Aula 4

Programação II

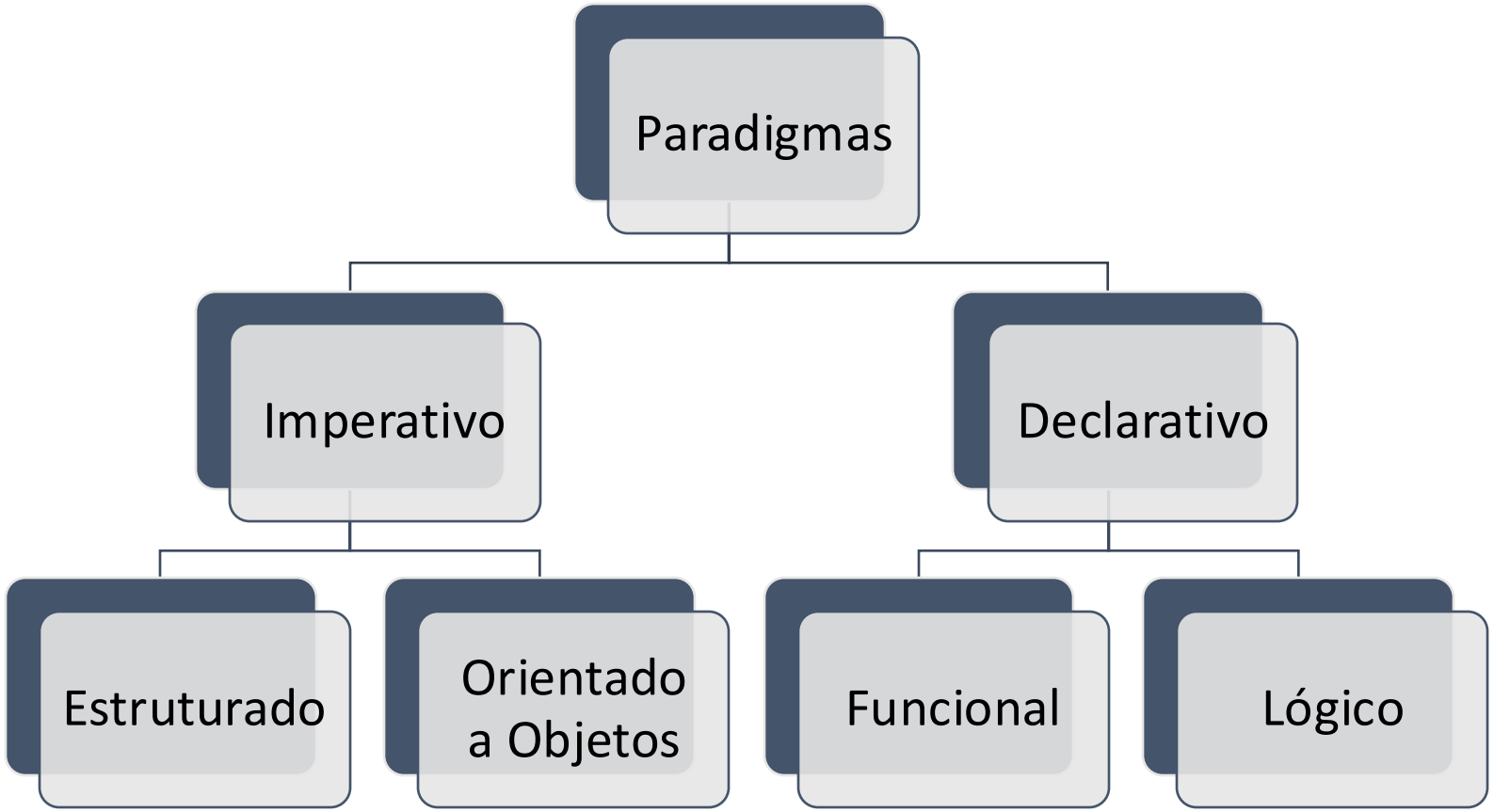
Prof. Sandino Jardim

CC-UFMT-CUA

Breve histórico

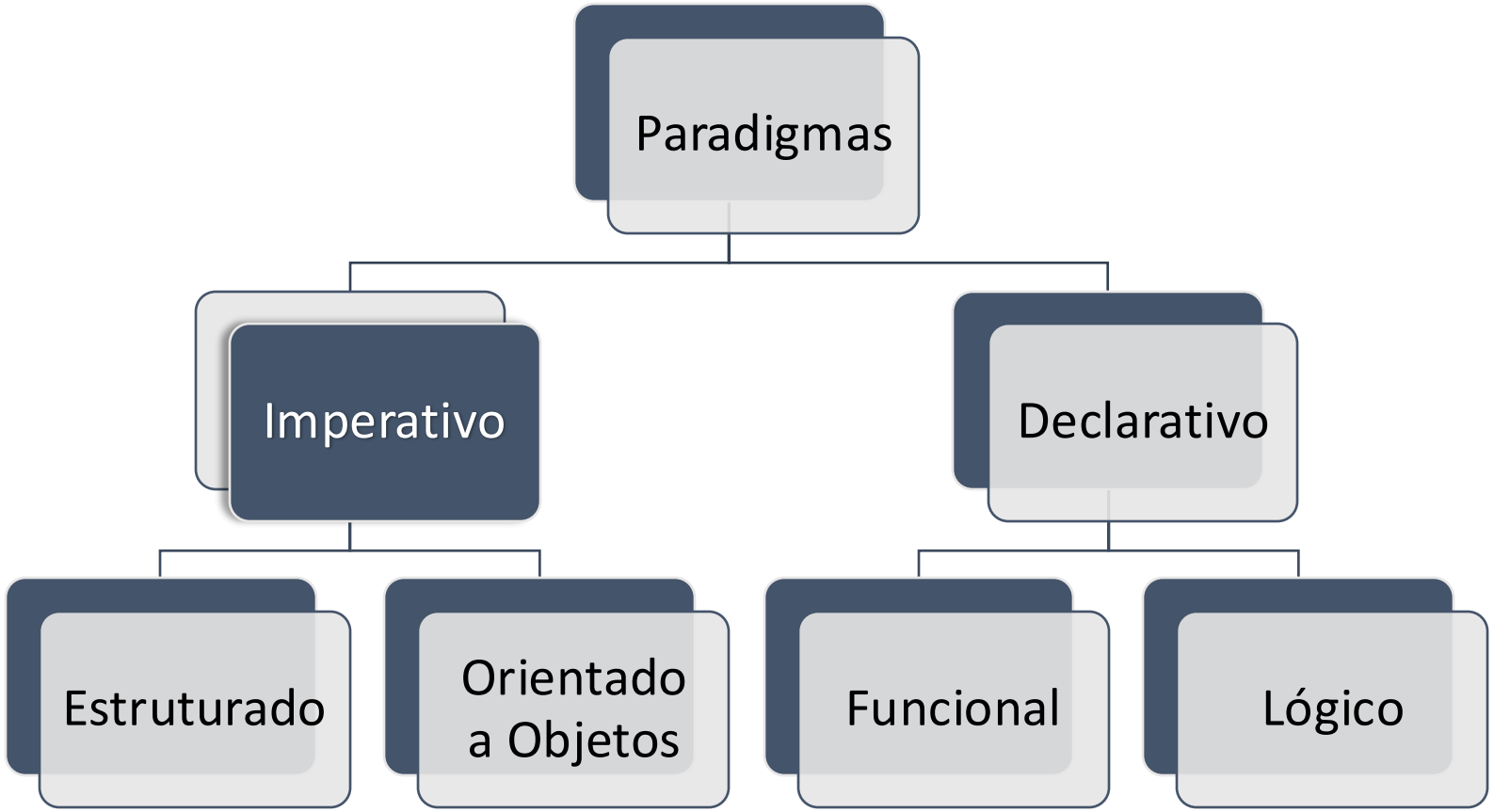
- Crise do Software
  - Dificuldade de escrever programas eficientes e úteis
  - Capacidade do hardware expandiu rapidamente
  - Métodos existentes eram inadequados
- Principais problemas:
  - Reusabilidade e extensibilidade de módulos
  - Representar entidades do mundo real
  - Projetar sistemas com interfaces abertas
  - Aumentar produtividade e diminuir custo
  - Gerenciar tempo de entrega
- POO
  - Existe como conceito desde anos 1950
  - Primeira aparição em uma linguagem no final dos anos 1960 (Simula)
  - Popularização a partir dos anos 1990

