CURSO TECNICO EM INFORMÁTICA
DISCIPLINA: LINGUAGEM DE PROGRAMAÇÃO

# ORIENTAÇÃO A OBJETOS

Vídeo

Prof. André Aparecido da Silva.
Disponível em: http://www.oxnar.com.br/aulas/java

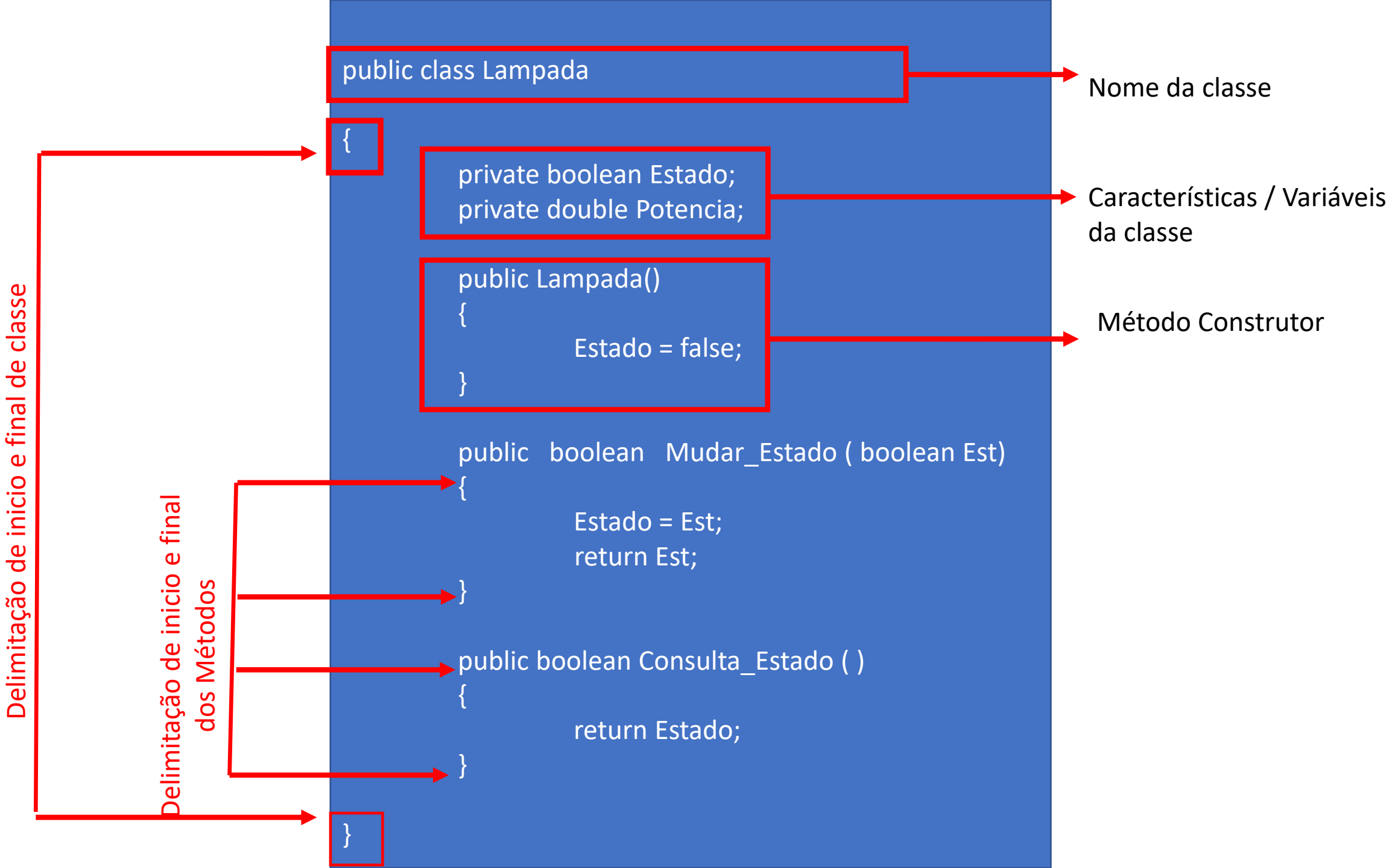
# ORIENTAÇÃO A OBJETOS

Programação Orientada a Objetos é um modelo de análise, projeto e programação de software baseado na composição e interação entre diversas unidades chamadas de 'objetos'.

# ORIENTAÇÃO A OBJETOS


# ORIENTAÇÃO A OBJETOS

# CLASSES


public class Lampada
{
private boolean Estado;
private double Potencia;
public Lampada()
{
Estado = false;
}
public boolean Mudar_Estado ( boolean Est)
{
Estado = Est;
return Est;
}
public boolean Consulta_Estado ( )
{
return Estado;
}
}
Nome da classe
Características / Variáveis da classe
Método Construtor
Delimitação de inicio e final de classe
Delimitação de inicio e final dos Métodos

CLASSES

```
public class Lampada
{
        private boolean Estado;
        private double Potencia;

        public Lampada()
        {
                Estado = false;
        }

        public boolean Mudar_Estado ( boolean Est)
        {
                Estado = Est;
        }

        public boolean Consulta_Estado ( )
        {
                return Estado;
        }
}
```

Método Construtor da classe

Método

Método

# CLASSES

```
public class Lampada
{
        private boolean Estado;
        private double Potencia;

        public Lampada()
        {
                Estado = false;
        }

        public boolean Mudar_Estado ( boolean Est)
        {
                Estado = Est;
        }

        public boolean Consulta_Estado ( )
        {
                return Estado;
        }
}
```

Características da classe

Método Construtor da classe

Método

Método

# OBJETIVOS DA POO

APROXIMAR O MUNDO REAL DO MUNDO DIGITAL.

