



# Java Orientado a Objetos

**RECODE**  
**pro**



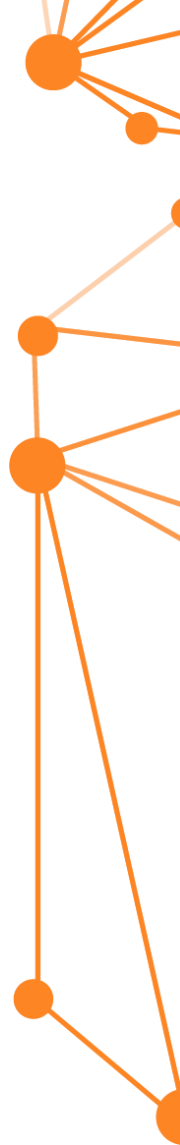
# Módulo 03 – Java Orientado a Objetos Aula 02



## Programação Orientada a Objetos em Java

A orientação a objetos, também conhecida como Programação Orientada a Objetos (POO) ou Object-Oriented Programming (OOP) é um paradigma de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos.

A programação orientada a objetos é uma forma especial de programar, mais próximo de como expressaríamos as coisas na vida real. Um objeto é uma coisa material ou abstrata que pode ser percebida pelos sentidos e descrita por meio das suas características, comportamentos e estado atual.



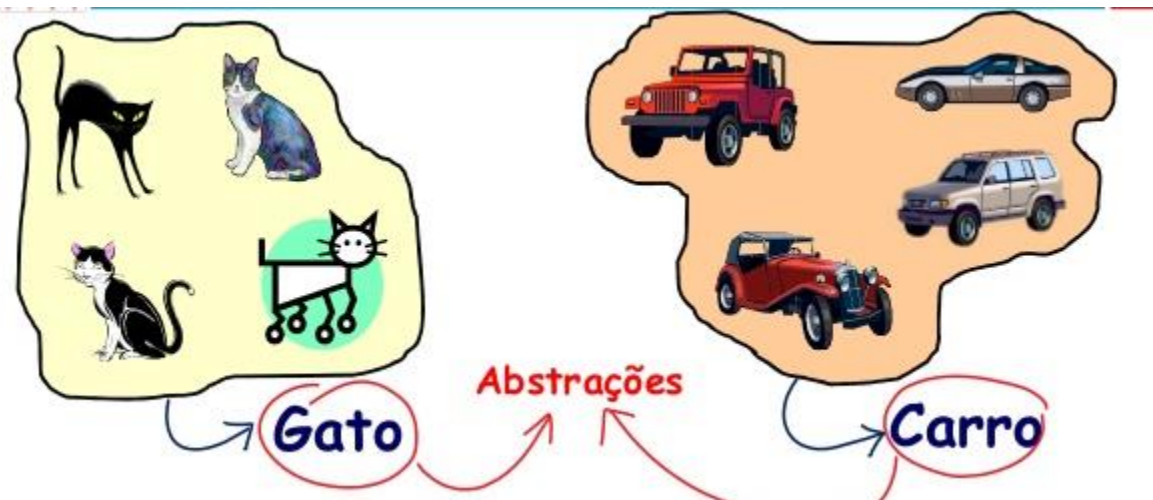
# Módulo 03 – Java Orientado a Objetos Aula 02



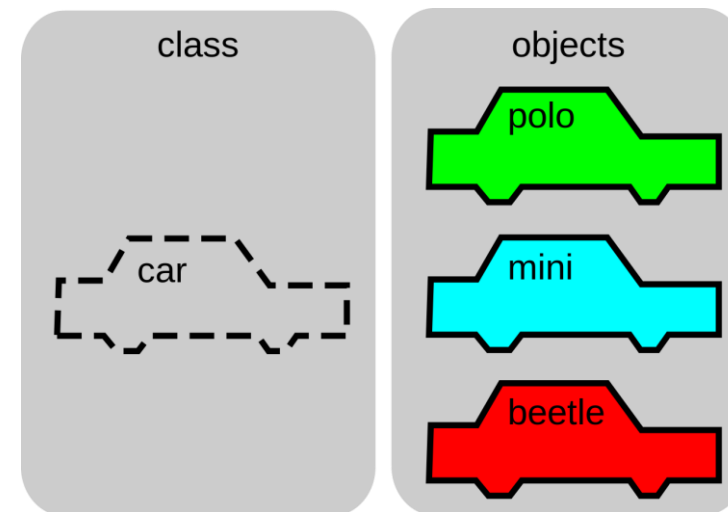
## Programação Orientada a Objetos em Java

### Abstração

É a habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. Em modelagem orientada a objetos, uma classe é uma abstração de entidades existentes no domínio do sistema de software.



