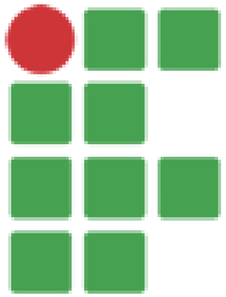


INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Disciplina

Modelagem de Domínio e Conceitos de Orientação a Objetos

PROF. FERNANDO TAMBERLINI ALVES

Paradigma Orientação a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

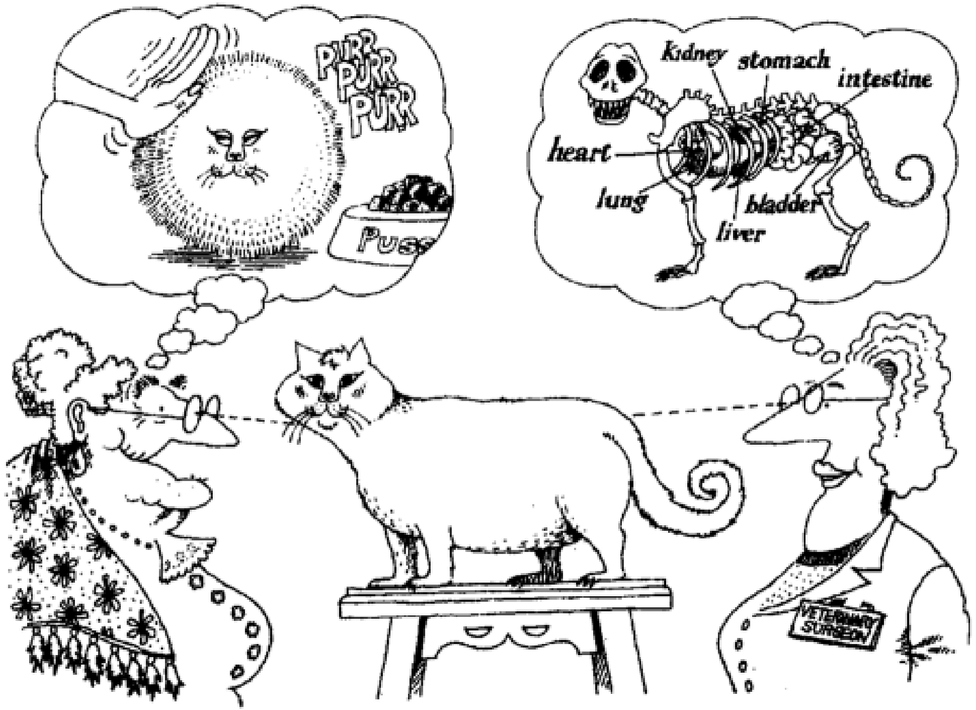
- ✓Abstração
- ✓Encapsulamento
- ✓Herança
- ✓Polimorfismo

Paradigma Orientação a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

✓Abstração

- A representação computacional do objeto real deve se concentrar nas características que são relevantes para a resolução do problema
- São criados somente os atributos e métodos necessários para o problema em mãos

