



**SUPER**  
**GEEKS**

# Programação Orientada a Objetos ou POO em português...

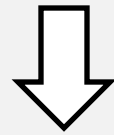


# OOP

## Object-Oriented Programming

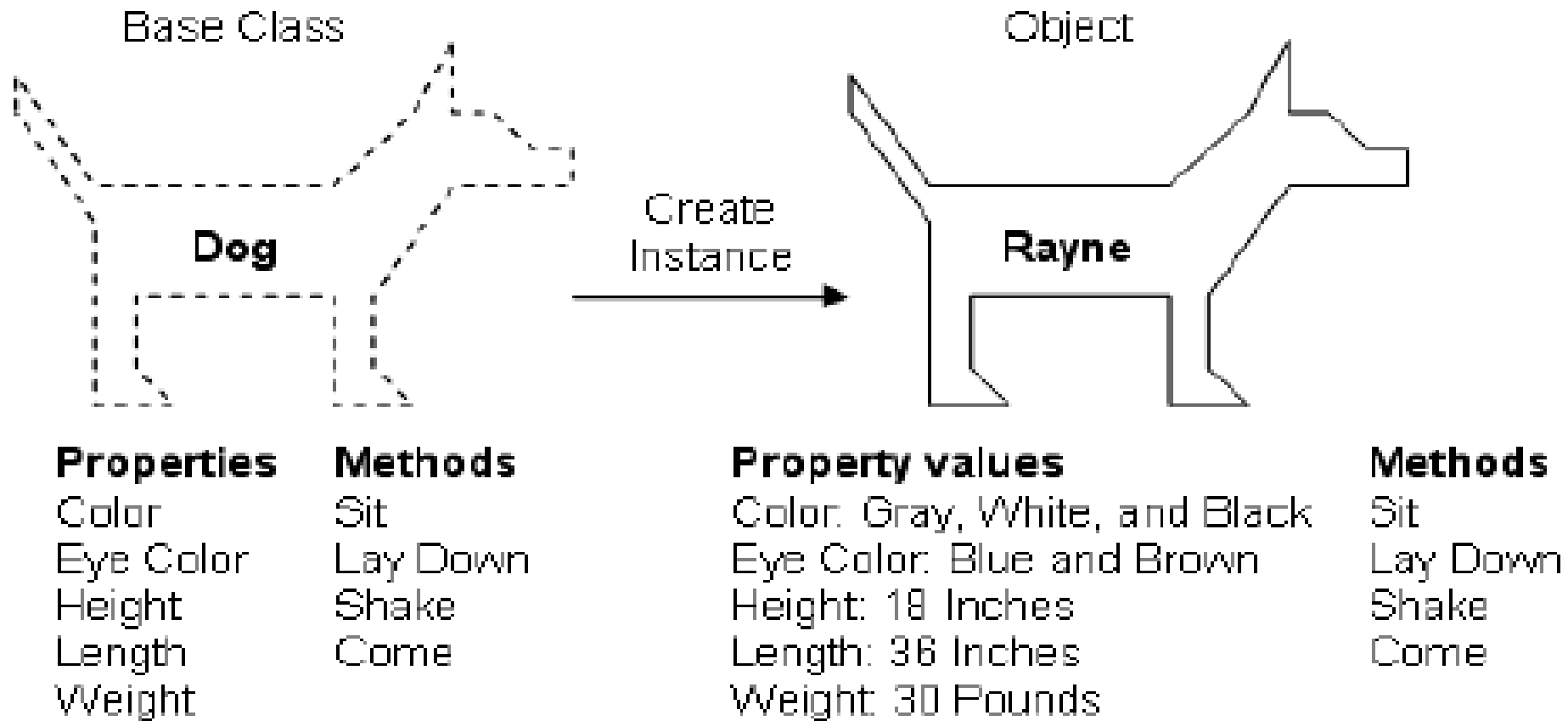


~~Orientação a Objetos~~



Pensar em Coisas

# Programação Orientada a Objetos





## Classe Dog

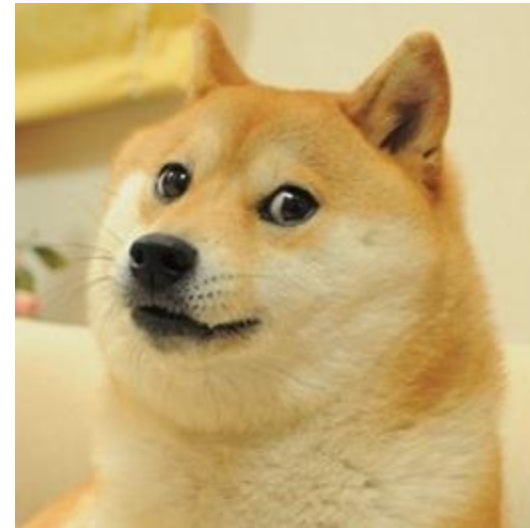
Propriedades (variáveis)		Métodos (funções)
Cor do pelo	---	Sentar()
Cor do olho	---	Deitar()
Altura	---	FingirDeMorto()
Comprimento	---	Buscar()
Peso	---	





## Classe Dog

Propriedades (variáveis)		Métodos (funções)
Cor do pelo	---	Sentar()
Cor do olho	---	Deitar()
Altura	---	FingirDeMorto()
Comprimento	---	Buscar()
Peso	---	



## Objeto Dog

Propriedades (variáveis)		Métodos (funções)
Cor do pelo	<b>Amarelo</b>	Sentar()
Cor do olho	<b>Preto</b>	Deitar()
Altura	<b>0.5m</b>	FingirDeMorto()
Comprimento	<b>0.6m</b>	Buscar()
Peso	<b>14kg</b>	



# Exemplo de Classe

```
class Dog
{
    string Color;
    string EyeColor;
