



Uniube

Programação Orientada a Objetos

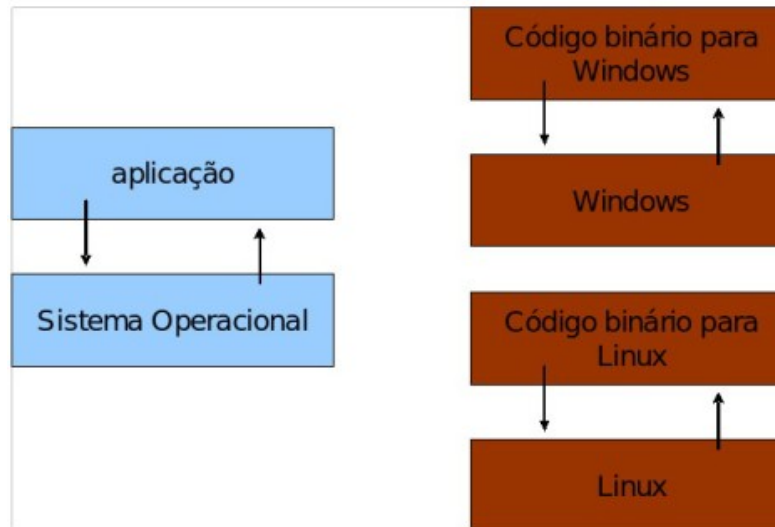
Prof. Me. Clênio Silva
clenioeduardo@yahoo.com.br

Máquina Virtual

- Em uma linguagem de programação C e Pascal:

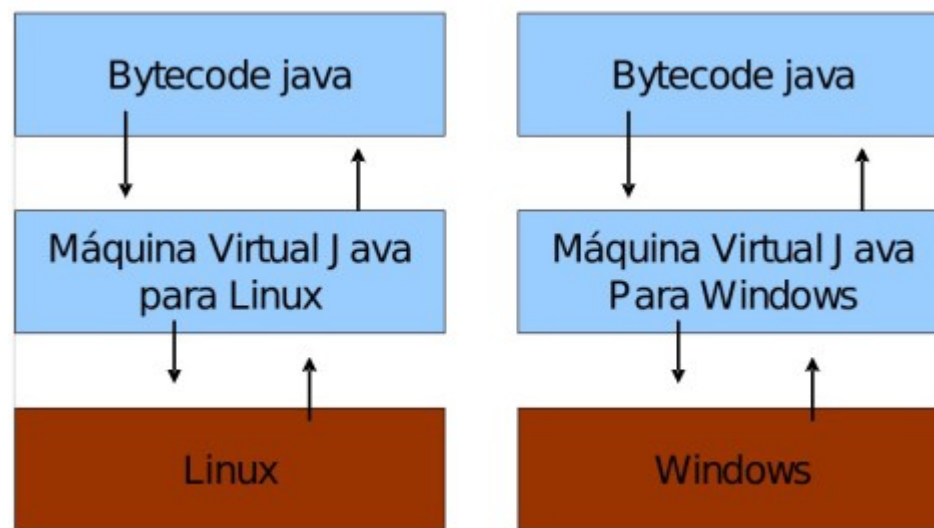


Código fonte é compilado para código de máquina específico de um sistema operacional



Máquina Virtual

O Java utiliza o conceito de máquina virtual, no qual existe, entre o sistema operacional e a aplicação, uma camada extra responsável por traduzir a aplicação para as respectivas chamadas do sistema operacional em que ela está rodando.



A aplicação roda sem nenhum envolvimento com o sistema operacional, sempre conversando apenas com a Java Virtual Machine (JVM).

