


PROGRAMAÇÃO DE COMPUTADORES



Técnico Integrado em
INFORMÁTICA

Professor Jefferson Chaves
jefferson.chaves@ifc-araquari.edu.br

- Revisar os conceitos e teorias de Programação Orientação a Objetos (POO);
- Revisar os conteúdos que envolvem os pilares da POO

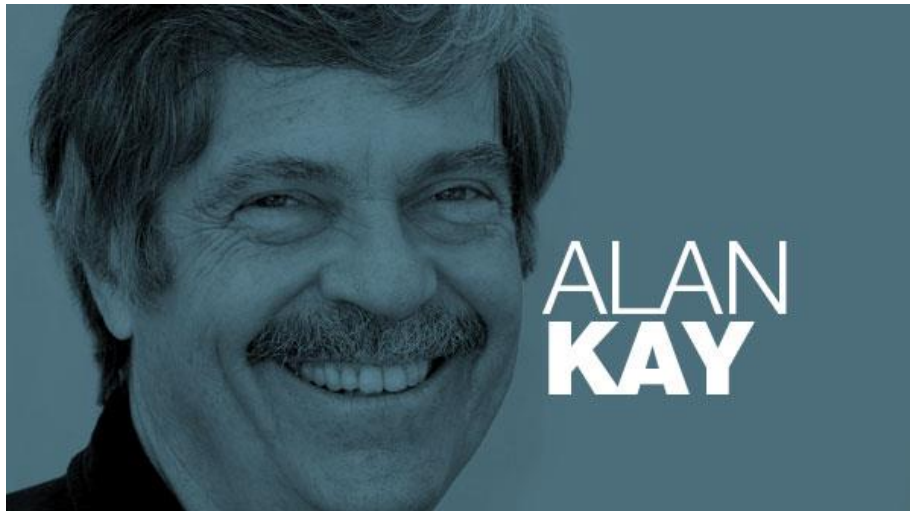
A faint, dark silhouette of an elephant is visible in the background on the left side of the slide, partially obscured by the text.