    float Height;
    float Length;
    float Weight;

    -referências
    public void Sit()
    {
        ...
    }

    -referências
    public void LayDown()
    {
        ...
    }
}
```





# Diferença entre Classe, Objeto e Instância

- Classe é um conjunto de propriedades (variáveis) e métodos
- Objeto é a variável do tipo da classe que criamos
- Instância é a sua existência, é o seu uso

Com o objeto da classe manipulamos a sua existência, definindo e/ou usando suas propriedades durante a execução do código

Então um objeto não existe sem a definição de uma classe!

Através da definição de uma classe, podemos descrever:

- **quais propriedades e/ou atributos o objeto vai ter**
- **qual o comportamento que o objeto da classe terá**

# Exemplo de Objeto e Instância

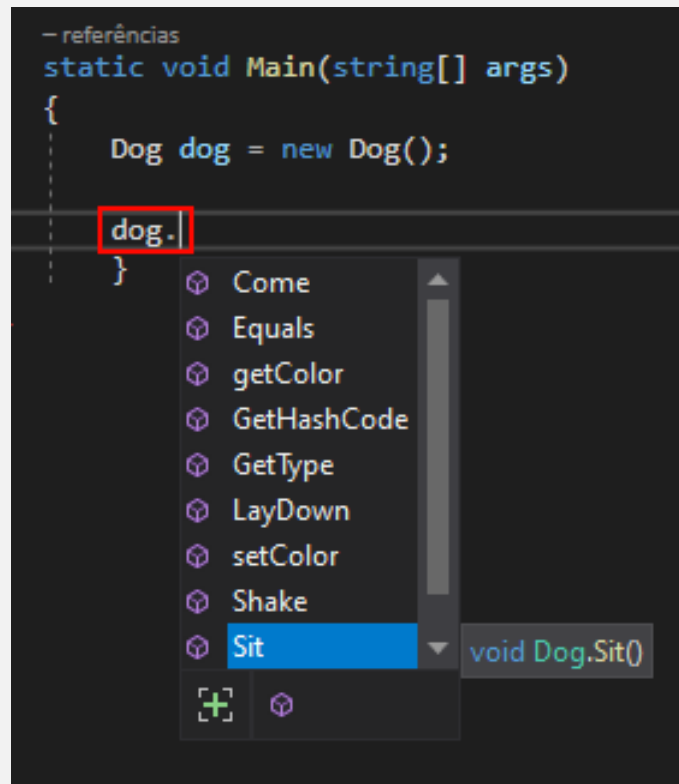
```
-referências
class Program
{
    -referências
    static void Main(string[] args)
    {
        Dog dog = new Dog();

        dog.Sit();
    }
}
```