# COMO A PROGRAMAÇÃO ERA FEITA ANTERIORMENTE?

Programação de baixo nível, ou seja, os dados eram informados diretamente para o computador da forma que ele entendia.

Se o computador era binário os dados eram informados neste formato, o mesmo aconteceria se o computador fosse decimal, os dados seriam informados de forma decimal e assim por diante.

# A PROGRAMAÇÃO NÃO ERA PADRONIZADA

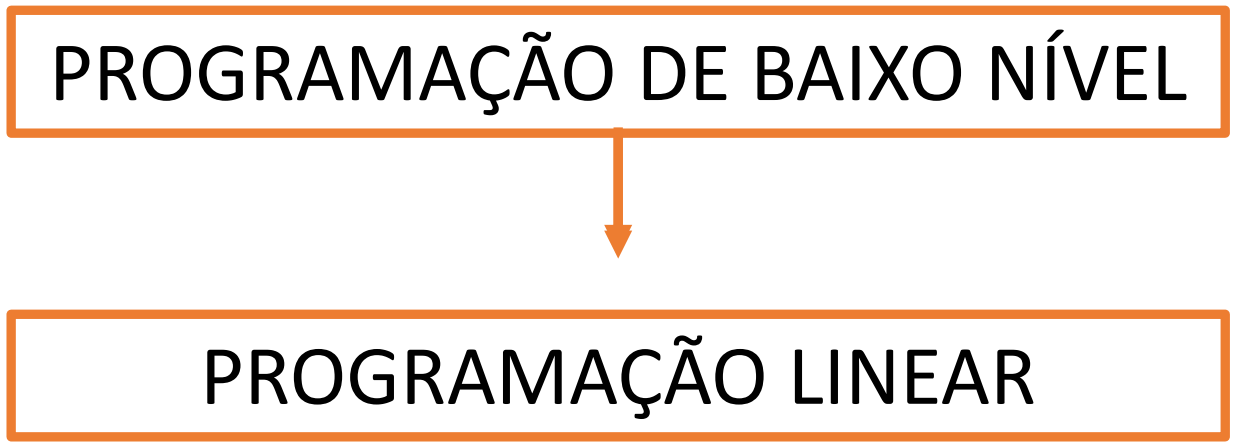
Um computador A era programado de uma certa forma, enquanto que um computador B poderia ser programado de um forma totalmente diferente.

# A PROGRAMAÇÃO NÃO ERA PADRONIZADA

Isto dificultava a programação dos computadores, mesmo porque a programação era realizada diretamente pelos engenheiros que haviam construído o hardware.

# A PRÓXIMA ETAPA FOI A PROGRAMAÇÃO LINEAR

Era a programação já possível de entender pelos programadores, mas sempre de cima para baixo e sem condicionantes (if) ou ciclos como o enquanto (while).

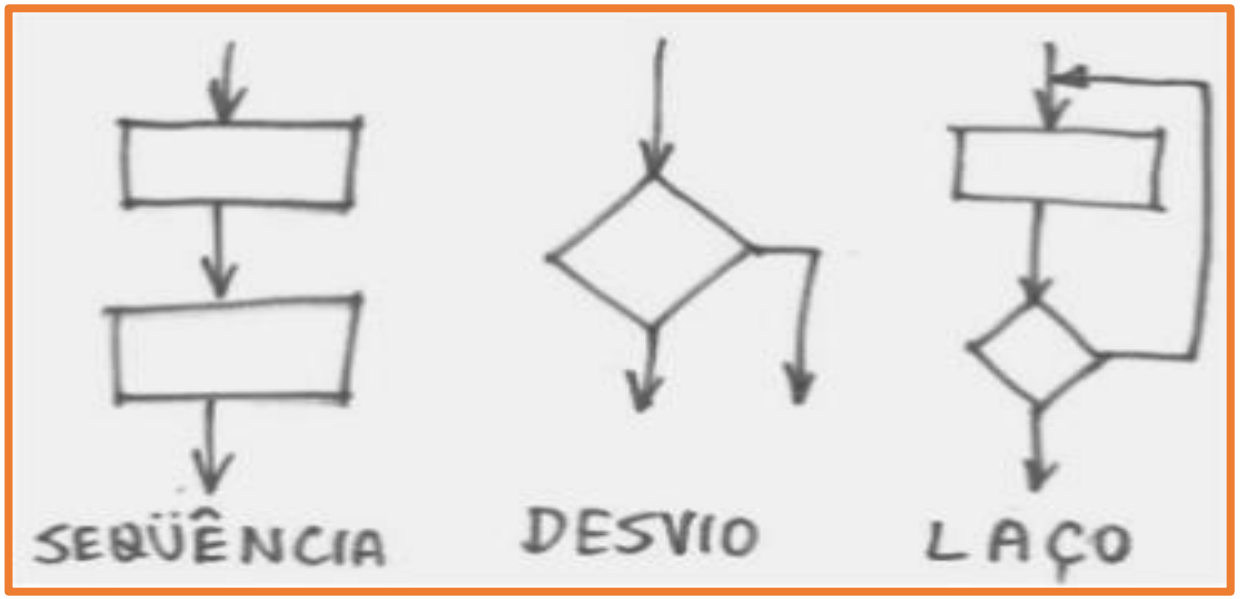


# A PRÓXIMA ETAPA FOI A PROGRAMAÇÃO ESTRUTURADA

Era a programação estruturada permitia pequenos pedaços da programação linear. Assim, poderia ser realizada em partes, não executando necessariamente todo o código.