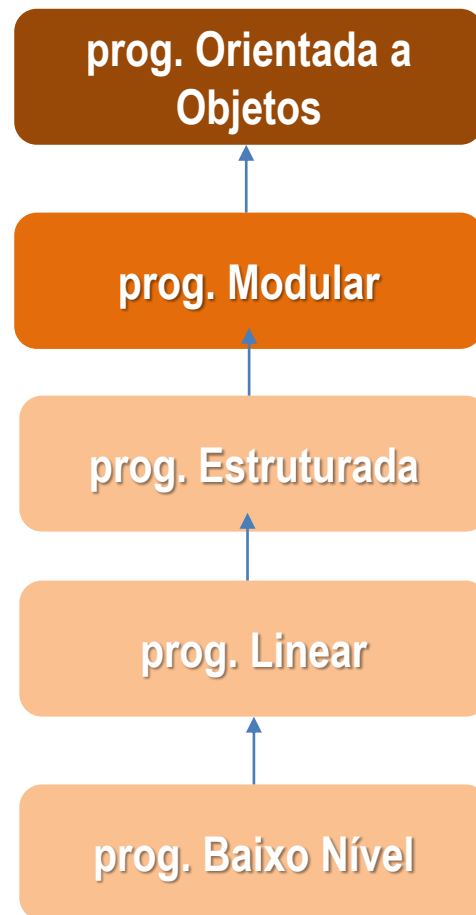
**PROGRAMAÇÃO DE
COMPUTADORES**

EVOLUÇÃO

ORIENTAÇÃO A OBJETOS BÁSICA

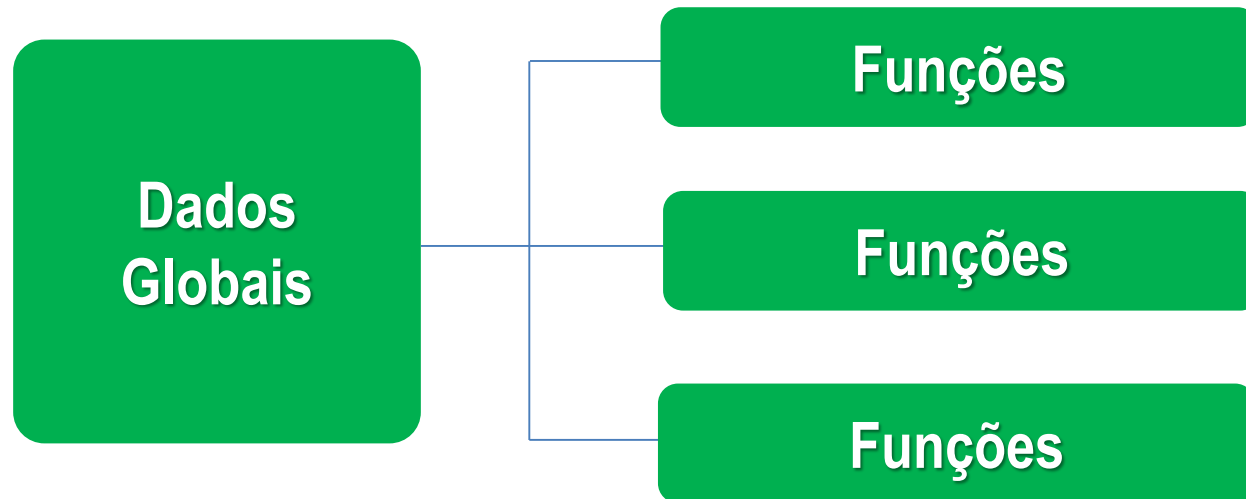
- A POO surgiu como uma evolução natural da programação;
- Proposto por Alan Kay;
 - Matemático e Biólogo;
 - Projetos focados na área de Educação de Crianças

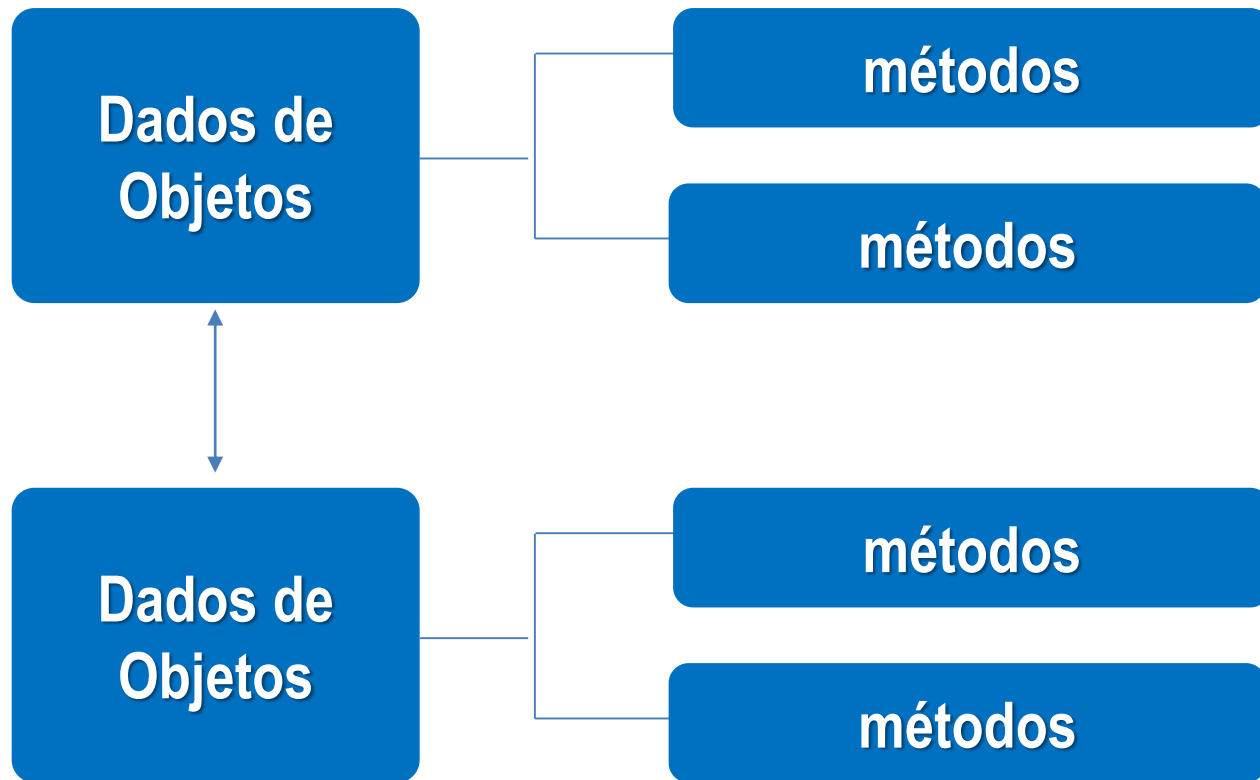




- A POO surgiu como uma evolução natural da programação;








- **Vantagens**

- **Conexão** forte entre dados e funcionalidades;
- Concentração de **responsabilidades**;
- Organização do código;
- Encapsulamento da lógica de negócios;
- **Polimorfismo** das referências;

A faint, dark green silhouette of an elephant is visible on the left side of the slide, partially obscured by the text.