Abstração é um processo de ocultar os detalhes de implementação e mostrar apenas funcionalidades para o usuário.

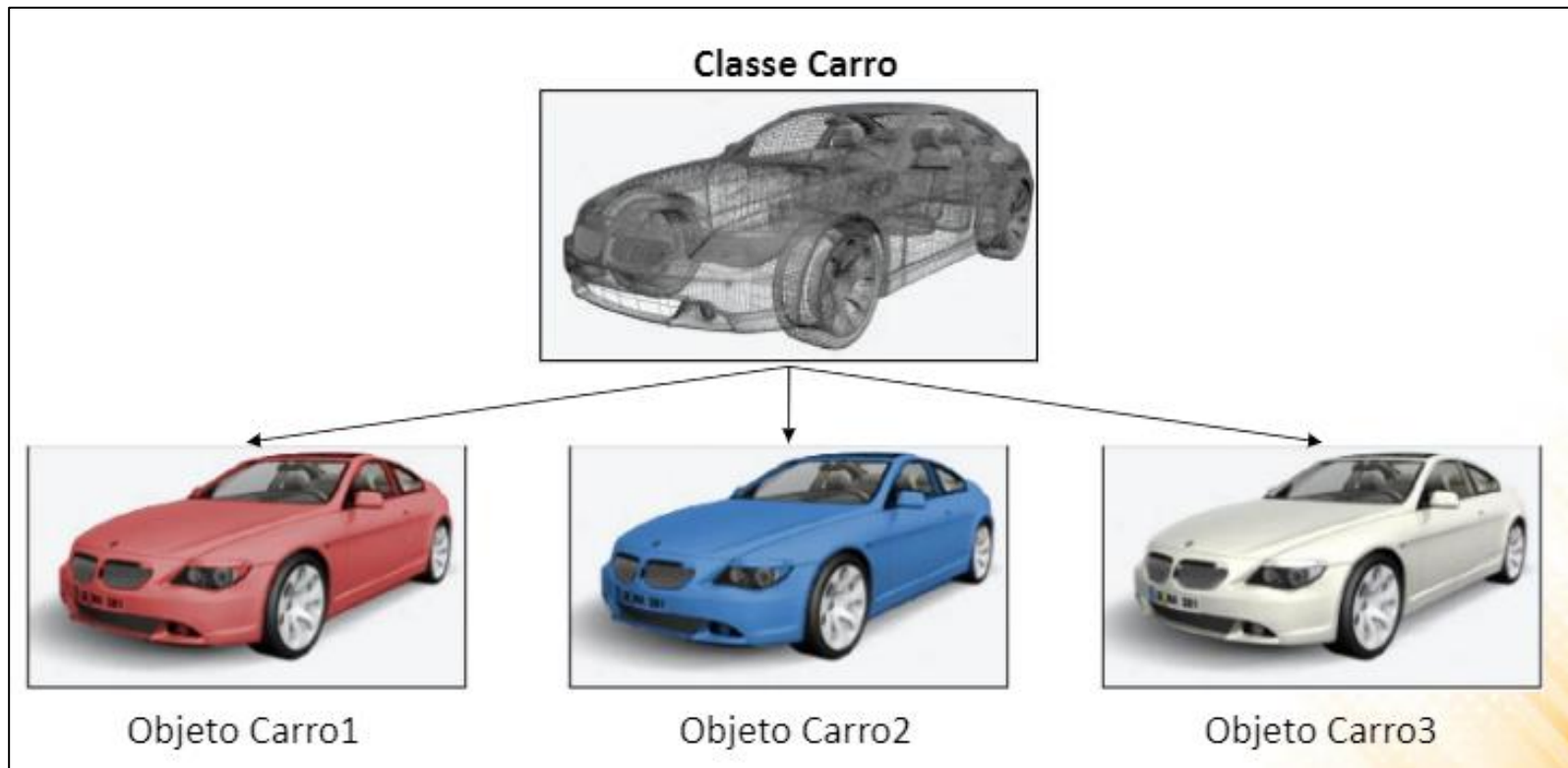


Fonte:  
<https://morettic.com.br/wp2/poo/poo-classes-abstratas/>

# Módulo 03 – Java Orientado a Objetos Aula 02



Exemplo de classes e suas instancias (objetos)