PROGRAMAÇÃO LINEAR

PROGRAMAÇÃO ESTRUTURADA



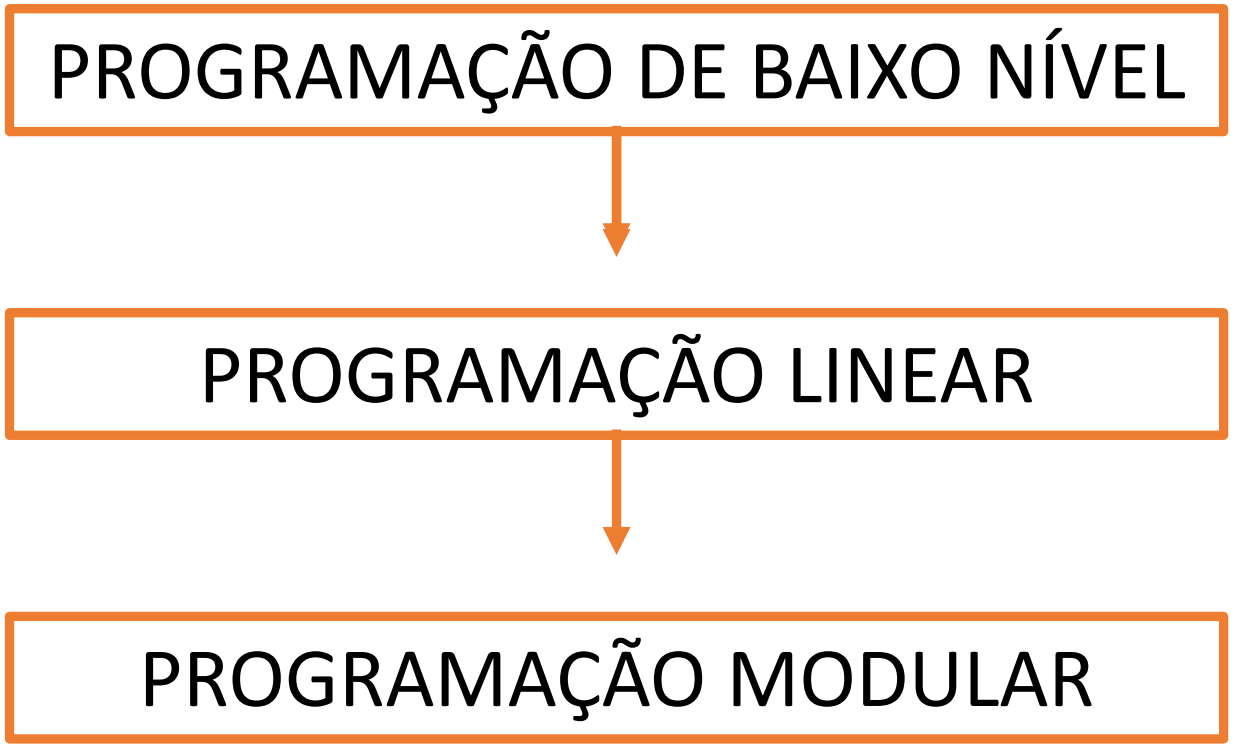
# EXEMPLO DE CÓDIGO EM LINGUAGEM ESTRUTURADA

## Programação Estruturada – Exemplo

```
1  principal() {
       inteiro i = 1; /* Declarando um inteiro */
3      imprima("Números pares até 100:");
       enquanto (i <= 100) {
5          se (i mod 2 == 0) {
               imprima(i);
7          }
           i = i + 1;
9      }
   }
```

# A PRÓXIMA ETAPA FOI A PROGRAMAÇÃO MODULAR

Permite a criação de módulos de programação estruturada.
Dividia o código em capsulas protegidas que poderiam compor sistemas maiores.



# SURGIMENTO DA POO

PROGRAMAÇÃO DE BAIXO NÍVEL

↓

PROGRAMAÇÃO LINEAR

↓

PROGRAMAÇÃO MODULAR

Posteriormente surgiu a programação orientada a objetos. Esta ampliava os conceitos das programação modular. Foi criada por Alan Kay.