- `dog`: objeto
- `new Dog()`: instância
- `dog.Sit()`: é aonde o método `Sit()` é acessado e chamado
- Como ele faz parte do objeto, precisamos usar o "." para chama-lo

# Operador de Acesso de Membro

- Depois que criamos a classe e instanciamos o seu objeto, precisamos usar seus métodos e variáveis, certo?
- É para isso que usamos o "." ao lado do objeto. É ele que mostra todos os métodos e variáveis declaradas como **PUBLICAS** da nossa classe.



```
- referências
static void Main(string[] args)
{
    Dog dog = new Dog();
    dog.
}
```

The dropdown menu shows the following methods:

- Come
- Equals
- getColor
- GetHashCode
- GetType
- LayDown
- setColor
- Shake
- Sit

The 'Sit' method is highlighted, and a tooltip shows: `void Dog.Sit()`



# Modificadores de Acesso

Também definido como **encapsulamento** de uma classe, definem o quão acessíveis são as propriedades e métodos de uma determinada classe para as outras

## public

- É o modificador de acesso que basicamente diz que pode ser visto por qualquer outra classe
- Precisamos especificar explicitamente se uma variável ou método será público

## private

- É o oposto do public onde "esconde" o método ou variável declarada como private
- Todos os métodos e variáveis são private por padrão



# Get & Set

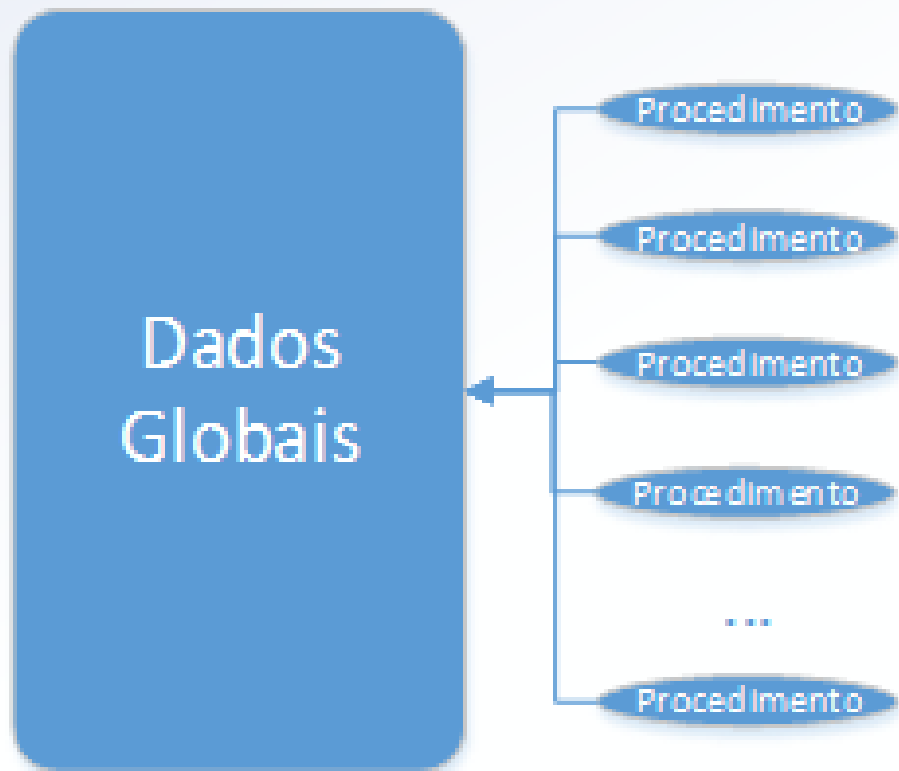
- Não é uma boa prática de programação deixar as variáveis de uma classe expostas como public. Mas então como poderemos acessá-las? Usando funções chamadas Getters & Setters!

```
-referências  
public void setColor(string color)  
{  
    Color = color;  
}  
  
-referências  
public string getColor()  
{  
    return Color;  
}
```



POO é um tipo de **Paradigma de Programação**, ou seja: uma forma de se pensar e organizar os nossos projetos. É a base de todo programa seja ele um jogo ou não!