# Módulo 03 – Java Orientado a Objetos Aula 02



```
public class Carro {  
  
    String modelo;  
    String chassi; // atributos da classe Carro  
    int qtdPortas;  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public void setModelo(String modelo) {  
        this.modelo = modelo;  
    }  
  
    public String getChassi() {  
        return chassi;  
    }  
    public void setChassi(String chassi) {  
        this.chassi = chassi;  
    }  
  
    public int getQtdPortas() {  
        return qtdPortas;  
    }  
}
```

Como ficaria a implementação da classe em Java??

```
    public void setQtdPortas(int qtdPortas) {  
        this.qtdPortas = qtdPortas;  
    }  
  
    public void acelerar() {  
        // código do método acelerar aqui  
    }  
  
    public void frear() {  
        // código do método frear aqui  
    }  
}
```

# Módulo 03 – Java Orientado a Objetos Aula 02



## Instancia – cópia de um objeto baseado em sua classe modelo:

Por exemplo, você cria uma classe Cachorro. Você criou apenas a ideia de um cachorro, mas para poder usar efetivamente, vai ter que criar uma instância desse cachorro (um objeto).

***Cachorro rex = new Cachorro(azul);***

### Objetos (instancias) da classe Carro:

```
Carro carro1 = new Carro();  
carro1.modelo = "Gol";  
carro1.qtdPortas = 4;  
carro1.chassi = "9c2xx250xxx003931";  
carro1.acelerar();  
carro1.frear();
```

```
Carro carro2 = new Carro();  
carro2.modelo = "Fusca";  
carro2.qtdPortas = 2;  
carro2.chassi = "7c2yy255xxy002225";  
carro2.acelerar();  
carro2.frear();
```

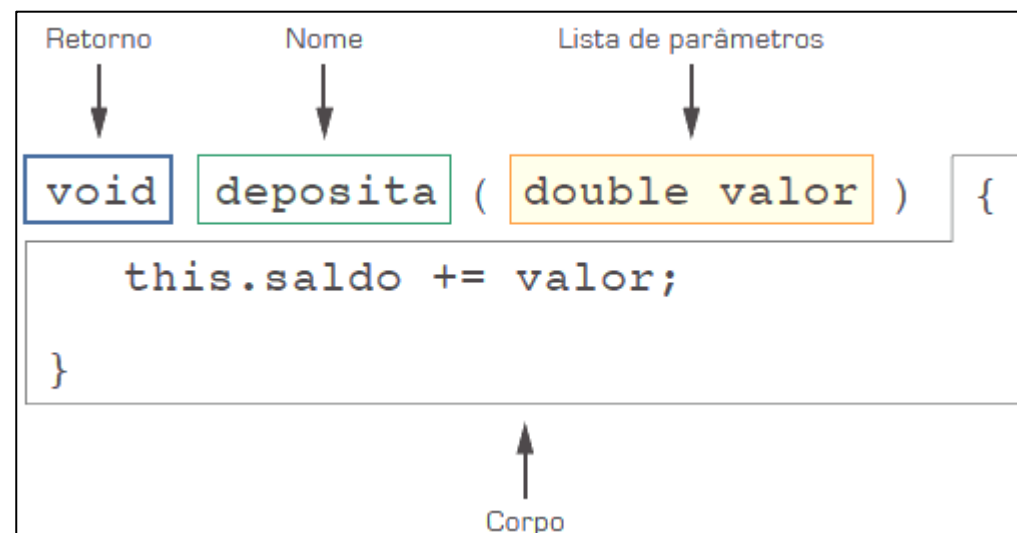
# Módulo 03 – Java Orientado a Objetos Aula 02



## Métodos:

São funções que realizam tarefas específicas e pode ser chamado por qualquer outro método ou classe, para realizar a referida função num determinado contexto. Os métodos possuem algumas características como:

- Podem ou não retornar um valor;
- Podem ou não aceitar argumentos (valores de entrada e de saída dos métodos);
- Após encerrar sua execução, o método retorna o fluxo de controle do programa para quem o chamou.