# POSTULADO DE ALAN KAY

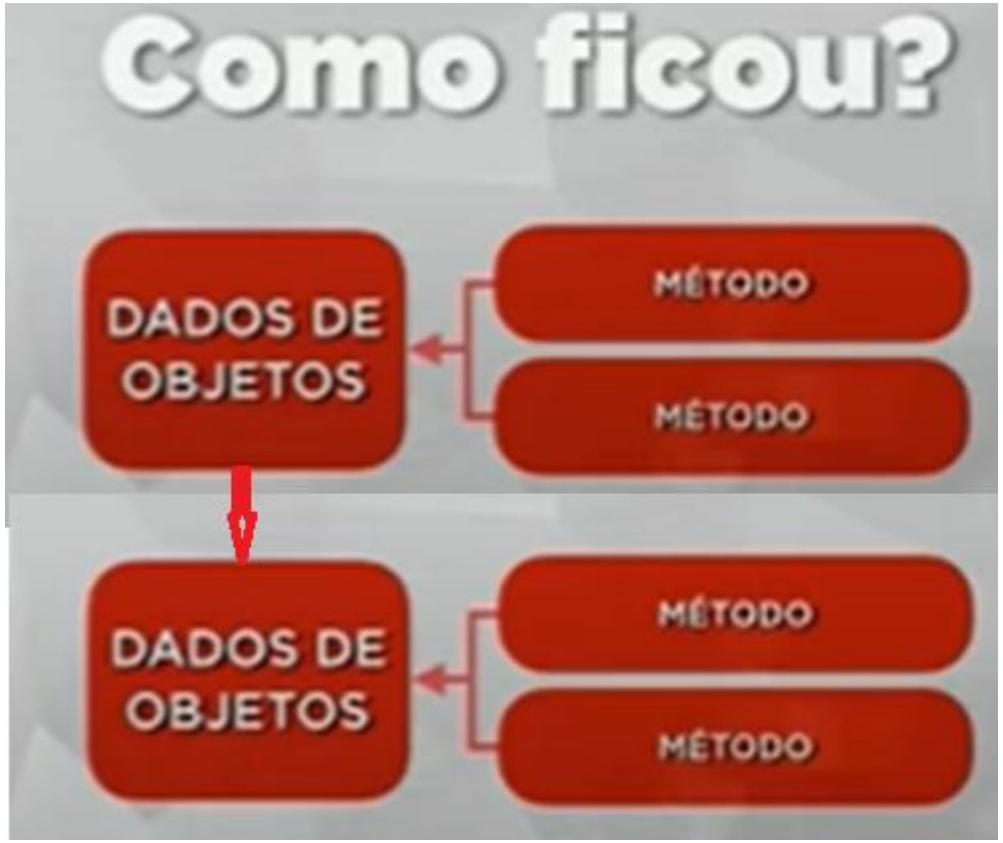
o computador ideal deveria funcionar como um organismo vivo, isto é, cada "célula" comportar-se-ia relacionando-se com outras a fim de alcançar um objetivo, contudo, funcionando de forma autônoma.

As células poderiam se reagrupar para resolver um outro problema ou desempenhar outras funções.

# QUALQUER COISA QUE SEJA UM OBJETO PODE SER UTILIZADO PARA EXPLICAR A POO

# A POO EXPLICA QUALQUER CONCEITO COM QUALQUER OBJETO

# COMO ERA E COMO É?

# LINGUAGENS POO

- C++
- Java
- PHP
- Phyton
- Ruby
- Visual Basic

É baseado em objetos, mas, não necessariamente precisa ser implementado.

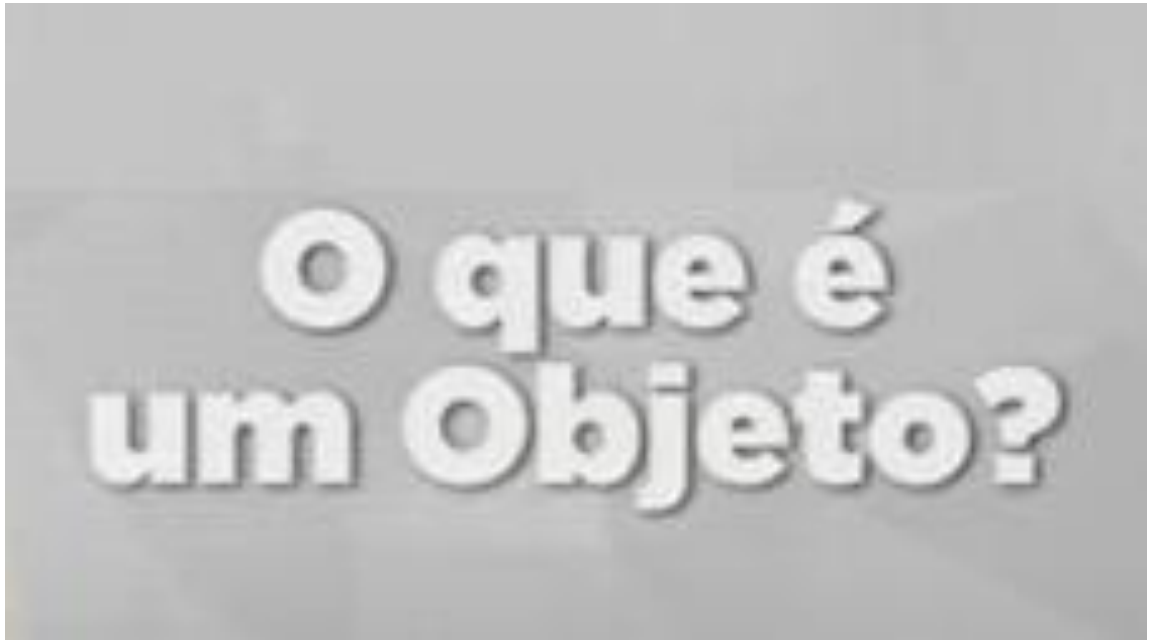
Totalmente implementado na POO.

Base em classes, mas nem todos os conceitos de POO são utilizados.