Hierarquia de Paradigmas



Hierarquia de Paradigmas

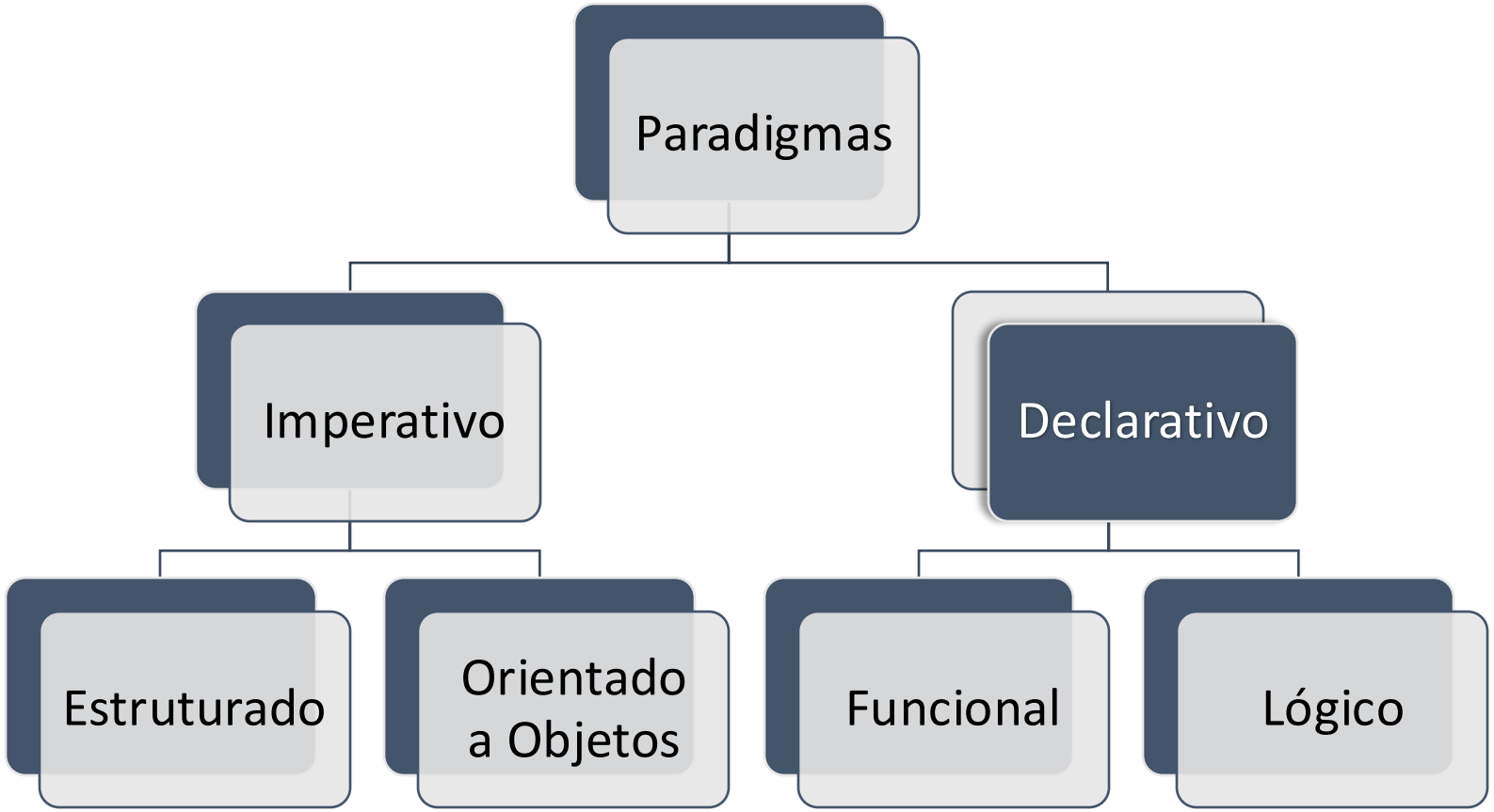
Imperativo



- Programas centrados no conceito de **estado** (variáveis) e **ações** (comandos)
- Computação como um processo que realiza mudanças de estado
- Especificam **como** o computador deve realizar uma tarefa