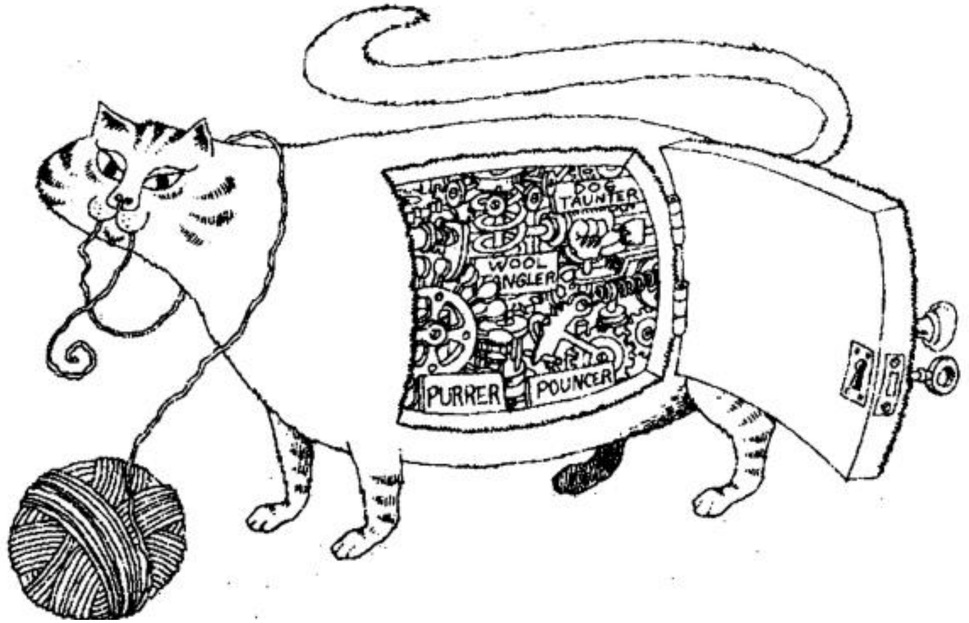


Paradigma Orientação a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

✓Encapsulamento

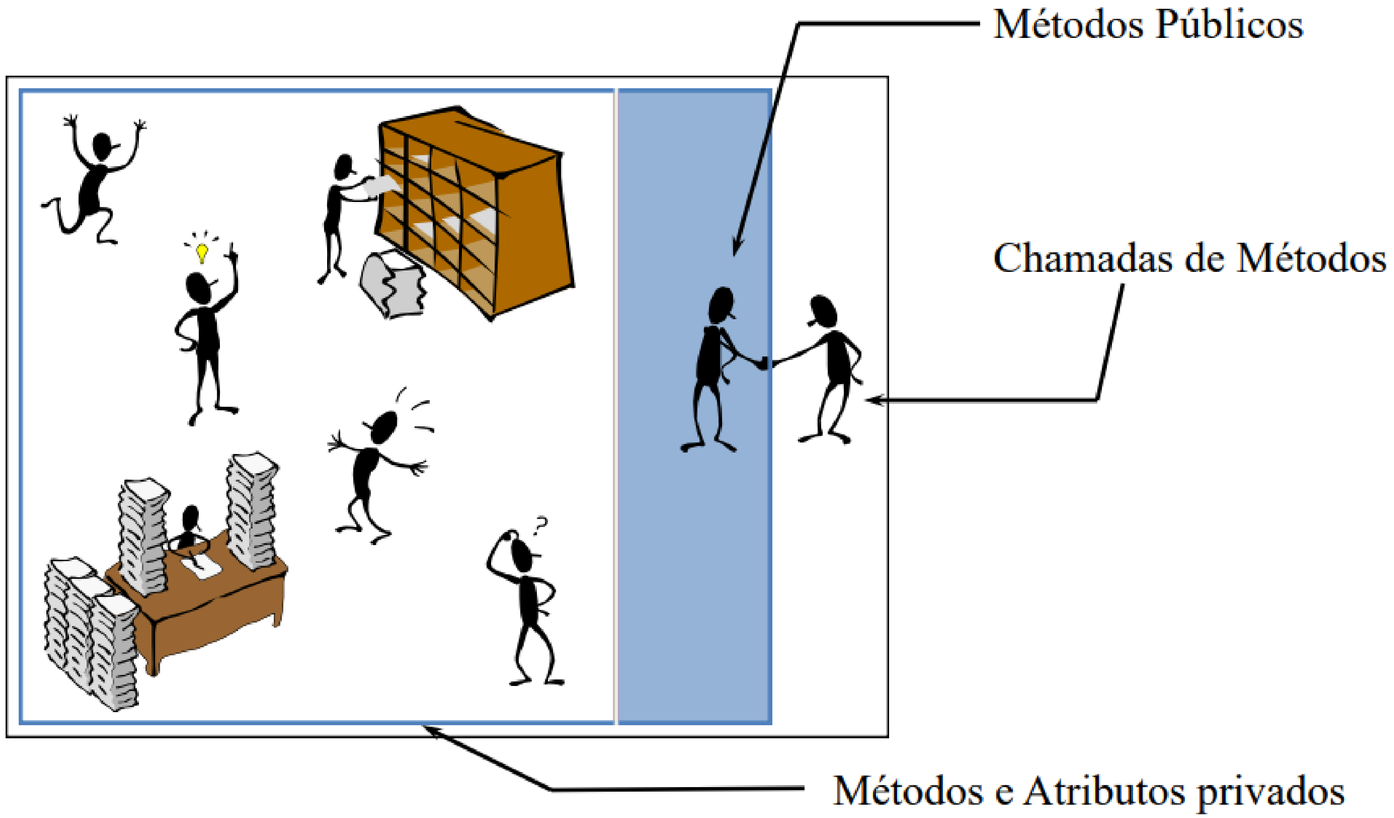
- O objeto deve esconder seus dados e os detalhes de sua implementação
- Os métodos formam uma "cerca" em torno dos atributos
- Os atributos não devem ser manipulados diretamente
- Os atributos somente devem ser alterados ou consultados através dos métodos do objeto