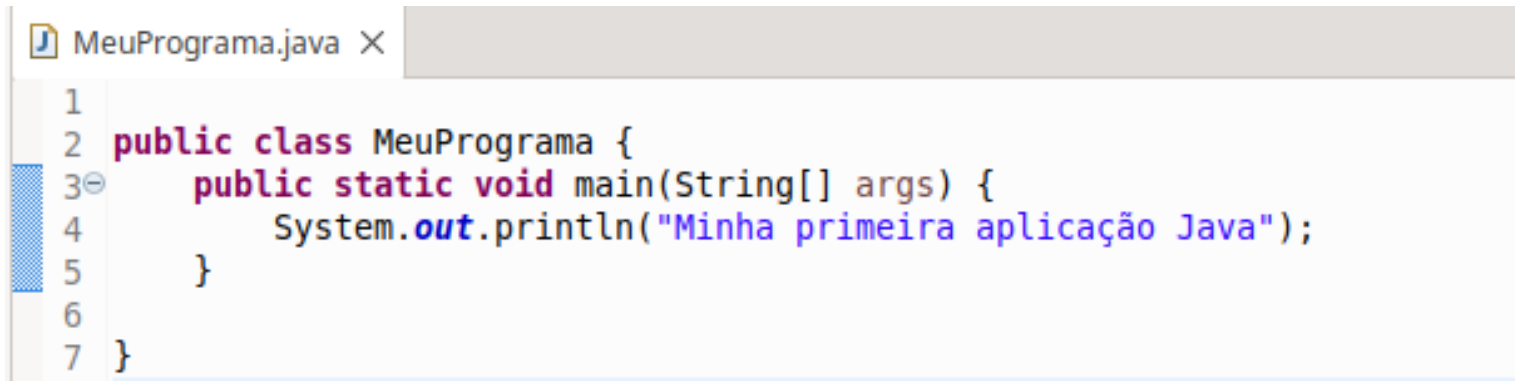
O que é necessário instalar para desenvolver em Java

- JRE: **Java Runtime Enviroment**. Ambiente de execução Java formado pela JVM e bibliotecas.
- JDK: **Java Development Kit**.

Tanto o JRE quanto o JDK podem ser baixados do site <https://www.oracle.com/br/java/technologies/downloads/> .

Recapitulando a ultima aula

- Executando o primeiro programa via linha de comando:

A screenshot of a Java IDE window titled 'MeuPrograma.java'. The code is as follows:

```
1  
2 public class MeuPrograma {  
3     public static void main(String[] args) {  
4         System.out.println("Minha primeira aplicação Java");  
5     }  
6  
7 }
```

- Gerando executavel: **javac MeuPrograma.java**
é gerado o arquivo **MeuPrograma.class**
- Executando: **java MeuPrograma**

Recapitulando a ultima aula

- Conceitos de classe, atributo e método:

```
1
2 public class Cachorro {
3     String raca;
4     int tamanho;
5
6     public void latir() {
7         System.out.println("au au...");
8     }
9
10 }
11
12
```

Classe: utiliza para representar o objeto que será instanciando
Atributo: Usado para representar as características de um objeto
Método: Usado para representar o comportamento de um objeto

Modificadores de Acesso

- **private** : na própria classe
- **protected** : em subclasses
- **public** : em qualquer lugar

