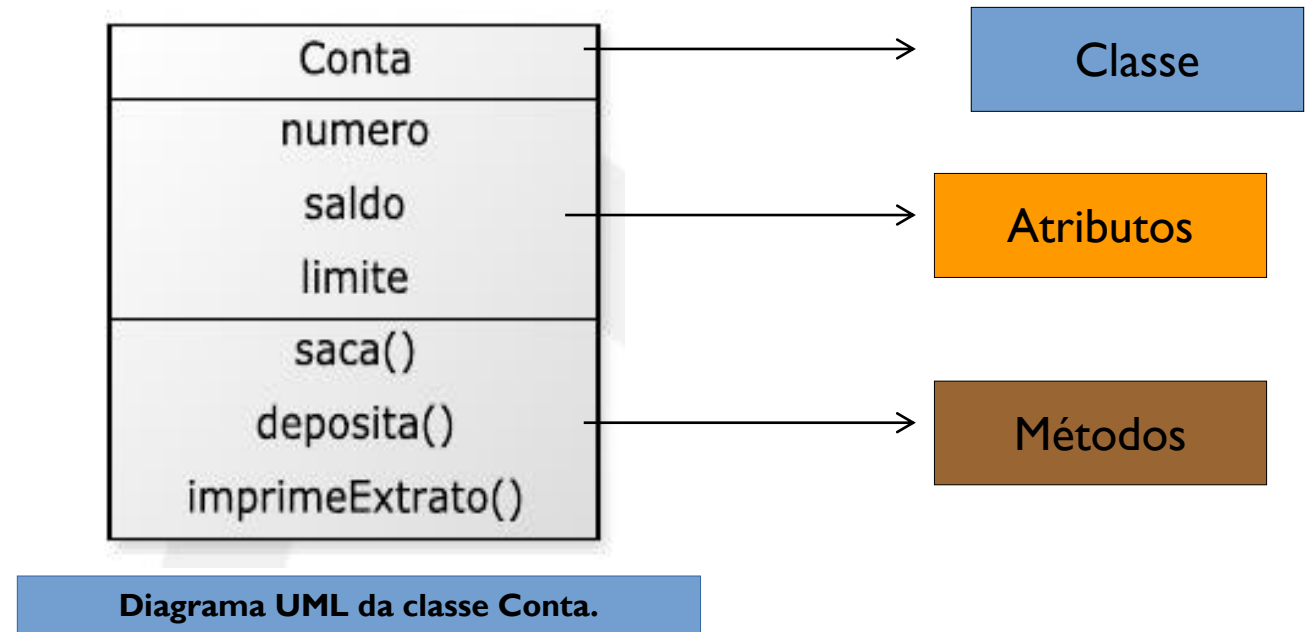
**Métodos**

# CLASSES

- **O desenvolvimento de classes é o resultado da abstração.** Uma vez definidos os devidos escopos, teremos condições de pensar em termos de classes.
- **Classes** consistem na unidade básica da construção de um programa POO e definem o conjunto de **atributos** mantidos por um conjunto de **objetos** e o **comportamento** que esses objetos devem respeitar.

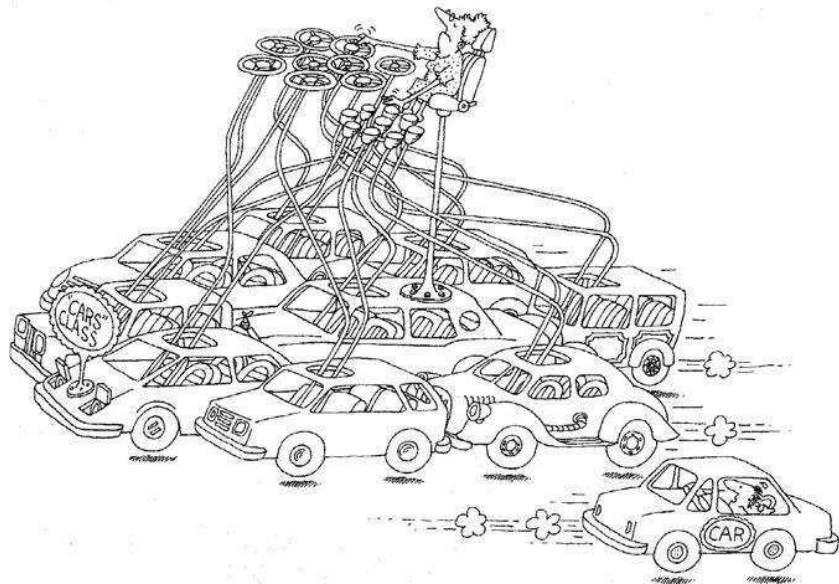
# CLASSES

Podemos representar uma classe através de diagramas UML. O diagrama UML de uma classe é composto pelo nome da mesma e pelos atributos e métodos que ela define.