Paradigma Orientação a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- ✓Encapsulamento

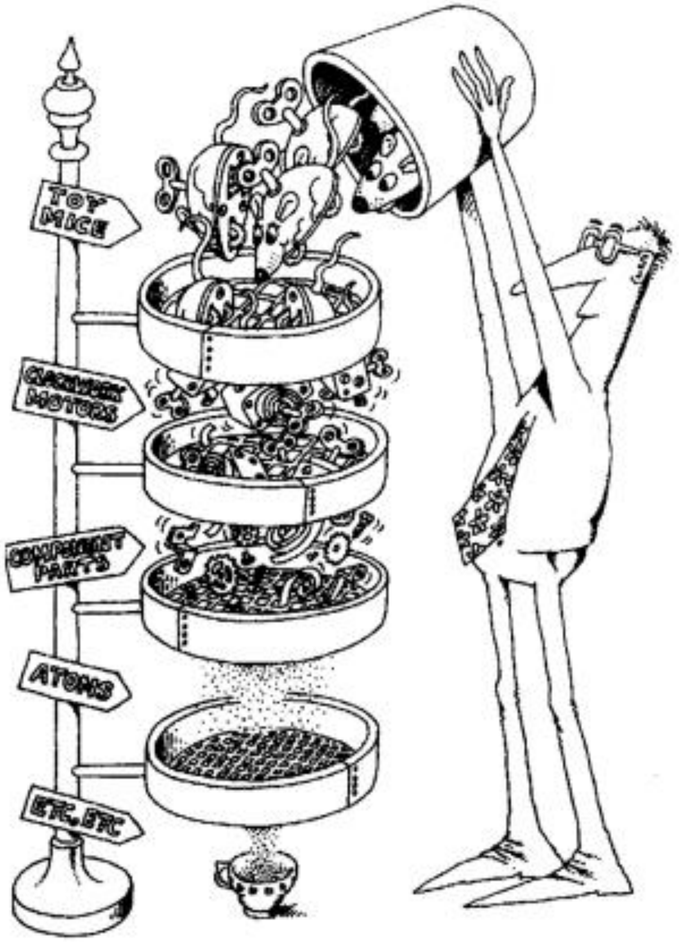


Paradigma Orientação a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

✓Herança

- Os objetos devem ser organizados no sistema de forma hierárquica
- Objetos herdam atributos e métodos dos seus ancestrais na hierarquia