**PROGRAMAÇÃO DE
COMPUTADORES**

CLASSES E OBJETOS

PROGRAMAÇÃO DE COMPUTADORES

ORIENTAÇÃO A OBJETOS



Objeto:

“(...)coisa material ou **abstrata** que pode ser percebida pelos sentidos e descrita por meio dos seus **atributos**, **comportamentos** e **estado**”

Objeto:

“(...)coisa material ou **abstrata** que pode ser percebida pelos sentidos e descrita por meio dos seus **atributos**, **comportamentos** e **estado**”



PROGRAMAÇÃO DE COMPUTADORES

ORIENTAÇÃO A OBJETOS



Uma **classe** deve responder a três perguntas para o objeto:

- **O que eu tenho;**

Atributos

- **O que eu faço;**

Métodos

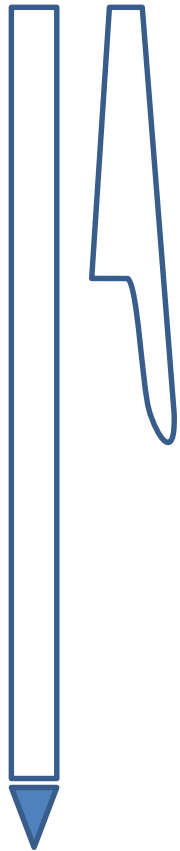
- **Como estou agora;**

Estado



- **Atributos:**
 - Modelo;
 - Cor;
 - Carga
 - Tampada;
- **Métodos:**
 - Escrever;
 - Rabiscar;
 - Tampar;
 - Destampar;