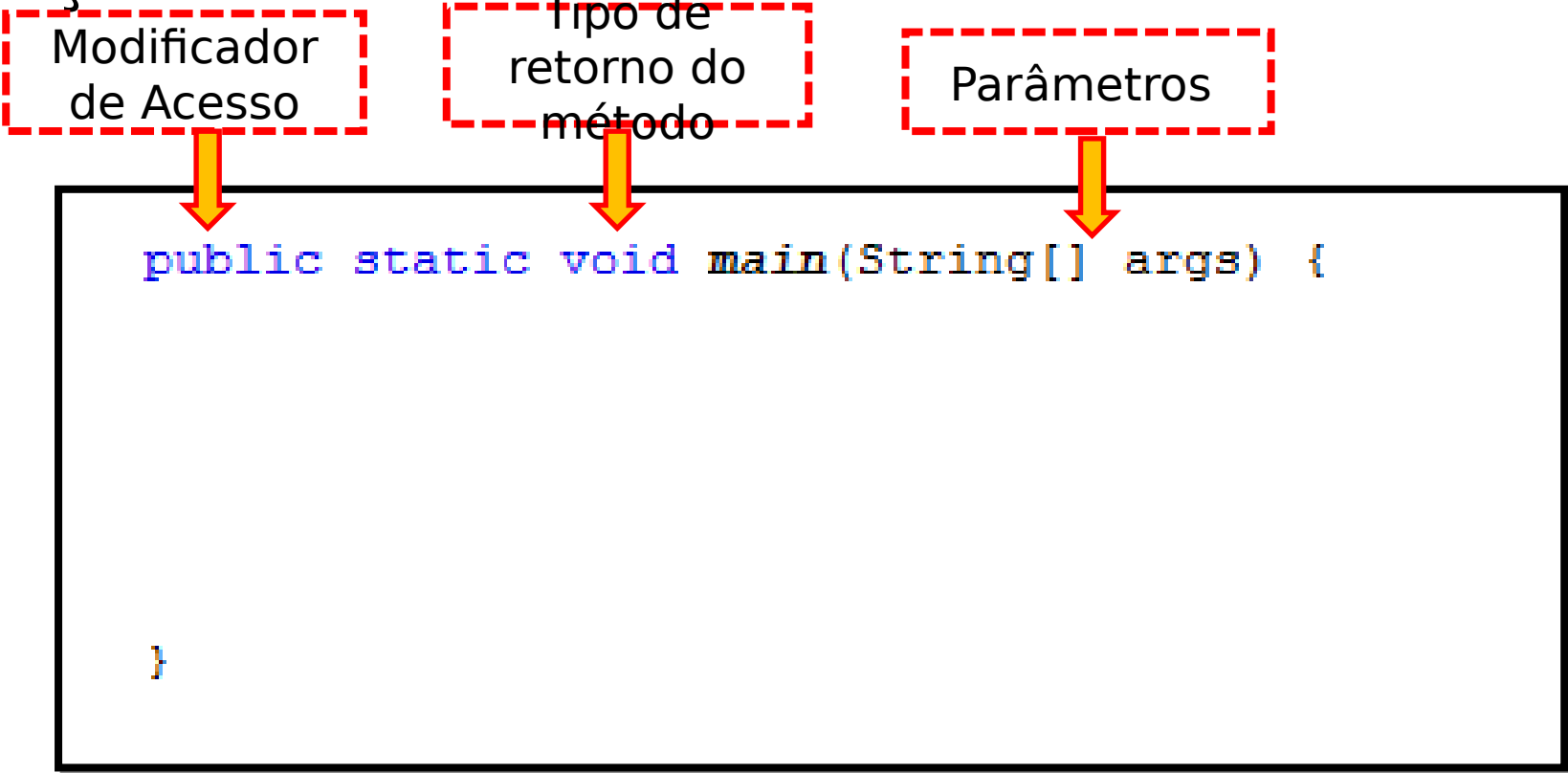
Modificadores de acesso

Construção de um método

Modificador
de Acesso

Tipo de
retorno do
método

Parâmetros



```
public static void main(String[] args) {  
  
}
```


Outros Modificadores

strictfp	abstract	volatile
static	final	synchronized
transient	native	@annotations

Modificadores de acesso

Como estávamos
fazendo

```
public class Funcionario {  
    String nome;  
    String departamento;  
    double salario;  
    String data_de_entrada;  
    String rg;
```

Utilizando modificador
de acesso **private**
para encapsular os
atributos

```
public class Funcionario {  
    private String nome;  
    private String departamento;  
    private double salario;  
    private String data_de_entrada;  
    private String rg;
```

Encapsulamento

- Agora que você já sabe como utilizar os modificadores de acesso vamos trabalhar com encapsulamento
- O encapsulamento permite tornar o código mais flexível e resistente a mudanças
- Encapsula dados e comportamentos

Benefício: não é necessário conhecer a implementação interna dos objetos para utilizar os mesmos.