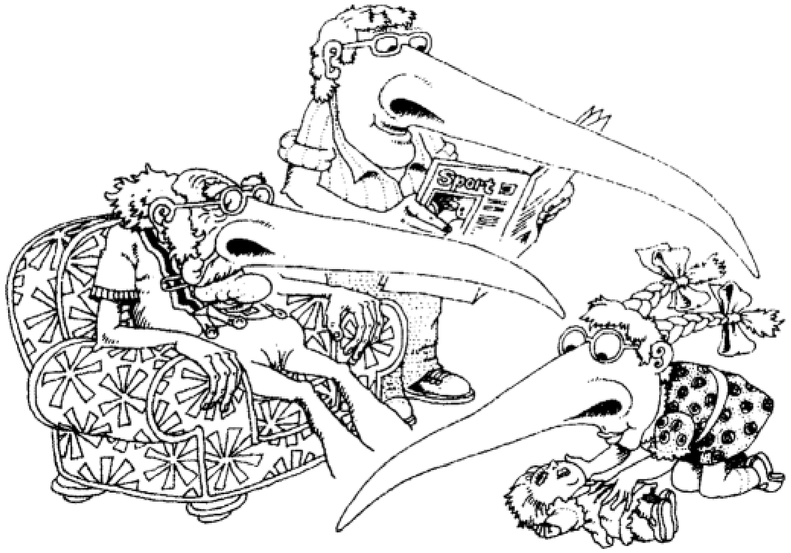
Paradigma Orientação a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

✓Herança

Para viabilizar a hierarquia entre objetos, as classes são organizadas em estruturas Hierárquicas

- A classe que forneceu os elementos herdados é chamada de superclasse
- A classe herdeira é chamada de subclasse
- A subclasse pode herdar os métodos e atributos de suas superclasses
- A subclasse pode definir novos tributos e métodos específicos



Paradigma Orientação a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

✓Polimorfismo

Permite que um objeto possa assumir diferentes formas (ou comportamentos) dependendo do contexto em que está sendo usado.

Isso significa que diferentes objetos podem responder de maneira diferente aos mesmos métodos, permitindo que um programa seja mais flexível e genérico.

Paradigma Orientação a Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

✓Polimorfismo

Uma subclasse pode redefinir (sobrescrever) um método herdado

- Este mecanismo é chamado de polimorfismo
- O polimorfismo se realiza através da recodificação de um ou mais métodos herdados por uma subclasse
- Em tempo de execução, o Compilador/Interpretador saberá qual implementação deve ser usada

Criando Objetos no Javascript

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Criamos um objeto abrindo e fechando chaves;
- Um objeto pode ser uma variável “let” ou uma constante “const”
- Um objeto parece com um array de chave e valor;
- Podemos iniciar um objeto com vários atributos;
- E os atributos pode ser acessados de duas formas:

```
let carro = {
  portas: 4
}
```

```
let carro = {
  portas: 4,
  cor: 'verde'
}

console.log(carro.portas);
console.log(carro['cor']);
```

Atributos e Tipo de Dados

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Atributos podem ser de qualquer tipo de dados;
- Boolean, number, string, array, objeto...

```
let carro = {
  portas: 4,
  cor: "verde",
  opicionais: [
    "teto solar", "blindagem", "ar condicionado"
  ],
  revisado: true
}

console.log(carro.portas);
console.log(carro["cor"]);
console.log(carro.revisado);
console.log(carro.opicionais);
```

Atributos e Tipo de Dados

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- As propriedades podem ter mais de uma palavra;
- Neste caso precisamos colocá-las entre aspas;
- Não é muito utilizado, opte por camelCase;
- É possível acessar um atributo usando uma variável.

```
let carro = {
  portas: 4,
  cor: "verde",
  "tem blindagem": true
}

console.log(carro.portas);
console.log(carro["cor"]);
console.log(carro["tem blindagem"]);
```

```
const robo = {
    bracos: 4,
    pernas: 2,
    arma: 'metralhadora',
    armaEspecial: 'foguete'
}

let a = 'arma';

console.log(robo[a]);
```

Métodos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Os métodos são as operações objetos;
- Os métodos podem recuperar e/ou alterar os dados dos atributos do objeto;
- Os métodos podem realizar qualquer operação que uma função realiza

```
const robo = {
    bracos: 4,
    pernas: 2,
    arma: 'metralhadora',
    armaEspecial: 'foguete',
    atirar: function() {
        console.log('pew pew pew');
    }
}

robo.atirar();
```

```
let pessoa = {
    nome: 'Matheus',
    getNome: function() {
        console.log("O nome é " + this.nome);
    },
    setNome: function(novoNome) {
        this.nome = novoNome;
    }
}

pessoa.getNome();
pessoa.setNome('Teste');
pessoa.getNome();
```

Objetos dentro de Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Como o objeto é também um tipo de dado, podemos ter objetos com objetos dentro;
- Utilizando como um array associativo, por exemplo:
- O objeto não é imutável, ele pode ganhar atributos e métodos ao longo do código;

```
let pessoa = {
    nome: 'Matheus',
    caracteristicas: {
        olhos: 'verdes',
        cabelo: 'castanho',
        brincos: false,
        oculos: false
    }
}

console.log(pessoa.caracteristicas.cabelo);
```

```
let pessoa = {
    nome: 'Matheus',
}

pessoa.idade = 29;

pessoa.falar = function() {
    console.log('Olá');
}

console.log(pessoa);
```

Trabalhando com Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Como é possível adicionar, também podemos excluir atributos dos objetos;
- A palavra reservada this dentro de um objeto, vai se referir a própria instância;
- Podemos utilizar para resgatar as propriedades em um método;

```
let pessoa = {
    nome: 'Matheus',
}

pessoa.idade = 29;

pessoa.falar = function() {
    console.log('Olá');
}

delete pessoa.idade;
delete pessoa.falar;

console.log(pessoa);
```

```
let pessoa = {
    nome: 'Matheus',
}

pessoa.idade = 29;

pessoa.falar = function() {
    console.log('Olá, eu tenho ' + this.idade + ' anos');
}

pessoa.falar();
```

Criando Objetos - Construtor

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Uma outra forma de criar objeto é pela constructor function;
- Uma grande vantagem, é que este método aceita parâmetros para o obj;
- Parecida com as constructor functions, porém não precisamos utilizar o new;
- O objeto é criado com o retorno da função

```
function Pessoa(nome) {
 this.nome = nome;
}

let pessoa1 = new Pessoa('Matheus');
let pessoa2 = new Pessoa('João');

console.log(pessoa1.nome);
console.log(pessoa2.nome);
```

```
function criarPessoa(nome) {
 return {
    nome: nome
 };
}

let pessoa1 = criarPessoa('Matheus');
let pessoa2 = criarPessoa('João');

console.log(pessoa1.nome);
console.log(pessoa2.nome);
```

Construtor - Atributo

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Quando um objeto é criado, sempre uma propriedade constructor é adicionada a ele;
- Contendo a referência de como o objeto foi criado

```
function newObj(x) {
  return {
    x: x
  };
}

let y = newObj(1);

function NewObjTwo(x) {
  this.x = x;
}

let z = new NewObjTwo(2);

console.log(y.constructor);
console.log(z.constructor);
```

Objeto Global

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Sempre que é iniciada uma página web traz um objeto chamado window;
- As variáveis globais são ficam atreladas a ele como propriedades;
- Os métodos da linguagem também podem ser invocados pela window;
- O this no escopo global também referenciado a window;

```
var teste = 'teste';

console.log(teste);
console.log(window.teste);
console.log(this.teste);
```

O operador instanceof

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Uma maneira de saber de qual instância (pai) vem algum objeto;
- Mais prático que utilizar o constructor;

```
function Robo(nome, arma) {
  this.nome = nome;
  this.arma = arma;
}

function Humano(nome) {
  this.nome = nome;
}

let android = new Robo('Xyz', 'Punhos');

console.log(android instanceof Robo);
console.log(android instanceof Humano);
```

Passando referência de objeto

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Quando você atribui um obj já criado para uma outra variável, você só está passando uma referência;
- Se alterar a referência, o original também é alterado;

```
let obj = {
  teste: 1,
}

let copia = obj;

copia.teste = 2;

console.log(obj.teste);
```

Comparando objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Você só consegue ter objetos iguais, criando uma referência;
- Objetos criados a partir de um construtor, sempre serão diferentes;

```
function Robo(nome, arma) {
  this.nome = nome;
  this.arma = arma;
}

let robo1 = new Robo('teste', 'revolver');
let robo2 = new Robo('teste', 'revolver');

console.log(robo1 === robo2);

let robo3 = robo1;

console.log(robo1 === robo3);
```

Object literals

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Função do ES6, que permite criar objetos mais rapidamente;
- Utilizando nomes de variáveis para nomes de propriedades;
- Também não precisamos declarar function para métodos no ES6
- Podemos também criar propriedades com variáveis ou retorno de funções;
- Ajudando a escrever menos código;

```
let x = 1;
let y = 2;

let obj = {x,y};

console.log(obj.x);
```

```
let megazord = {
  nome: 'Megazord',
  arma: 'espada laser',
  explodirTudo() {
    console.log("BOOM!");
  }
}

megazord.explodirTudo();
```

```
let tipo = 'tipo_de_';

let carro = {
  [tipo+"carro"]: "SUV"
}

let barco = {
  [tipo="barco"]: "Iate"
}

console.log(carro.tipo_de_carro);
```

Copiando Atributos e Comparando Objetos

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Os objetos herdam métodos do objeto pai Object, e podemos utilizá-los;
- Para copiar propriedades utilizamos o método assign;
- Podemos comparar os objetos com o método is
- Teremos basicamente os mesmos resultados de ===
- Salvo para NaN com NaN e +0 com -0 (que neste método são considerados como iguais)

```
let robo1 = {
  arma: 'lança granada'
}

let robo2 = {

}

Object.assign(robo2, robo1);

console.log(robo2);
```

```
console.log(Object.is(robo1, robo2));

robo3 = robo1;

console.log(Object.is(robo1, robo3));
```

Destructuring

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Um outro recurso que veio com o ES6 trazendo algumas funcionalidades;
- Podemos criar várias variáveis com uma linha só, desestruturando um objeto;
- Há também a possibilidade de utilizar o destructuring para mudar o valor de variáveis já criadas;
- O destructuring funciona com arrays também

```
let config = {
  ip: '127.0.0.1',
  port: '80',
  blocked: true,
}

let {ip, port, blocked} = config;

console.log(ip);
console.log(port);
console.log(blocked);
```

```
let config = {
  ip: '127.0.0.1',
  port: '80',
  blocked: true,
}

let ip = '192.168.0.60';
let port = '8000';

({ip, port} = config);

console.log(ip, port);
```

```
let numeros = [1,2,3];

const [num1,num2,num3] = numeros;

console.log(num1,num2,num3);
```

Destructuring e rest operator

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- O rest operator é utilizado quando não sabemos quantos argumentos virão para o destructuring;
- Podendo criar um array com um tamanho infinito, com os restos dos elementos passados;

```
let [a, ...b] = [1,2,3,4,5,6,7,8];

console.log(a, b);
```

Criando Classes

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Criando a classe Triangulo

```
class Triangulo{
    constructor(base,altura){
        this.b = base;
        this.h = altura;
    }
    calcularArea(){
        return this.b*this.h/2;
    }
}

const triangulo6 = new Triangulo(5,8);
```