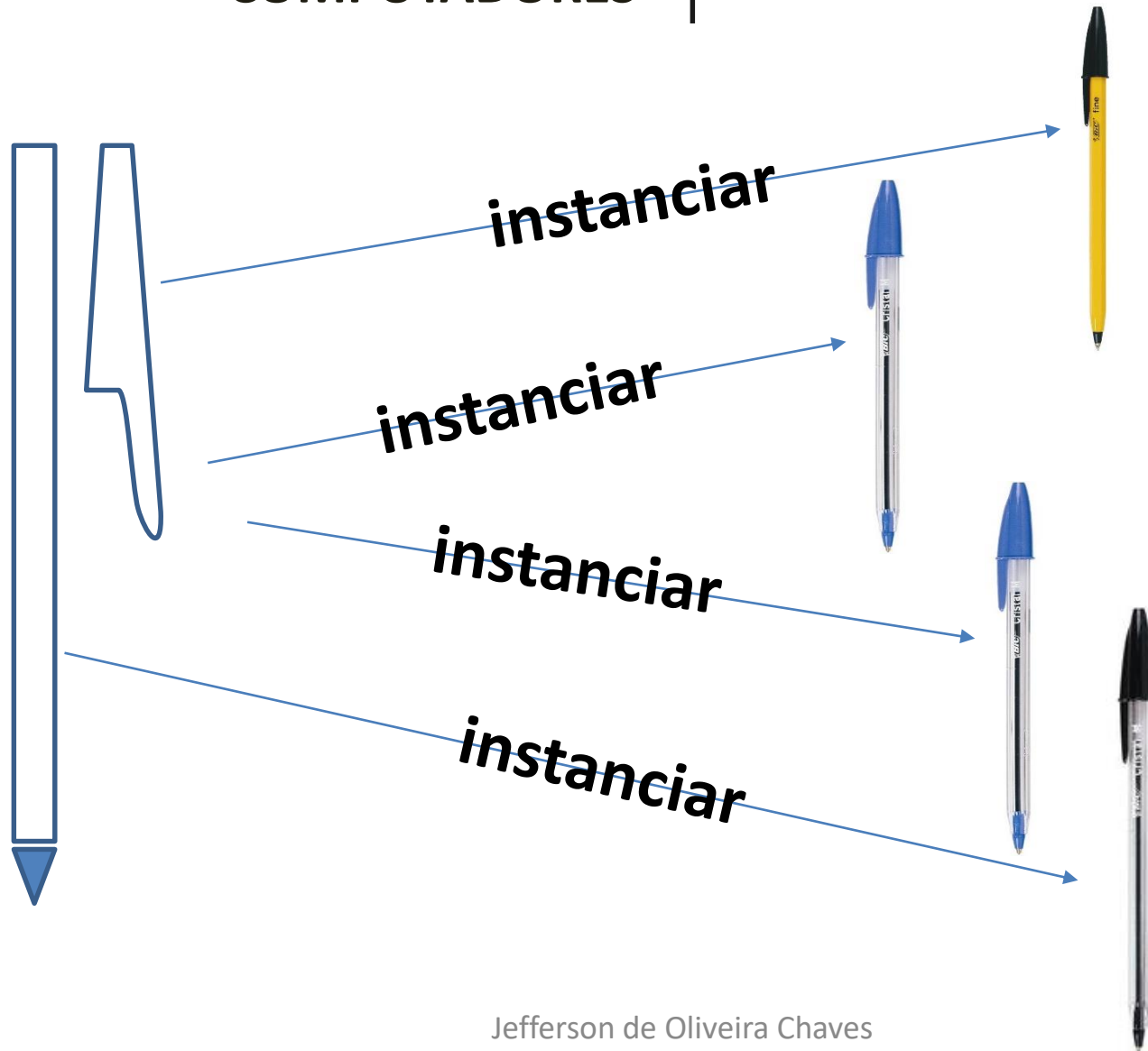
instanciar


\$can = new Caneta



PROGRAMAÇÃO DE COMPUTADORES

ORIENTAÇÃO A OBJETOS



A faint, dark silhouette of an elephant is visible in the background on the left side of the slide, partially obscured by the text.

**PROGRAMAÇÃO DE
COMPUTADORES**

RESUMINDO



- **Classe:**

- define os atributos e métodos que serão compartilhados por objetos;
- Uma classe definitivamente não é um objeto;



- **Objeto:**

- É a instância de uma classe;

- **PILARES DA POO**



ABSTRAÇÃO




ENCAPSULAMENTO



HERANÇA



POLIMORFISMO

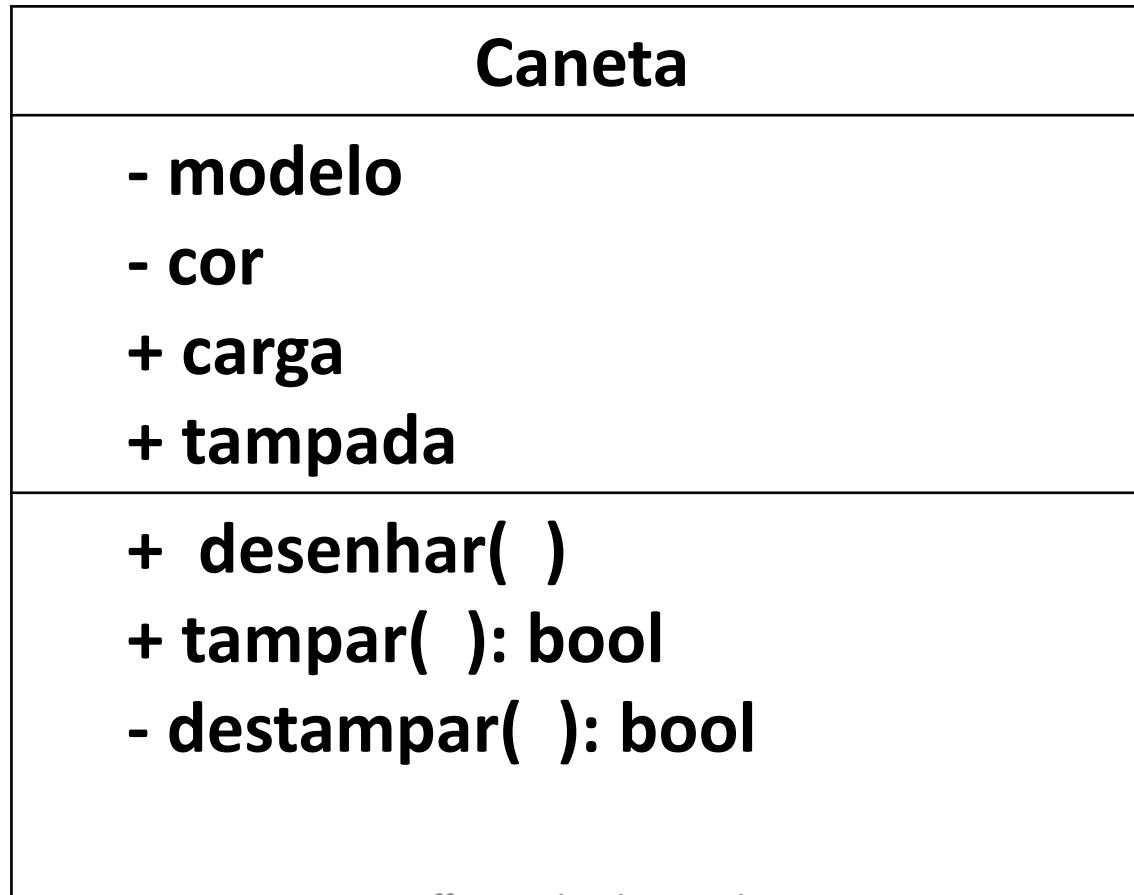
A faint, dark silhouette of an elephant is visible in the background on the left side of the slide, partially obscured by the text.