Hierarquia de Paradigmas

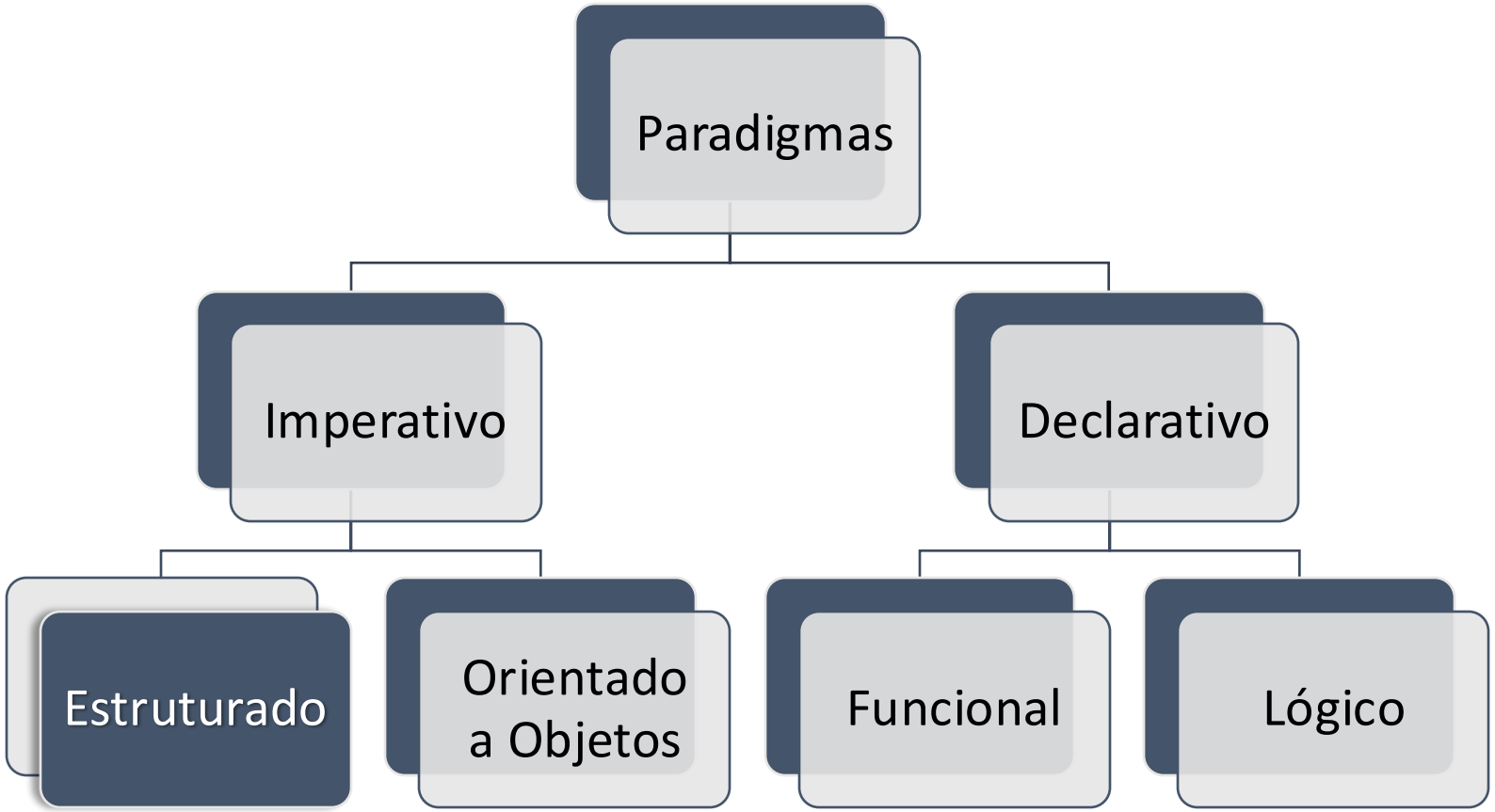
Declarativo



- Descrevem relação explícita e precisa entre as entradas e saídas
- Variáveis são incógnitas e não representam células de memória
- Especificam **o que** são as tarefas (Ex: LISP, PROLOG, SQL)

Hierarquia de Paradigmas

Estruturado



- Também conhecido como estrutural
- Separação clara entre dados e funções
- Modelo centrado nas funções