# CLASSES

- São especificações para objetos;
- Representam um conjunto de objetos que compartilham características e comportamentos comuns.



Todo carro tem em comum:

**Característica**

Cor

Pneu

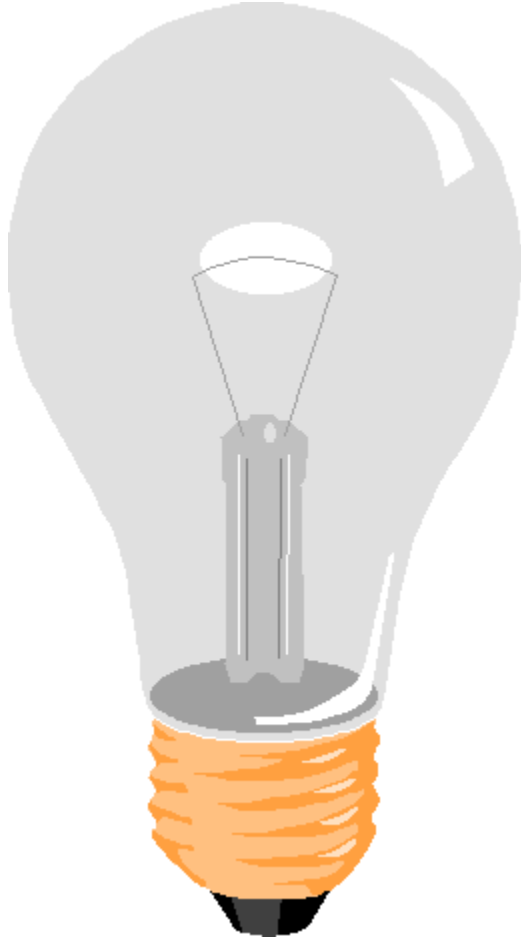
Direção

**Comportamento**

Dirigir

Frear

# CLASSES



**Nome da classe**

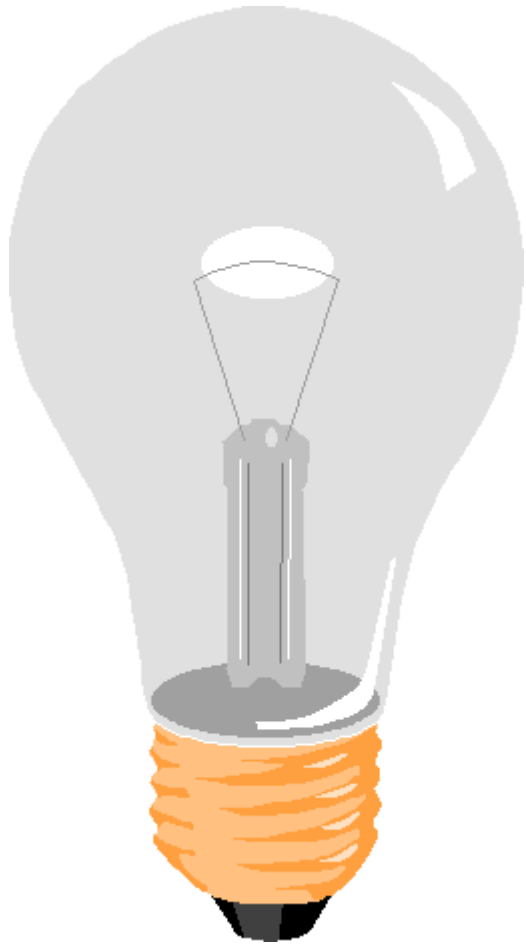
**Atributos**

**métodos**

**Lampada**

-ligada: Boolean  
-potencia: double

+ligar():void  
+desligar(): void  
+estaLigada(): boolean



- Classe Lampada
  - Atributos
    - potencia, ligada
  - métodos
    - ligar, desligar, estaLigada

Lampada
-ligada: Boolean -potencia: double
+ligar():void +desligar(): void +estaLigada(): boolean

# CLASSES

Declaração de uma classe em **Java**:

```
[<modificadores_da_classe>] class <nome_da_classe>  
  
    [extends <nome_superclasse>]  
  
    [implements <interface_1>, <interface_2>, ...] {  
  
        // variáveis e métodos da classe  
    }
```

# CLASSES EM JAVA - EXEMPLO

```
public class Lampada {  
    private boolean ligado;  
    public void ligar( ) {  
        ligado= true;  
    }  
    public void desligar() {  
        ligado=false;  
    }  
    public bool estaLigada( ){  
        return ligado;  
    }  
}
```



# DEFINIÇÃO DE OBJETOS

- Tudo em OO é OBJETO
- Um objeto é algo do mundo real :
  - Concreto ou Abstrato
- A percepção dos seres humanos é dada através dos objetos
- Um objeto é uma entidade que exibe algum comportamento bem definido.