**PROGRAMAÇÃO DE
COMPUTADORES**

**VISIBILIDADE EM UM
OBJETO**

- Abrindo um parênteses para UML:
 - Diagrama de casos de uso;
 - Diagrama de classes;

- Diagrama de classes;



Acesso as ligações



- **+ Público:** a classe atual, outras classe e objetos poderão obter seu valor e modifica-la;
- **- Privado:** somente a classe atual tem acesso a esse atributo ou método;
- **# Protegido:** Somente a mãe e seus filhos;

- **+ Público:** a classe atual, outras classe e objetos poderão obter seu valor e modifica-la;
- **- Privado:** somente a classe atual tem acesso a esse atributo ou método;
- **# Protegido:** Somente a mãe e seus filhos;

- E como permitir o acesso **CONTROLADO** a um atributo privado?



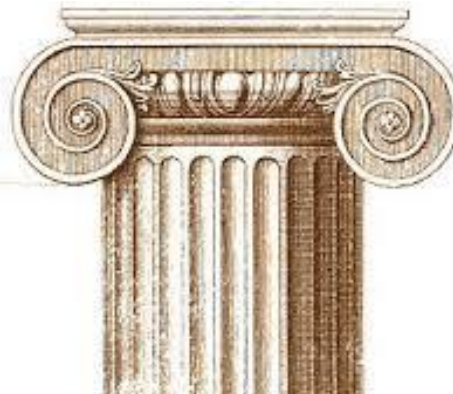
- **PILARES DA POO**



ABSTRAÇÃO




ENCAPSULAMENTO



HERANÇA



POLIMORFISMO

A faint, dark green silhouette of an elephant is visible in the background on the left side of the slide.

**PROGRAMAÇÃO DE
COMPUTADORES**

MÉTODOS ESPECIAIS

PROGRAMAÇÃO DE COMPUTADORES

ORIENTAÇÃO A OBJETOS



```
$cof = new Cofre( );  
$cof->saldo += 100;
```

**Informando ou alterando
atributos**

PROGRAMAÇÃO DE COMPUTADORES

ORIENTAÇÃO A OBJETOS



```
$cof = new Cofre( );  
$cof->saldo += 100;  
$cof->setSaldo($cli, 100)
```

Informando ou alterando atributos

PROGRAMAÇÃO DE COMPUTADORES

ORIENTAÇÃO A OBJETOS



```
$cof = new Cofre( );  
$total = $cof->saldo;
```

Acessando atributos



```
$cof = new Cofre( );  
$total = $cof->saldo;  
$total = $cof->getTotal ($cli);
```


Acessando atributos

- “O modificador **private** faz com que o atributo não possa ser lido e nem modificado de fora do escopo da classe”;
- “O modificador **protected** tem a mesmo comportamento que o private, mas permite o ~~acesso~~ as classes filhas”;

- Para permitir seu acesso (já que tem seu acesso *restrito*) a prática é criar dois métodos:
 - um que retorne o valor (get);
 - outro que informe ou modifique o valor (set);
- A **convenção** é colocar **get** ou **set** antes do nome do atributo;
- Não devemos criar **getters** e **setters** sem um motivo explícito.