POO – Visão Geral

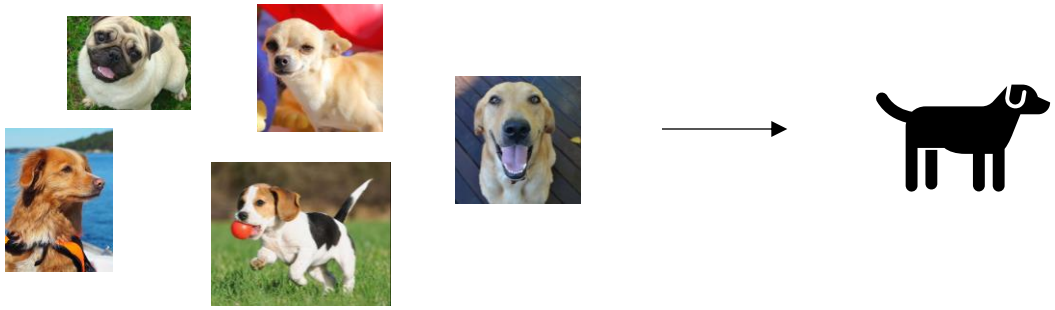
- Idéias-chave:
  - Abstração de dados
    - Separação entre interface e implementação
  - Herança
    - Modelagem de relacionamentos entre tipos similares
  - Vinculação dinâmica
    - Uso de objetos similares, ignorando detalhes que os diferem

POO – Visão Geral

- Programas são estruturados em módulos (**classes**) que agrupam um estado e operações
- Classes são usadas como tipos, cujas instâncias são **objetos**
- Programador usa **abstrações** mais próximas do mundo real
- Facilidade de **reutilização** do código
- **Separação** entre interface e implementação

Abstração

- “Processo conceitual onde **regras gerais** e conceitos são derivados de exemplos específicos”
- Conceito que atua como uma **definição comum** para todos os conceitos subordinados
- Seleção somente dos aspectos considerados **relevantes** para um **propósito particular**
- O **contrário** de **especificação**.



Abstração em POO

- Necessária para se concentrar apenas nos aspectos relevantes para o problema
- Define limites para o tamanho da classe
- Permite que a especificação possa ser gradativa
- Reduz complexidade de programação e utilização

Abstração em POO

Abstração

- Definição de classe

Especificação

- Instanciação de objeto

Objetos

- Objetos possuem **estrutura** (dados) e **comportamento** (funções)
- A estrutura de um objeto é representada em termos de **atributos**
- O comportamento de um objeto é representado pelo conjunto de **operações** que podem ser executadas a partir dele

Objetos

Exemplos

Class Impressora

Atributos
Velocidade
Resolução
Modo

Comportamento
`imprimirTeste()`
`imprimir()`
`ligar()`
`desligar()`

Objetos

Exemplos

`Impressora canon;`

Atributos
Velocidade: 32ppm
Resolução: 1600dpi
Modo: laser

Comportamento
`imprimirTeste()`
`imprimir()`
`ligar()`
`desligar()`

Classes

- Objetos com a mesma estrutura e o mesmo comportamento são agrupados em **classes**

Atributos
Velocidade
Resolução
Modo
Comportamento
`imprimirTeste()`
`imprimir()`
`ligar()`
`desligar()`

Atributos
Fabricante
Estilo
Cor
Comportamento
`tampar()`
`destampar()`
`escrever()`

Classes

- Descrevem um conjunto de objetos do mesmo tipo
- Cada objeto é dito ser uma instância de uma classe
- Cada instância de uma classe tem seus próprios valores para cada atributo
- Compartilham o mesmo comportamento com outras instâncias

Classes

- Classe: impressora

Atributos
Velocidade: 32ppm Resolução: 1600dpi Modo: laser
Comportamento

`imprimirTeste()`
`imprimir()`
`ligar()`
`desligar()`

Atributos
Velocidade: 20ppm Resolução: 1200dpi Modo: jato de tinta
Comportamento

`imprimirTeste()`
`imprimir()`
`ligar()`
`desligar()`

Classes

Exercício

- Defina a classe: Carro

Atributos
Cor Marca Modelo
Comportamento
partida( ) acendeFarol( ) buzina( ) acelera( ) freia( )