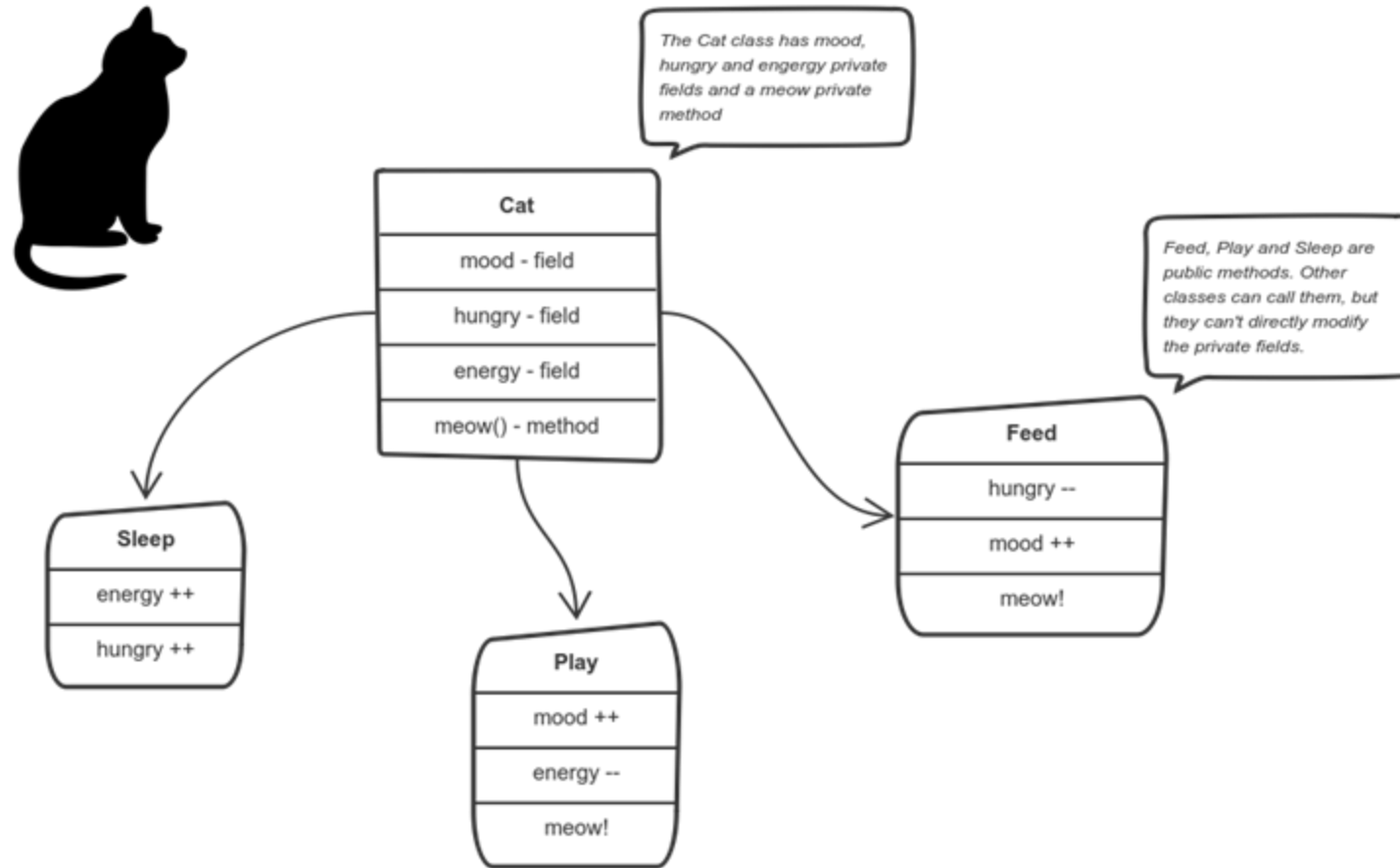
### Programação Estruturada



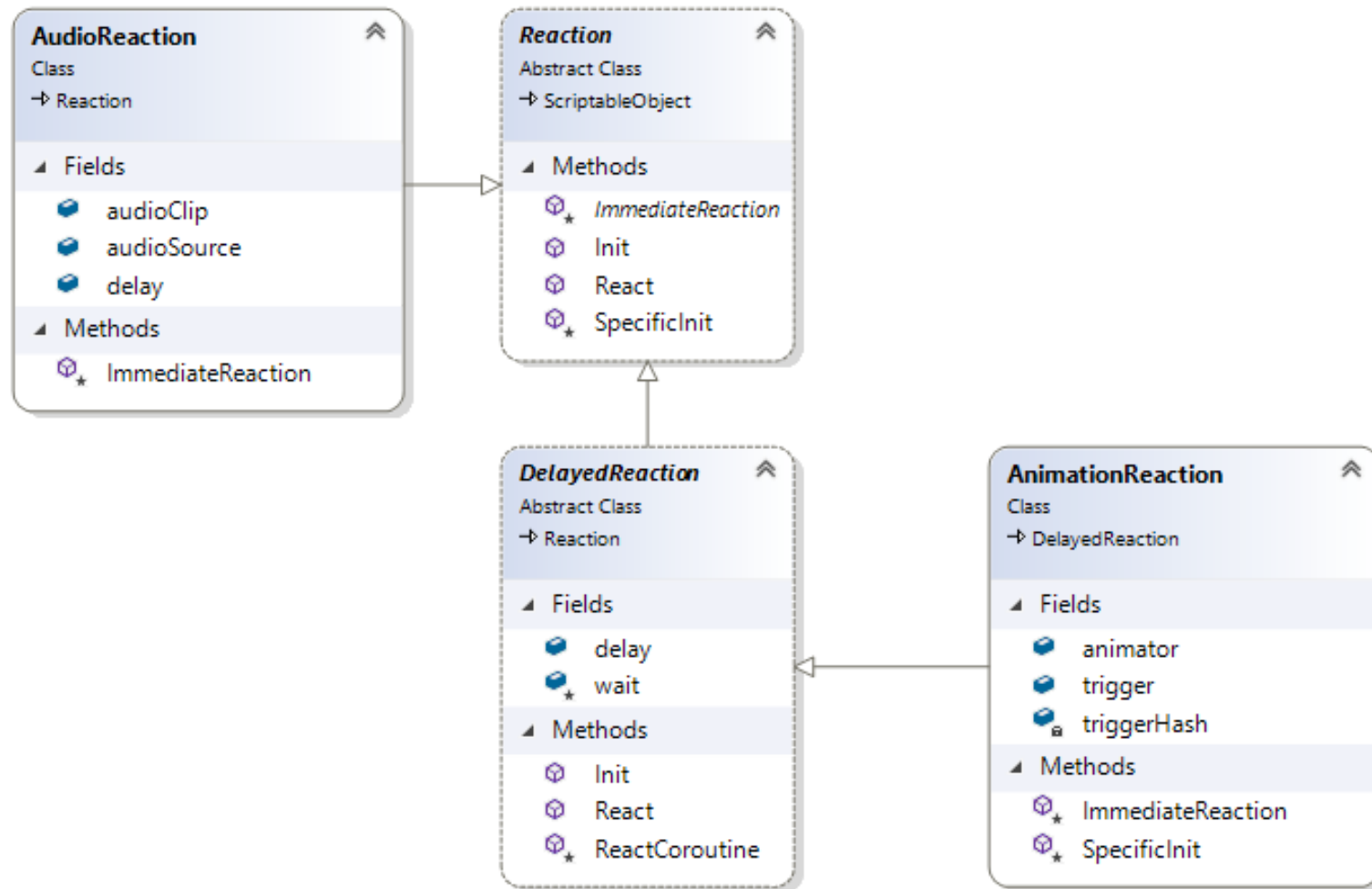
### Programação Orientada a Objetos



# Diagrama de Classes



# Diagrama de Classes





# Modificador `Static`

- Já sabemos que para manipular uma classe precisamos instanciar o seu objeto. Mas e se pudéssemos usa-la sem criar um objeto toda vez? Pois é, essa é a função do modificador `static`.
- Por base, nossas classes não são estáticas ou non-static
- Por ter um comportamento bastante próprio não vamos aborda-lo em aula mas se for do seu interesse, pesquise!
- Programação é pura pesquisa. Não se sinta preso às aulas para aprender mais. Use todas as ferramentas a sua mão para alcançar seus objetivos como desenvolvedor.