# VANTAGENS DA POO

- CONFIÁVEL
- OPORTUNO
- MANUTENIVEL
- EXTENSIVEL
- REUTILIZAVEL
- NATURAL

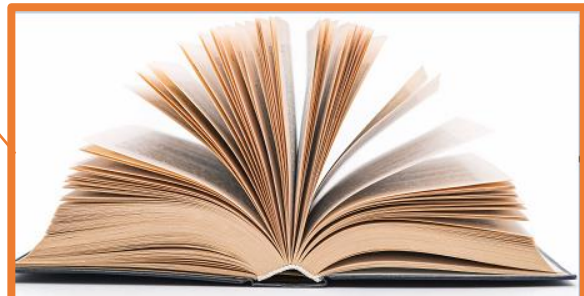
# PROGRAMAÇÃO ORIENTADA A OBJETOS



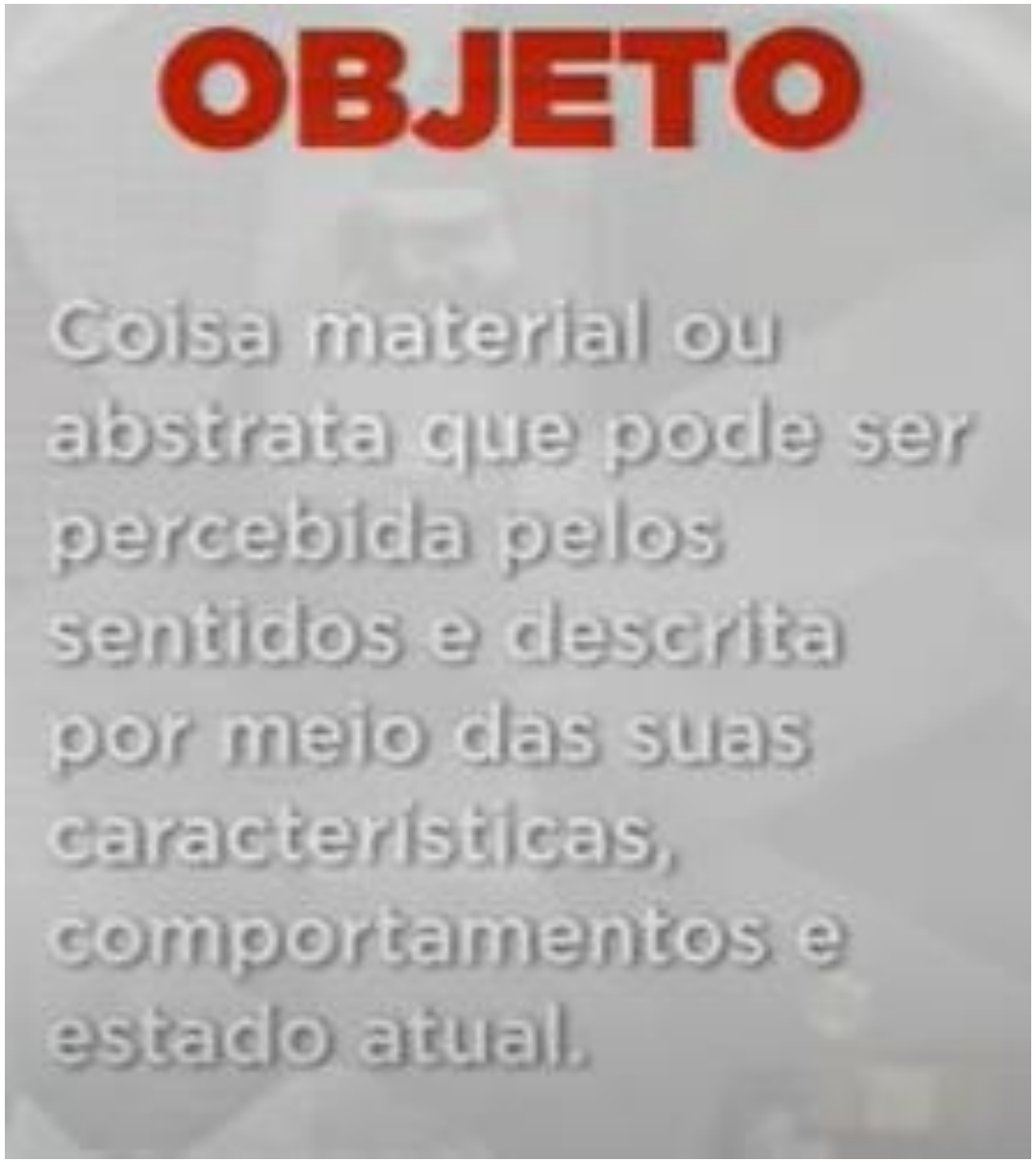
# PROGRAMAÇÃO ORIENTADA A OBJETOS



Objeto

# DEFINIÇÃO


OBJETO
Coisa material ou abstrata que pode ser percebida pelos sentidos e descrita por meio das suas características, comportamentos e estado atual.

# TODO OBJETO TEM

- Características.
- Estado.
- Comportamento.

# TODO OBJETO TEM

- Que características, estados e comportamentos tem o carro fiat 147?

# DEFINIÇÃO DE CLASSE

- As **classes** de programação são receitas de um objeto, aonde têm características e comportamentos, permitindo assim armazenar propriedades e métodos dentro dela.

Uma **classe** geralmente representa um substantivo, por exemplo: uma pessoa, um lugar, algo que seja “abstrato”.

# TODAS AS CLASSES TEM

- Coisas que a classe tem
- Coisas que a classe faz
- Como estou agora

# TODAS AS CLASSES TEM

- O que o objeto caneta tem?