Encapsulamento

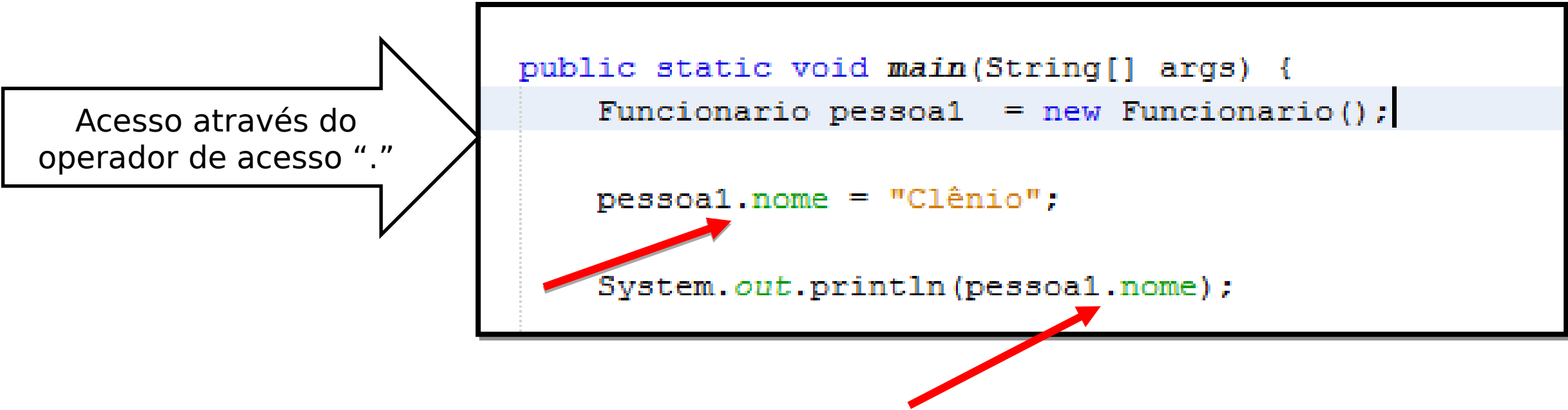
- Métodos e variáveis
- GET / IS = Para captura
- SET = Para configuração

Funcionario
nome : String salario : double ativo : boolean
getSalsrio() : double setSalario(salario : double isAtivo() : boolean setAtivo(ativo : boolean) : void setSenha(senha : String) : void isSenhaCorreta(senha : String) : boolean

Encapsulamento

Acesso através do
operador de acesso “.”

```
public static void main(String[] args) {  
    Funcionario pessoal = new Funcionario();  
  
    pessoal.nome = "Clênio";  
    System.out.println(pessoal.nome);  
}
```

The diagram illustrates the concept of encapsulation. On the left, a text box explains that access is done through the dot operator. On the right, a code block shows a Java program. A large black arrow points from the text box to the code. Two red arrows highlight specific parts of the code: one points to the dot in 'pessoal.nome' on the assignment line, and the other points to the dot in 'pessoal.nome' on the print statement line.

Encapsulamento

- Utilizando encapsulamento para atribuir e recuperar o valor do atributo de um objeto

Aqui tanto a atribuição quanto a recuperação foi feita através de métodos. Esses métodos são chamados getters e setters .

```
public static void main(String[] args) {  
    Funcionario pessoal = new Funcionario();  
  
    pessoal.setNome("Clênio Eduardo");  
  
    System.out.println(pessoal.getNome());  
}
```

Encapsulamento

Método getter do atributo
nome.

```
public String getNome() {  
    return nome;  
}
```

Método setter do atributo
nome.

```
public void setNome(String nome) {  
    this.nome = nome;  
}
```

Praticando...

- Utilizando encapsulamento vamos implementar o seguinte diagrama de classes:

