

Antes de  
iniciar...

Dúvidas ???

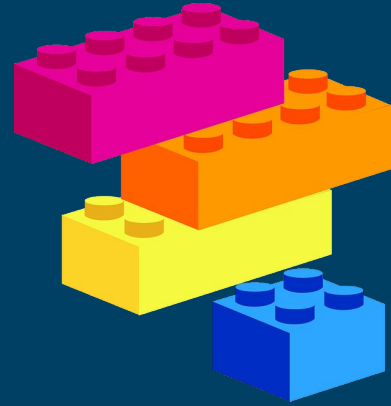


# Programação Orientada a Objetos

POO  
Introdução

# Modularização

Antes de POO...

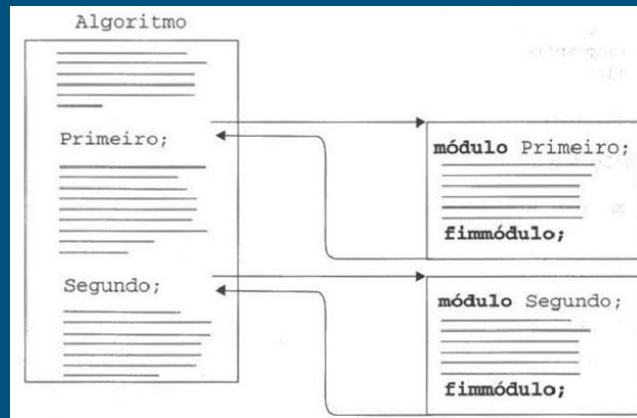


## O que é ?

- Modularizar é dividir um programa em Sub-rotinas, chamadas de Procedimentos ou Funções. Estes são blocos de programa que executam determinada tarefa.

\*\*\* Na Programação Orientada a Objetos chamamos Métodos

# O que é ?



emerson@paduan.pro.br

# Para quê ?

Algumas vantagens:

- Dividir e estruturar o problema em pequenas partes para facilitar o desenvolvimento;
- Evitar repetição de código em vários locais;
- Facilitar a localização e correção de problemas;
- Facilitar a manutenção do código;

emerson@paduan.pro.br

# Sintaxe básica

```
Tipo-de-retorno NomeDoMétodo ( lista de parâmetros ) {  
    //corpo do módulo  
    retorno  
}
```

- **Nome do método:** Valem as regras de nome de variáveis
- **Lista de parâmetros:** Opcional. Tipos e nomes das variáveis que o método irá receber.
- **Corpo:** Instruções que realizam a operação pretendida.
- **Tipo de Retorno:** Valor que o método retorna caso exista.

emerson@paduan.pro.br

# Praticando

Entendendo por meio de exemplos:

```
void linha();
```

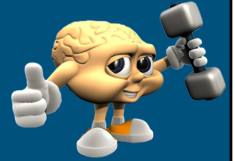
```
void linha(int );
```

```
void linha(int, char);
```

```
int soma (int, int);
```

emerson@paduan.pro.br

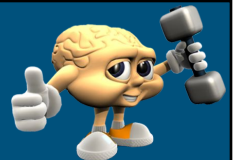
# Exercício 1



Escreva um método em Java que receba um número inteiro fornecido como parâmetro, e retorne se o número é par ou não.

[emerson@paduan.pro.br](mailto:emerson@paduan.pro.br)

# Exercício 2



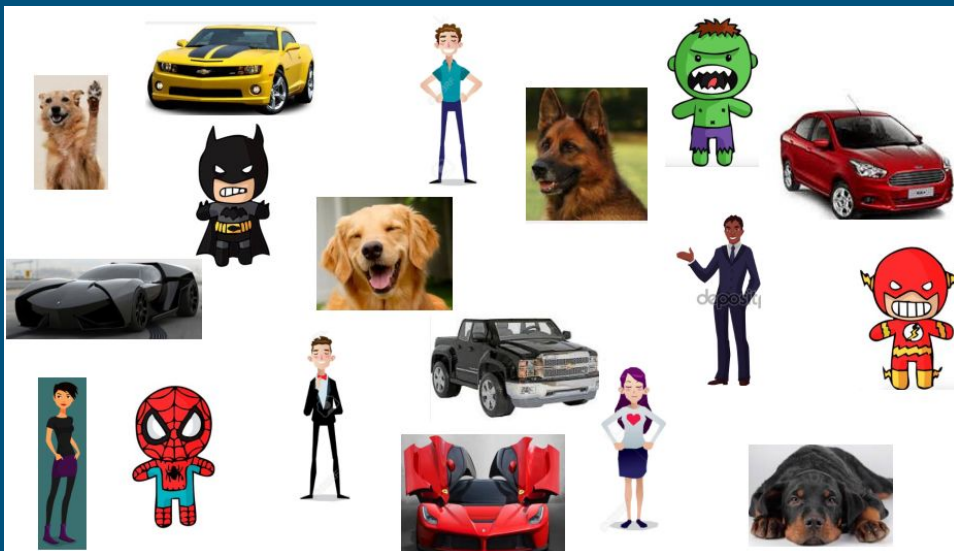
Escreva um método em Java que retorne o menor entre três números inteiros fornecidos como parâmetros.