Atributos

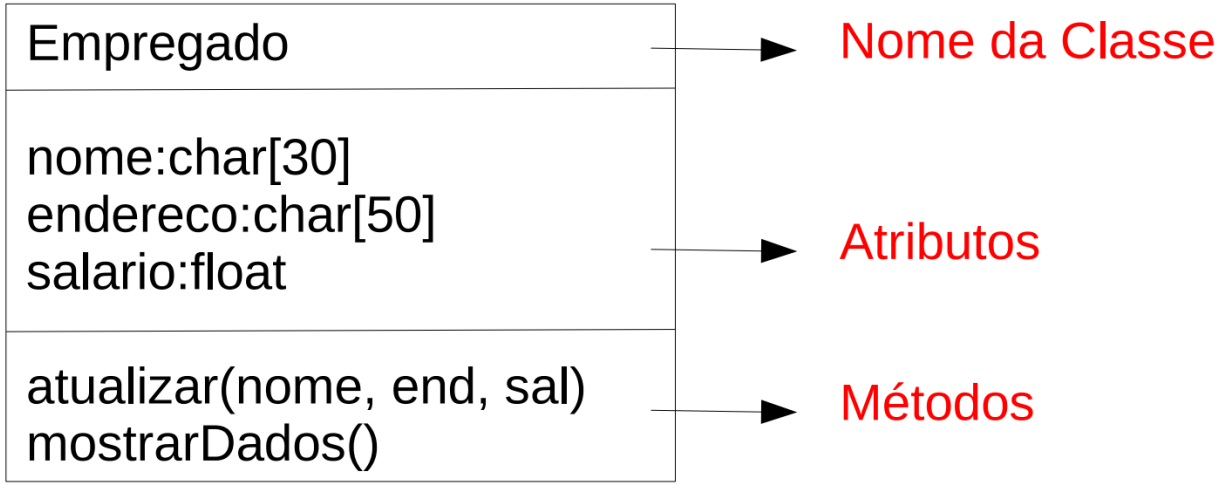
- Características específicas dos objetos
- Cada atributo tem um valor para um objeto particular
- Para cada atributo deve ser definido o **nome** do atributo e o **tipo** do valor que pode ser armazenado

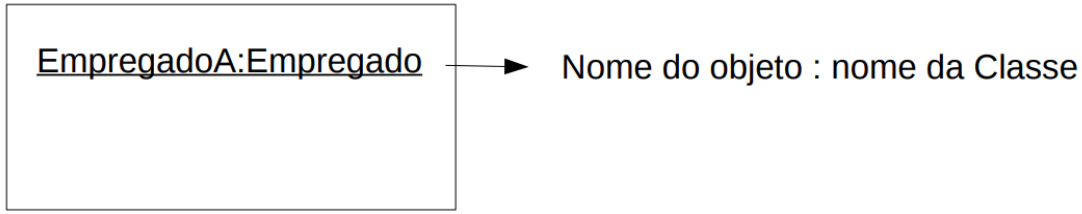
Métodos

- Funções que podem ser aplicadas em/por um objeto de uma classe
- Conjunto de métodos forma o comportamento
- Objetos da mesma classe compartilham os mesmos métodos
- Como funções, podem ter parâmetros