→ Modelo;

→ Carga;

→ Cor;

→ Tampa;

# O QUE O OBJETO CANETA FAZ?

→ Escrever;
→ Rabiscar;
→ Pintar;
→ Tampar;
→ Estourar;

# ESTADOS QUE A CANETA PODE TER?

→ Escrevendo;
→ Guardada;
→ Rabiscando;
→ Estourada;

# TODAS AS CLASSES TEM

- Coisas que a classe tem → Atributos
- Coisas que a classe faz → Métodos
- Como estou agora → Estados

TODO OBJETO VEM APARTIR DE UM MOLDE,
OU SEJA, UMA CLASSE.

# CLASSE CANETA

```
class Caneta
{
        String Modelo;
        String Cor;
        float  Ponta;
        int Carga;
        boolean Tampada = false;                    //Tampada ou destampada;
        public Caneta()
        {
        }
        public void Metodo_Rabiscar()
        {
                if((Tampada==true)||(Carga==0)
                        {System.out.prinln("ERRO AO TENTAR ESCREVER!!!");}
                else
                        {System.out.prinln("RABISCAR!!!");
        }
        public void Metodo_Tamapar()
                {Estado = true;  }
...
```

# CLASSE CANETA

```
class Caneta
{
        String Modelo = “Momblac”;
        String Cor    = “Preto”;
        float  Ponta  = 0.5;
        int Carga     =  90;
        boolean Tampada = false;                    //Tampada ou destampada;
        public Caneta()
        {
        }
        public void Metodo_Rabiscar()
        {
                if((Tampada==true)||(Carga==0)
                        {System.out.prinln(“ERRO AO TENTAR ESCREVER!!!”);}
                else
                        {System.out.prinln(“RABISCAR!!!”);
        }
        public void Metodo_Modificar_Tamapar(boolean Tampada1)
                {Estado = Tampada1  }
…
```

CLASSE CANETA

# ESTADO DA CLASSE

REFERE-SE A UM MOMENTO.

Neste momento ....

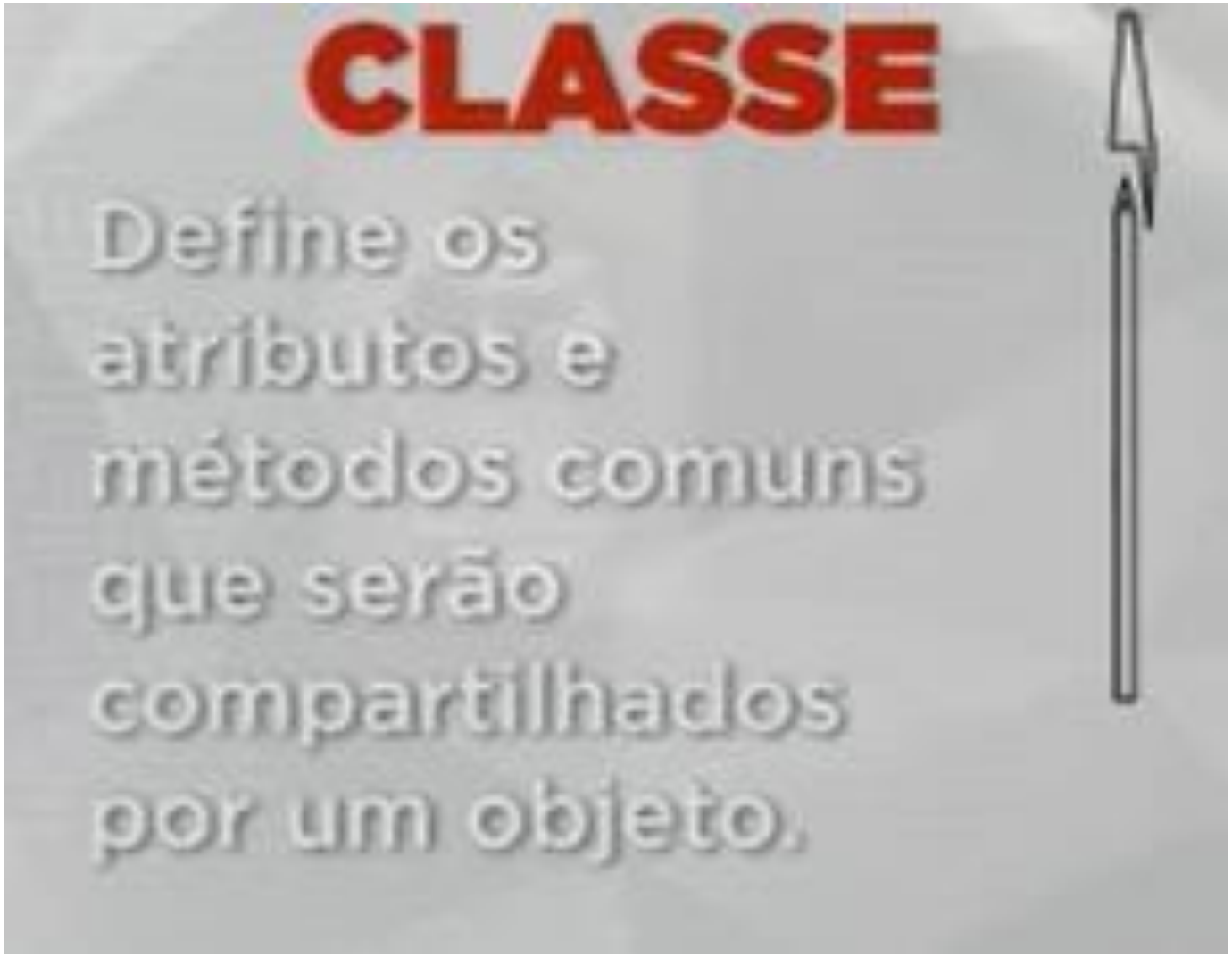
A caneta esta Destampada;
A caneta é Azul;
A caneta esta com 90% da carga;