- **Métodos Getters**

- Métodos acessores permitem o acesso controlado aos atributos;
- Normalmente são usados para validação da informação;
- Isso garante que todo atributo será íntegro.


A faint, dark silhouette of an elephant is visible on the left side of the slide, partially obscured by the text.

**PROGRAMAÇÃO DE
COMPUTADORES**

ATRIBUTOS DE CLASSE

- Atributos e métodos estáticos pertencem à classe e não a uma instância;
- Podem ser utilizados sem ter que instanciar a classe;
- Variáveis estáticas são compartilhadas por todos os objetos;
- Nesse caso, utiliza-se a palavra chave **static**;

- Métodos estáticos acessam **apenas variáveis estáticas e métodos estáticos**;
- Isso porquê não há objeto criado no momento em que o método estático é chamado.

A faint, dark silhouette of an elephant is visible in the background on the left side of the slide, partially obscured by the text.

**PROGRAMAÇÃO DE
COMPUTADORES**

CONSTRUTORES

- Quando usamos a palavra chave **new** estamos criando um objeto;
- Sempre que o new é chamado ele executa o **construtor da classe**;
- Construtor é um “método mágico” que é executado toda vez que o objeto é **instanciado**.
- Caso o construtor não estiver definido, um construtor padrão é utilizado


- Ao se declarar um construtor, o construtor padrão não será mais fornecido



- Um construtor tem por objetivo:
 - Realizar operações de inicialização;
 - Inicializar os atributos do objeto;
- A ideia é simples: **obrigar (ou dar a possibilidade)** o usuário de uma classe a passar argumentos para o objeto durante seu processo de criação.

Qual a necessidade de um construtor ?

Obrigar ou possibilitar o usuário de uma classe a passar argumentos para o objeto durante o processo de criação do mesmo

A faint, dark silhouette of an elephant is visible in the background on the left side of the slide, partially obscured by the text.

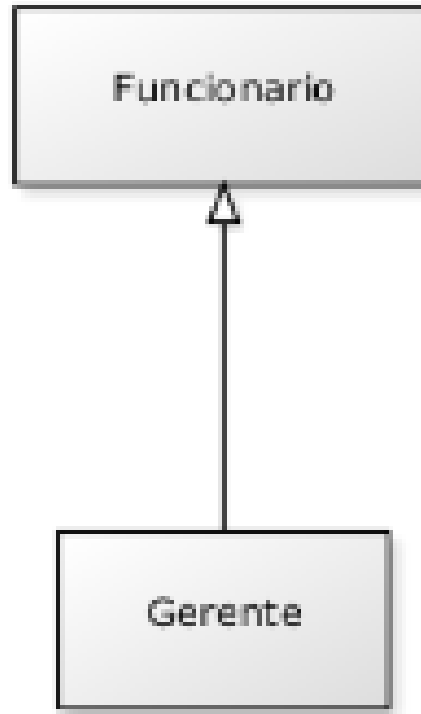
**PROGRAMAÇÃO DE
COMPUTADORES**

HERANÇA

- Existe um jeito, de relacionarmos uma classe de tal maneira que uma delas herda tudo que a outra tem;
- Isto é uma relação de classe mãe e classe filha;
- No nosso caso, gostaríamos de fazer com que o Gerente tivesse tudo que um Funcionario tem, gostaríamos que ela fosse uma extensão de Funcionario;
- Fazemos isto através da palavra chave extends;

PROGRAMAÇÃO DE COMPUTADORES

HERANÇA



Super e Sub classe

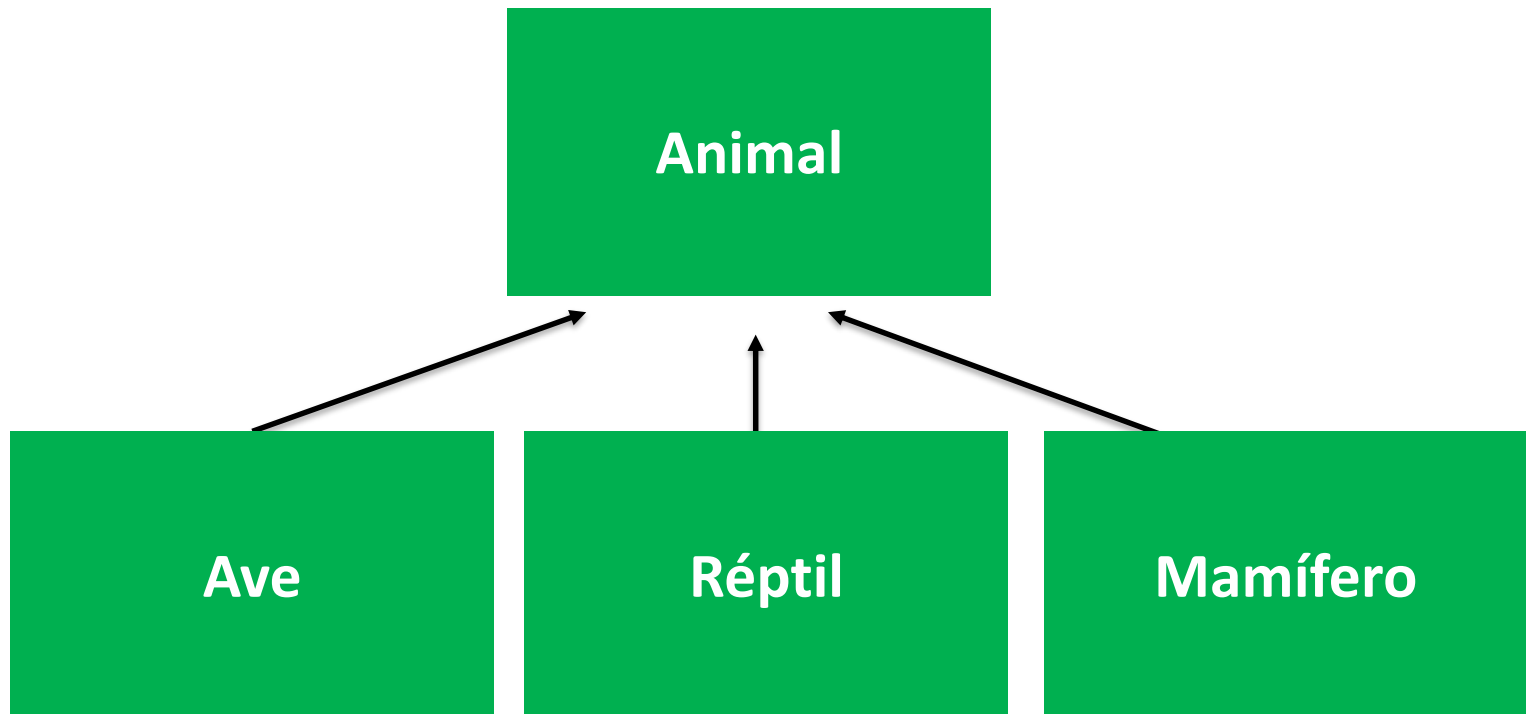
- A nomenclatura mais encontrada é que Funcionario é a superclasse de Gerente, e Gerente é a subclasse de Funcionario. Dizemos também que todo Gerente é um Funcionário.
- Outra forma é dizer que Funcionario é classe mãe de Gerente e Gerente é classe filha de Funcionario.

- E o modificador de acesso?
- Sempre usar o *#protected*?
- Posso sobrescrever?

- A **herança** além de outras vantagens, permite que o ***polimorfismo***;
- ***Polimorfismo*** é a capacidade de um objeto poder ser referenciado de várias formas.

- **Qual prejuízo teríamos se não pudessemos fazer esse tipo de referência?**
- Repare que esse poder é muito maior que a pequena vantagem que a herança traz em herdar código.

- **Vamos idealizar um sistema de zoológico**



- Usamos a palavra chave abstract para impedir que ela possa ser instanciada.
- Qual efeito isso causa?

- Um método abstrato indica que todas as classes filhas concretas (que não forem abstratas) devem reescrever esse método ou não executarão.
- É como se você herdasse a responsabilidade escrever aquele método.

1. Organize sua aplicação do banco: crie uma aplicação que permita a criação de contas em um banco.
2. Todo Banco deve ter uma lista de gerentes;
3. Toda Conta criada deve possuir um Banco e um Gerente(daquela banco). Se não houver, informe o usuário para que ele mesmo crie.
4. Você deve permitir escolher o tipo de conta.
5. Crie um método extrato na Conta que imprima na tela todos os dados da conta, como saldo, gerente e banco.