Criando Classes

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Criando a classe TrianguloV com parâmetros privados

```
class TrianguloV{
    #b;
    #h;
    constructor(base,altura){
        this.b = base;
        this.h = altura;
    }
    calcularArea(){
        return this.b*this.h/2;
    }
    getBase(){
        return this.#b
    }
    getAltura(){
        return this.#h
    }
}
```

Criando Classes

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

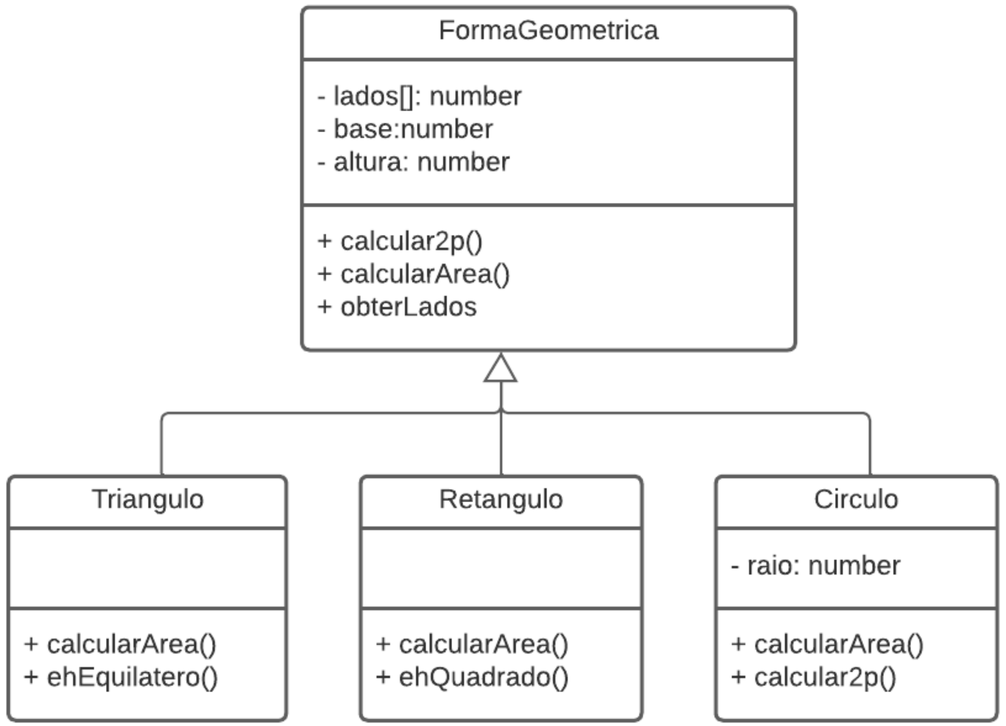
Criando a subclasse TrianguloSub que é filha da classe Forma.

```
class TrianguloSub extends Forma{
    #b;
    #h;

    constructor(lados,base,altura){
       super(lados);
       this.#b=base;
       this.#h=altura;
    }
    calcularArea(){
        return this.#b*this.#h/2;
    }

}
```

Exercício Prático

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

- Criar as classes descritas abaixo com a implementação dos seus métodos:



Exercício Prático

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Criando a classe FormaGeometrica

```
class FormaGeometrica{
    #lados;
    constructor(vetorLados){
        this.#lados = vetorLados;
    }
    calcular2p(){
        let perimetro=0
        for(let i=0;i<this.#lados.length;i++)
            perimetro+=this.#lados[i];
        return perimetro;
    }
    calcularArea(){
        return this.#lados[0]*this.#lados[0]
    }

    obterLados(){
        return this.#lados
    }
}
```

Dúvidas?

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Rio de Janeiro

Dúvidas?

fernando.alves@ifrj.edu.br

ftamberlini.dev.br

Referências:

- **Notas de Aula do Prof. Leonardo Murta – UFF**
- **Livro "Object-Oriented Analysis and Design with Applications"**