# Módulo 03 – Java Orientado a Objetos Aula 02



Para fixar nossos conhecimentos em Classe, Atributos e métodos em Java, vamos implementar uma classe que represente um caixa com dois métodos: sacar e depositar, esse modelo deve ser construído com um saldo inicial de R\$ 1000 e atualizado de acordo com o método chamado.

```
public class Caixa {  
  
    public double saldo = 1000;  
  
    void sacar(double valor) {  
        this.saldo = saldo - valor;  
    }  
  
    void depositar(double valor) {  
        this.saldo = saldo + valor;  
    }  
  
    double exhibirSaldo() {  
        return this.saldo;  
    }  
}
```

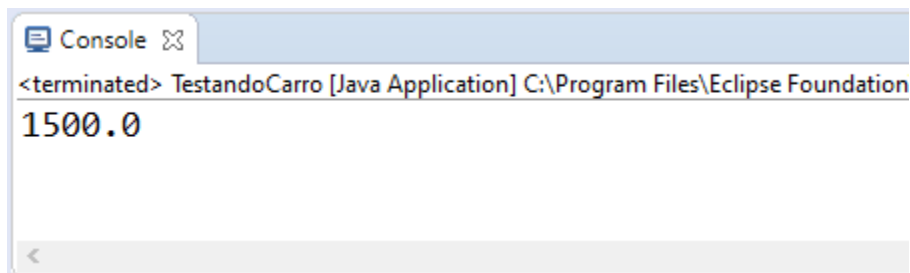


# Módulo 03 – Java Orientado a Objetos Aula 02



Dentro do método principal **public static void** main(String[] args) criamos um objeto do tipo Caixa, fizemos um depósito de R\$ 500 e chamamos o método `exibirSaldo()` para mostrar o valor atualizado do atributo saldo que passa a ser de R\$ 1500.0 conforme código e figura abaixo.

```
Caixa caixa = new Caixa(); // criando objeto da classe
caixa.depositar(500); // chamando o método depositar
System.out.println(caixa.exibirSaldo());
```

A screenshot of a Java IDE console window. The title bar says "Console". The text inside the console reads: "<terminated> TestandoCarro [Java Application] C:\Program Files\Eclipse Foundation" followed by "1500.0" on the next line. There is a scroll bar on the right side of the console window.

```
<terminated> TestandoCarro [Java Application] C:\Program Files\Eclipse Foundation
1500.0
```

Saída no console:

# Módulo 03 – Java Orientado a Objetos Aula 02



## Encapsulamento, métodos assessores Get e Set,

Uma das ideias mais importantes da orientação a objetos é o encapsulamento. Encapsular significa esconder a implementação dos objetos. O encapsulamento favorece principalmente dois aspectos de um sistema: a manutenção e o desenvolvimento.

A manutenção é favorecida pois, uma vez aplicado o encapsulamento, quando o funcionamento de um objeto deve ser alterado, em geral, basta modificar a classe do mesmo.

**Exemplo:** interface do celular que encapsula todas as funcionalidades internas sem acesso do usuário, operações de um caixa eletrônico que encapsula todas as atividades que o cliente faz no equipamento, entrada do cartão, operação escolhida e todo o restante do processo

# Módulo 03 – Java Orientado a Objetos Aula 02



## Níveis de visibilidade

No Java, há quatro níveis de visibilidade: privado, padrão, protegido e público. Podemos definir os níveis privado, protegido e público com os modificadores `private`, `protected` e `public` respectivamente. Quando nenhum modificador de visibilidade é utilizado o nível default é aplicado.

**Privado (private):** O nível privado é aplicado com o modificador `private`. O que pode ser privado? Atributos, construtores, métodos, classes aninhadas ou interfaces aninhadas.

**Padrão (default):** O nível padrão é aplicado quando nenhum modificador é utilizado. O que pode ser padrão? Atributos, construtores, métodos, classes de todos os tipos e interfaces de todos os tipos

**Protegido (protected) :** O nível protegido é aplicado como modificador `protected`. O que pode ser protegido? Atributos, construtores, métodos, classes aninhadas ou interfaces aninhadas.