Classes e objetos

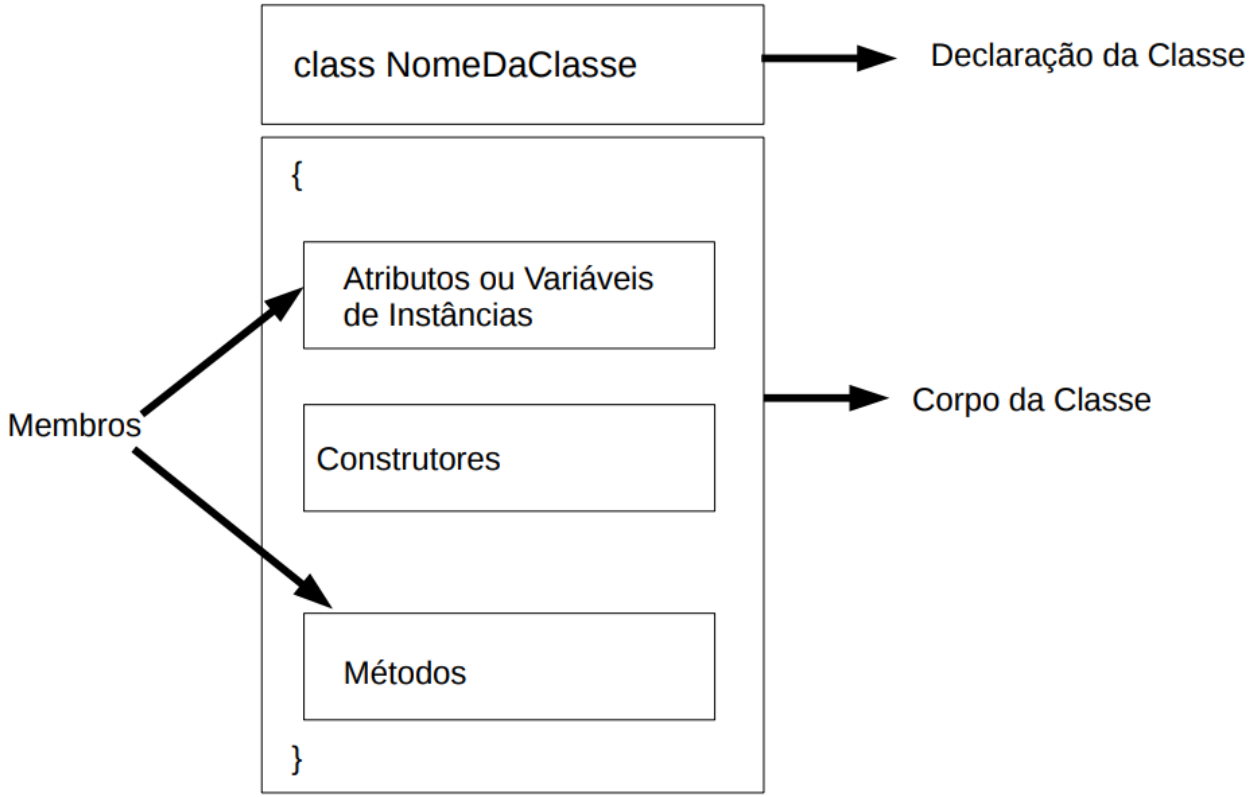
Representação UML





Definição de classes em C++

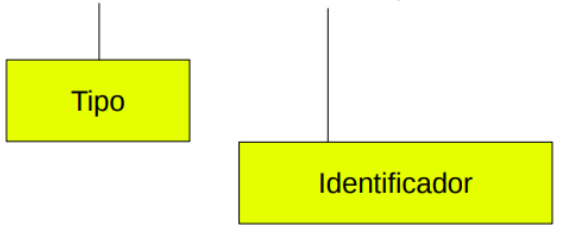
- Estrutura da classe



Definição de classes em C++

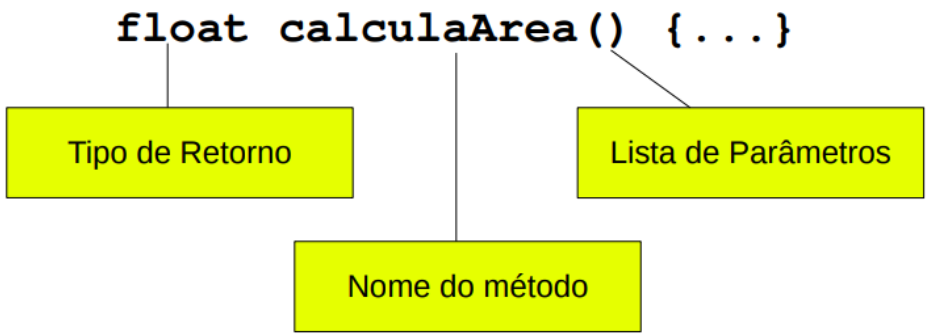
- Variáveis de instância
  - Conjunto de identificadores que guardam os valores dos atributos de um determinado objeto
  - Exemplo:

```
float base;
float altura;
```



Definição de classes em C++

- Métodos
  - Declaração na forma de uma função
  - Parâmetros são passados quando o método é **invocado**




```
t1.calculaArea()
```

Instanciação, atribuição e invocação em C++

- Instanciação
  - Por enquanto:

```
Triangulo t1;
Triangulo t2;
```

- Atribuição
  - Por enquanto:

```
t1.base = 2.5;
t1.altura = 2.0;
```

- Invocação

```
t1.calculaArea();
```

Exercício de exemplo

- Escreva a classe Lampada, definindo além dos atributos comuns a uma lâmpada, o atributo do tipo booleano *ligada*
- Crie um método chamado **`ligar`**(), um chamado **`desligar()`** e um método chamado **`status()`**. Os dois primeiros devem alterar o atributo ligada, enquanto o segundo deve mostrar na tela se a lâmpada está ligada ou desligada.
- Instancie dois objetos desta classe e teste seus métodos no programa principal