**OBJETO = DADOS + OPERAÇÕES**

# OBJETOS

- Objetos podem ser considerados uma imitação do comportamento de entidades reais.
- Tal como em sistemas reais, em uma POO não é viável abrir um objeto e olhar em seu interior e tampouco alterar seu estado.
- Nesse paradigma, a única forma de fazer evoluir um programa é permitir que **objetos compartilhem dados** entre si a partir de **trocadas explícitas de mensagens**.

- **Características**

- Dados representam características

- São chamados **atributos**
    - São as variáveis do objeto
    - Ex: nome, sexo, raça

Animal
-nome: String -sexo: String -raca: String
+emitirSom() +dormir() +caminhar()



- **Comportamento**

- Operações definem comportamento

- São os **métodos** de um objeto
    - São as funções que são executadas por um objeto
    - Ex: emitirSom, dormir, caminhar

Animal
-nome: String -sexo: String -raca: String
+emitirSom() +dormir() +caminhar()



# OBJETOS

- **Estado**

- Representado pelos valores dos atributos de um objeto
- Ex: nome: Bobby, sexo: macho, raça: beagle

Animal
-nome: String -sexo: String -raca: String
+emitirSom() +dormir() +caminhar()



# OBJETOS

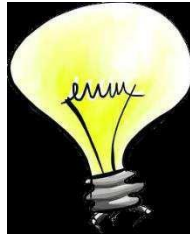
- **Identidade**
  - Um objeto é único, mesmo que o seu estado seja idêntico ao de outro;
  - Seu valor de referência
- Os valores dos **DADOS** são modificados a partir das **OPERAÇÕES** sobre estes dados



Animal
-nome: String -sexo: String -raca: String
+emitirSom() +dormir() +caminhar()

# OBJETOS - PROPIEDADES

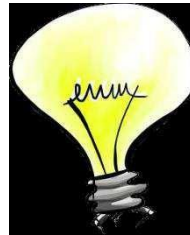
- Estado



Acesa

Apagada

- Comportamento



Acender

Apagar

- Identidade



# MÉTODOS

**Método** corresponde a uma ação (operação ou comportamento) que uma classe pode realizar.

**Cada método funciona de forma independente, sendo utilizado apenas quando a ação é solicitada.**

**Todo o método é criado seguindo regras:**

- As tarefas que um objeto pode realizar são definidas pelos seus métodos.
- Os métodos são os elementos básicos para a construção dos programas OO.
- Métodos não podem ser criados dentro de outros métodos ou fora de uma classe.



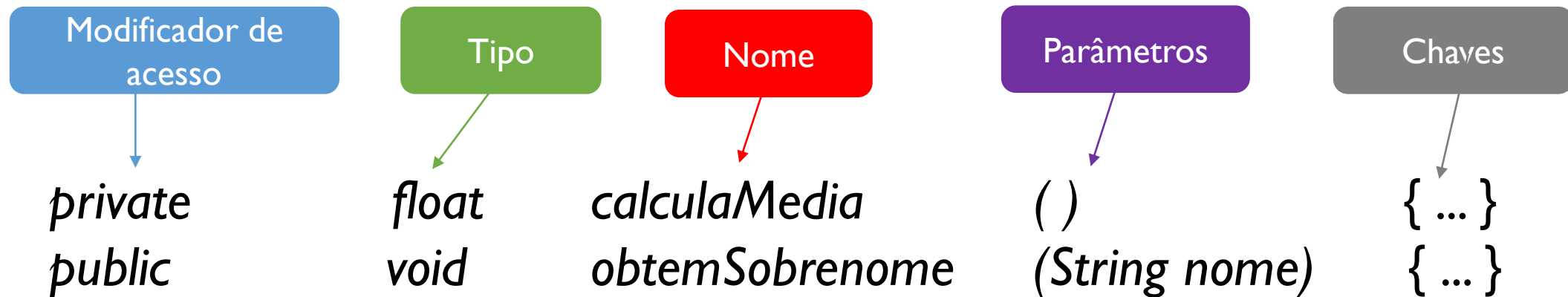
# CRIANDO MÉTODOS

- Cada método deve incluir as **2 partes (cabeçalho e o corpo)** apresentadas
- **Cabeçalho (header)** - o cabeçalho de um método fornece informações sobre como outros métodos pode interagir com ele. *Um cabeçalho de método também é chamado de **declaração**.*
- **Corpo (body)** - O corpo do método contém o declarações que realizam o trabalho do método. *O **corpo de um método** é chamado de **implementação**.*