# INSTANCIA DE UMA CLASSE


```
Caneta Caneta1 = new
Caneta();
Caneta1.cor(“Azul”);
Caneta1.Modelo (“Bic”);
Caneta1.Ponta = 0.7;
Caneta1.Carga = 100;
```

```
Caneta Caneta2 = new Caneta();
Caneta2.cor(“Preta”);
Caneta2.Modelo (“Uni-Ball”);
Caneta2.Ponta = 0.5;
Caneta2.Carga = 60;
```

# CLASSE E OBJETOS



# ABSTRAÇÃO

É utilizada para a definição de entidades do mundo real. Sendo onde são criadas as classes. Essas entidades são consideradas tudo que é real, tendo como consideração as suas características e ações.

| Entidade | Características | Ações |
|---|---|---|
| Carro, Moto | tamanho, cor, peso, altura | acelerar, parar, ligar, desligar |
| Elevador | tamanho, peso máximo | subir, descer, escolher andar |
| Conta Banco | saldo, limite, número | depositar, sacar, ver extrato |

# Para padronizar a Classe


- Inicial das classes com letras maiúsculas;
- Atributos com letras minúsculas;
- Métodos como verbos;
- Mais...

OUTRA FORMA:

- Variáveis com a palavra var antes. Exemplo: var_Idade, var_Peso;

Métodos com a palavra método antes.

Metodo_Somar(int x, int y);

# Visibilidade dos atributos, métodos e classes

- + publico
- - privado
- # protegido

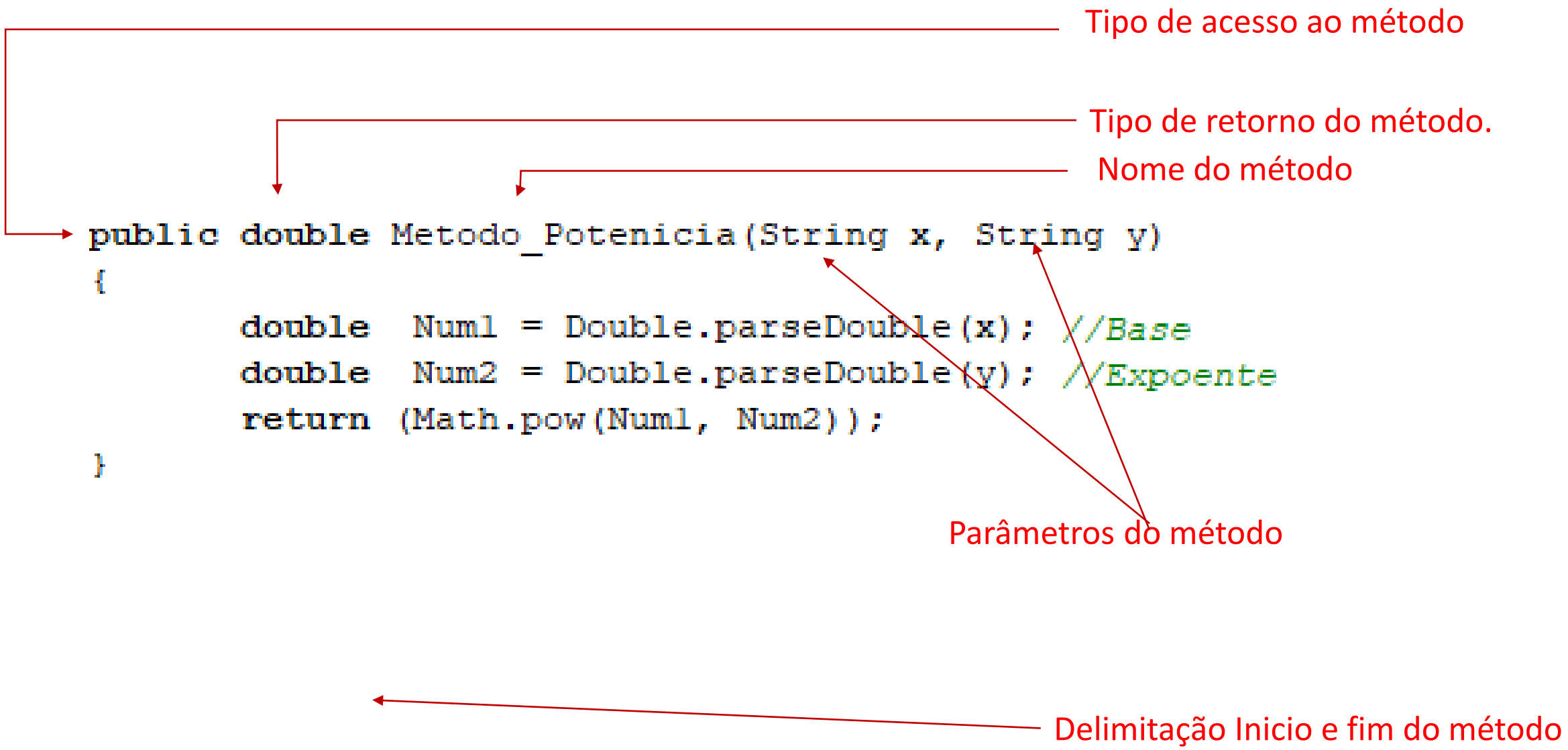
```
public class aluno
{
        public  String var_Nome = “Asgard Bertoldi”;
        private Idade =   15;
        protect String  Var_Profissao = “Estudante”;
}
```

# Métodos

Método em Java é equivalente a uma função, sub-rotina ou procedimento em outras linguagens de programação. Não existe em Java o conceito de métodos globais.