[emerson@paduan.pro.br](mailto:emerson@paduan.pro.br)

# POO

Programação Orientada a Objetos

emerson@paduan.pro.br



Como organizar ?

emerson@paduan.pro.br



Características comuns

[emerson@paduan.pro.br](mailto:emerson@paduan.pro.br)

## Classe

Uma classe é um modelo que define, especifica um objeto. É uma abstração (representação) dos objetos.

Ela define os dados (**ATRIBUTOS**) e os comportamentos (**MÉTODOS**) do Objeto.

[emerson@paduan.pro.br](mailto:emerson@paduan.pro.br)

# Exemplo



emerson@paduan.pro.br

## ATENÇÃO

Uma classe é um **MODELO!**

Não se coloca dados ou se utiliza diretamente uma classe.

É necessário criar um objeto a partir da classe



emerson@paduan.pro.br



# Exemplo

TODOS os objetos criados a partir da classe, possuem os mesmos atributos e métodos, mas com valores diferentes.



emerson@paduan.pro.br

# Exemplo

```
class Pessoa {  
    String nome;  
  
    void apresentar(){  
        System.out.println("Olá! Eu sou " + nome);  
    }  
}
```

```
class Exemplo{  
    main () {  
  
        Pessoa p = new Pessoa();  
  
        p.nome = "Emerson";  
        p.apresentar();  
    }  
}
```

emerson@paduan.pro.br

# Construtores

São métodos especiais utilizados para inicialização dos atributos de um objeto no “momento” da criação do objeto.

## Detalhes:

Os construtores podem ter ou não parâmetros

O nome do construtor DEVE ser o mesmo da classe

Construtores NÃO possuem valor de retorno

emerson@paduan.pro.br

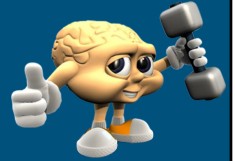
# Exemplo

```
public class Pessoa {  
    String nome;  
    float salario;  
  
    public Pessoa(String nome, float salario){  
        this.nome = nome;  
        this.salario = salario;  
    }  
  
    public void exibir(){  
        System.out.println("Pessoa: " + nome + ": R$ " + salario);  
    }  
}
```

```
class Exemplo{  
    main () {  
  
        Pessoa p = new Pessoa("Marcos", 5000 );  
  
        p.exibir();  
    }  
}
```

emerson@paduan.pro.br

## Exercício 3



Crie a classe veículo, com os atributos modelo, marca e consumo (quantos km/l).

Faça um construtor para inicializar os atributos da classe.

Escreva um método para exibir os dados do carro (modelo, marca) e outro para retornar o valor do consumo.

Faça o main para testar a classe criada.

[emerson@paduan.pro.br](mailto:emerson@paduan.pro.br)

## Encapsulamento



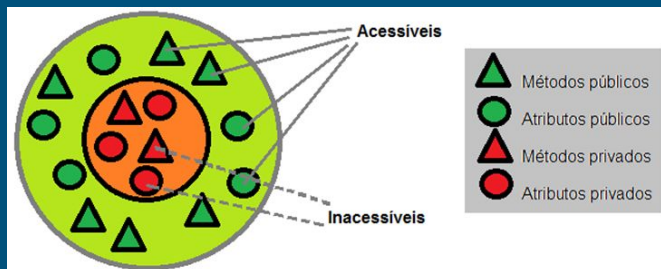
"Esconder / Proteger"

[emerson@paduan.pro.br](mailto:emerson@paduan.pro.br)

# Conceito

É necessário garantir que os dados (atributos) dos objetos não sejam modificados de forma indevida.

A modificação de forma indevida interfere no funcionamento do objeto e pode gerar erros.



emerson@paduan.pro.br

# Modificadores

Modificador	Funcionalidade
<b>public</b>	permite que qualquer outra parte da aplicação tenha acesso ao membro
<b>Padrão (default)</b>	membros que não foram marcados com nenhum modificador explicitamente. Só podem ser acessados por outras classes dentro do mesmo pacote
<b>protected</b>	os membros são acessíveis por classes dentro do mesmo pacote e por classes derivadas (mesmo em pacotes diferente)
<b>private</b>	só é acessível dentro da própria classe em que foi declarado

emerson@paduan.pro.br

## get / set

Em Java utilizamos nomes de métodos iniciando com get ou set para indicar métodos que alteram ou obtêm valores de atributos privados.

Exemplo:

getHora: obtém o valor do atributo hora

setHora: altera o valor do atributo hora

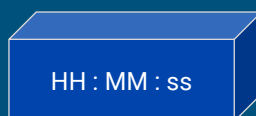
emerson@paduan.pro.br

## Exercício 4 - Relógio

Escreva a classe Relógio, com os atributos hora, minuto e segundo, e com um construtor que recebe horas, minutos e segundos para inicializar o relógio. Faça um método da classe para exibir a hora atual. Crie os get's e set's para os atributos.

Faça um programa (main) que crie um objeto do tipo Relógio e exiba a hora atual do relógio.

O relógio não pode aceitar valores inválidos



**Relógio**

emerson@paduan.pro.br