# ASSINATURA DO MÉTODO

O cabeçalho (assinatura) do método é a primeira linha de um método. Ele contém o seguinte:

- **Modificador de acesso:** public ou private;
- **Tipo do retorno:** identifica o tipo de dados que a variável irá armazenar;
- **Nome:** começando com letra minúscula;
- **Parâmetros (opcional)**
- **Chave**



# MÉTODOS – TIPOS DE RETORNO

- Um tipo de retorno descreve o **tipo de dados que o método envia de volta para sua chamada método**
- Nem todos os métodos retornam um valor para seus métodos de chamada;
- Os métodos sem retorno tem um tipo de retorno de específico chamada de **void**.

# MÉTODOS – PARÂMETROS

- Alguns métodos exigem que os dados sejam enviados a eles quando forem chamados.
- Os dados que você usa em uma chamada a um método são chamados de **argumentos ou parâmetros**.
- Métodos que recebem parâmetros são flexíveis porque podem produzir diferentes resultados dependendo de quais dados eles recebem.

# MÉTODOS – PARÂMETROS

- Como um exemplo da vida real, quando você faz uma reserva em um restaurante, não precisamos criar um método diferente para cada data do ano em todas as horas possíveis do dia.
- Em vez disso, você pode fornecer a data e hora **como parâmetros**, para a pessoa que faz a chamada do método.
- O método que faz o registro da reserva, é então realizado da mesma maneira, não importa o que data e hora são fornecidas.


# CRIANDO MÉTODO COM APENAS UM PARÂMETRO

- Quando um método pode receber um parâmetro, sua declaração contém os mesmos elementos que um que não aceita um parâmetro
  - modificadores de acesso (opcional)
  - o tipo de retorno para o método,
  - nome do método
  - conjunto de parênteses que inclui dois itens:
    - tipo do parâmetro e o nome

***public static void predictRaise(double salary)***

No cabeçalho do método para `predictRaise()`, o parâmetro `salary` indica que o método receberá um valor do tipo `double`, e que dentro método, o valor passado será conhecido como `salary`.

# CRIANDO MÉTODO COM APENAS UM PARÂMETRO



The diagram illustrates the components of a method parameter. Two boxes at the top are connected by lines to the parameter 'salary' in the code below. The box on the left, labeled 'Parameter data type', points to the word 'double'. The box on the right, labeled 'Parameter identifier', points to the word 'salary'.

```
public static void predictRaise(double salary)
{
    double newSalary;
    final double RAISE_RATE = 1.10;
    newSalary = salary * RAISE_RATE;
    System.out.println("Current salary: " +
        salary + "    After raise: " +
        newSalary);
}
```

# EXERCÍCIOS COMPLEMENTARES

- I) Faça um resumo sobre a **Histórico da Programação Orientada a Objetos (POO)**, contendo os seguintes tópicos :
- Quando surgir e qual foi a primeira linguagem de programação a utilizar conceitos de POO ?
  - Quem foi o criador da linguagem C# e Java ?
  - Diferença entre programação estrutural e programação orientada a objetos
  - Quais linguagens de programação são exclusivamente baseado no paradigma da programação estrutural ?



# EXERCÍCIOS COMPLEMENTARES

2) Sobre a **linguagem de programação Java**, responda as seguintes questões:

- a) Quais são os principais mitos e verdades ?
- b) Quais Framework/API que fazem parte do ecossistema Java
- c) Quais a diferença entre JDK, JRE e JVM ?
- d) Quais os principais etapas de compilação de um código-fonte escrito em Java ?
- e) Explique como é feito a instalação do Java e configuração das variáveis de ambientes JAVA\_HOME e PATH.
- f) Quais são as principais IDE para escrever códigos Java. Qual é o seu favorito e porque ?

# EXERCÍCIOS COMPLEMENTARES

3) Sobre os conceitos de **Programação Orientada a Objeto (POO)**, responda:

- a) Quais são os principais benefícios da POO em relação a Programação Estrutural ?
- b) Quais são os 3 principais pilares da POO ?
- c) O que são classes ? Dê exemplo justificar a sua resposta
- d) O que são objetos ? Quais as diferença entre eles ?
- e) Explique a diferença entre abstração, encapsulamentos e modularidade ?