Todos os métodos devem sempre ser definidos dentro de uma classe.

# Métodos


Tipo de acesso ao método
Tipo de retorno do método.
Nome do método
public double Metodo_Potenicia(String x, String y)
{
double Num1 = Double.parseDouble(x); //Base
double Num2 = Double.parseDouble(y); //Expoente
return (Math.pow(Num1, Num2));
}
Parâmetros do método
Delimitação Inicio e fim do método

# Métodos

Corpo do método.

```
public double Metodo_Potenicia(String x, String y)
{
        double  Num1 = Double.parseDouble(x); //Base
        double  Num2 = Double.parseDouble(y); //Expoente
        return (Math.pow(Num1, Num2));
}
```

Delimitação Inicio e fim do método

# EXEMPLOS DE MÉTODOS

```
import java.math.*;
public class Calculadora_Avancada
{
        public Calculadora_Avancada()
        {         }
        public int Metodo_Fatorial(String x)
        {
                int Num1 = Integer.parseInt(x); int Res = Num1 -1;
                while(Num1 >0)
                {
                        Num1 = Num1 * (Res); Res = Res -1;
                }
                return Res;
        }
        public double Metodo_PI()
                {return (Math.PI);}
        public double Metodo_Raiz_Cubica(String x)
        {
                double Num1 = Float.parseFloat(x);
                return (Math.cbrt(Num1));
        }
        public double Metodo_Potenicia(String x, String y)
        {
                double  Num1 = Double.parseDouble(x); //Base
                double  Num2 = Double.parseDouble(y); //Expoente
                return (Math.pow(Num1, Num2));
        }
        public double Metodo_Logaritmo(String x)
        {
                double Num1 = Float.parseFloat(x); //Num
                return (Math.log(Num1));
        }
```

# EXEMPLOS DE MÉTODOS

```
public float Metodo_Soma(String x, String y)
{
        float Num1 = Float.parseFloat(x);
        float Num2 = Float.parseFloat(y);
        return (Num1+Num2);
}
public float Metodo_Subtracao(String x, String y)
{
        float Num1 = Float.parseFloat(x);
        float Num2 = Float.parseFloat(y);
        return (Num1-Num2);
}

public float Metodo_Multiplicacao(String x, String y)
{
        float Num1 = Float.parseFloat(x);
        float Num2 = Float.parseFloat(y);
        return (Num1*Num2);
}
public float Metodo_Divisao(String x, String y)
{
        float Num1 = Float.parseFloat(x);
        float Num2 = Float.parseFloat(y);
        return (Num1/Num2);
}
public double Metodo_Raiz_Quadrada(String x)
{
        double Num1 = Double.parseDouble(x);
        return (Math.sqrt(Num1));
}
```

# Atividade

- Elaborar 03 classes.  (Clientes, produtos e vendas).

Veja quais atributos, estados e métodos que seriam necessário.

Vamos fazer e trocar ideias sobre isto. Não se preocupe se as classes não estiverem corretas neste momento.

Enviar por e-mail as 03 classes por email.
andre@oxnar.com.br

# MAIS DEFINIÇÕES

# Esqueleto do programa Java

```
/*ESQUELETO DO CÓDICO JAVA*/

/*Semana 03.
 *Turma: Info 3
 *Prof. André Silva.
 **/

public class Oxnar
{
        public static void main(String args [])
        {
                System.out.println("OLÁ MUNDO");
        }
}
```

# Esqueleto do programa Java

```
/*ESQUELETO DO CÓDICO JAVA*/
```

Comentário com uma linha

```
/*Semana 03.
 *Turma: Info 3
 *Prof. André Silva.
 **/
```

Comentário com diversas linhas

```
public class Oxnar
{
        public static void main(String args [])
        {
                System.out.println("OLÁ MUNDO");
        }
}
```

Tipo de acesso

Nome da classe

Delimitação Inicio e fim da Classe

# Esqueleto do programa Java

```
/*ESQUELETO DO CÓDICO JAVA*/

/*Semana 03.
 *Turma: Info 3
 *Prof. André Silva.
 **/

public class Oxnar
{
        public static void main(String args [])
        {
                System.out.println("OLÁ MUNDO");
        }
}
```

Tipo de acesso ao método

Tipo de retorno do método.

Nome do método.

Parâmetros do método

Delimitação Inicio e fim do método

# Outra implementação do programa

```
/*ESQUELETO DO CÓDICO JAVA*/

/*Semana 03.
 *Turma: Info 3
 *Prof. André Silva.
 **/

public class Oxnar2
{
        public Oxnar2()
        {
                System.out.println("Oxnar - Olá mundo");
        }
        public static void main(String args [])
        {
                new Oxnar2();
        }
}
```

**Método construtor da classe**

- É público;
- Pode ou não receber argumentos;
- Não retorna nada;
- Tem o mesmo nome da classe;

EXEMPLO WHILE

```
public class ExemploWhile
{
        public static void main (String args [])
       {new ExemploWhile(); }
        int x = 0;
        int y = 1;
        public  ExemploWhile()
        {
                while(x <1000)
                {
                        x = x + y;
                        System.out.println(“”+x);
                        y = y + x;
                        System.out.println(“”+y);
                }
        }


}
```