**Público (public) :** O nível público é aplicado quando o modificador `public` é utilizado. O que pode ser público? Atributos, construtores, métodos, classes de todos os tipos e interfaces de todos os tipos. Os itens em nível de visibilidade público podem ser acessados de qualquer lugar do código da aplicação.

# Módulo 03 – Java Orientado a Objetos Aula 02

Vamos para um exemplo em Java mostrando os métodos de acesso



## GET e SET

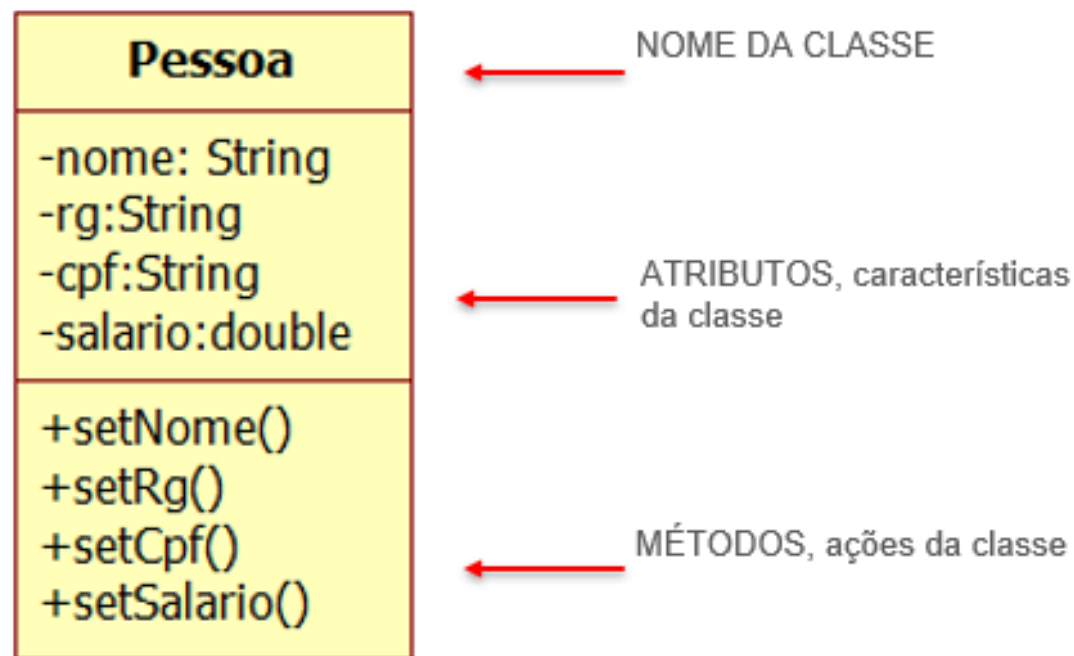
```
public class Cliente {  
    private String nome;  
    private String email;  
    private Date dataNasc;  
  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
  
    public String getEmail() {  
        return email;  
    }  
}
```

```
        public void setEmail(String email) {  
            this.email = email;  
        }  
  
        public Date getDataNasc() {  
            return dataNasc;  
        }  
  
        public void setDataNasc(Date dataNasc) {  
            this.dataNasc = dataNasc;  
        }  
    }  
}
```

# Módulo 03 – Java Orientado a Objetos Aula 02



**Exercício 01** – transforme o modelo de classe abaixo em códigos Java.



# Módulo 03 – Java Orientado a Objetos Aula 02



## Resolução exercício 01

```
public class Pessoa {  
  
    private String Nome;  
    private String RG;  
    private String CPF;  
    private double Salario;  
  
    public void setNome(String nome) {  
        this.Nome = nome;  
    }  
    public void setRG(String rg) {  
        this.RG = rg;  
    }  
    public void setCPF(String cpf) {  
        this.CPF = cpf;  
    }  
    public void setSalario(double salario) {  
        this.Salario = salario;  
    }  
}
```

# Módulo 03 – Java Orientado a Objetos Aula 02



## Exercício 02

Fazer um programa para ler os dados de um produto em estoque (nome, preço e quantidade no estoque). Em seguida:

- Mostrar os dados do produto (nome, preço, quantidade no estoque, valor total no estoque)
- Realizar uma entrada no estoque e mostrar novamente os dados do produto
- Realizar uma saída no estoque e mostrar novamente os dados do produto

Para resolver este problema, você deve criar uma CLASSE conforme projeto ao abaixo:

Product
- Name : string - Price : double - Quantity : int
+ TotalValueInStock() : double + AddProducts(quantity : int) : void + RemoveProducts(quantity : int) : void

# Módulo 03 – Java Orientado a Objetos Aula 02



## Resolução exercício 02

```
public class Product {  
    public String name;  
    public double price;  
    public int quantity;  
  
    public double totalValueInStock() {  
        return price * quantity;  
    }  
  
    public void addProducts(int quantity) {  
        this.quantity += quantity;  
    }  
  
    public void removeProducts(int quantity) {  
        this.quantity -= quantity;  
    }  
  
    public String toString() {  
        return name + ", $ " + String.format("%.2f", price) + ", " +  
quantity + " units, Total: $ "  
                                + String.format("%.2f",  
totalValueInStock());  
    }  
}
```



# Módulo 03 – Java Orientado a Objetos Aula 02

Resolução Exercício 02 PRODUTO Aluno : Lucas Emanuel Santana Dos Santos



```
public class Produto {  
    private String nome;  
    private double preco;  
    private int quantidade;  
    public Produto(String nome, double preco, int qtdEstoque) {  
        this.nome = nome;  
        this.preco = preco;  
        this.quantidade = qtdEstoque;  
    }  
  
    public String getNome() {  
        return nome;  
    }  
    public double getPreco() {  
        return preco * quantidade;  
    }  
    public int getQtdProduto() {  
        return quantidade;  
    }  
    public void setQuantidade(int pQuantidade) {  
        quantidade = pQuantidade;  
    }  
}
```

**Class Produto**

# Módulo 03 – Java Orientado a Objetos Aula 02

Resolução Exercício 02 PRODUTO Aluno : Lucas Emanuel Santana Dos Santos



```
import java.util.ArrayList;
import java.util.List;

public class Estoque {

    private static List<Produto> estoque = new ArrayList<>();

    public static List<Produto> getEstoque() {
        return estoque;
    }

    public static void setEstoque(Produto pProduto) {
        estoque.add(pProduto);
    }

}
```

**Class Estoque**

# Módulo 03 – Java Orientado a Objetos Aula 02

Resolução Exercício 02 PRODUTO Aluno : Lucas Emanuel Santana Dos Santos



```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {

        Scanner ler = new Scanner(System.in);
        String loop;
        String nomeProduto;
        double precoProduto;
        int qtdProduto;

        do{
            System.out.print("Coloque o nome do produto: ");
            nomeProduto = ler.next();

            System.out.print("Coloque o preço do produto: ");
            precoProduto = ler.nextDouble();

            System.out.print("Coloque a quantidade do produto: ");
            qtdProduto = ler.nextInt();

            Produto produto = new Produto(nomeProduto,precoProduto,qtdProduto);

            Estoque.setEstoque(produto);
```

**Class Main**

# Módulo 03 – Java Orientado a Objetos Aula 02

Resolução Exercício 02 PRODUTO Aluno : Lucas Emanuel Santana Dos Santos



```
System.out.print("Quer colocar outro produto ao estoque? (S/N)");
    loop = ler.next();

    }while(loop.equals("s"));

    System.out.println("Mostrando estoque: ");
    System.out.println(" ");
    for (int i = 0; i < Estoque.getEstoque().size(); i++) {

        System.out.println("Id: " + (i + 1));
        System.out.println("Nome: " + Estoque.getEstoque().get(i).getNome());
        System.out.println("Preço: " + Estoque.getEstoque().get(i).getPreco());
        System.out.println("Quantidade:
" + Estoque.getEstoque().get(i).getQtdProduto());
        System.out.println("=====");

    }

}

}
```

**Class Main**

# RECODE



[www.recode.org.br](http://www.recode.org.br)

**RECODE**  
**pro**

[recodepro.org.br](http://recodepro.org.br)

Institucional



[/rederecode](#)



[/recoderede](#)