



INSTITUTO FEDERAL
Sertão Pernambucano
Campus Salgueiro

Programação I

2º ano - EMI Informática

Prof. Heraldo Gonçalves Lima Junior
heraldolimajr.com.br

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some solid and some hollow, connected by thin lines. The overall structure is a dense, branching network.

1.

Paradigmas de Programação

1.1. Introdução

- **O que é uma linguagem?**
 - “Conjunto de regras que estabelecem normas de comunicação”.
 - Caso as partes envolvidas na comunicação falem línguas diferentes, surge a necessidade de um tradutor (intermediário).



1.1. Introdução

- **O que é uma linguagem de Programação?**
 - Também é um conjunto de regras que estabelecem normas de comunicação entre o programador e o computador.
 - Uma LP deve ser extremamente **formal e exata**.
 - Uma linguagem ambígua torna-se difícil de ser traduzida para uma linguagem de máquina.

1.2. Evolução das LPs

- **Primeira Geração: Linguagem de máquina**
 - Código de Máquina (0s e 1s).
- **Segunda Geração:**
 - Linguagem de Montagem - **Assembler**

1.2. Evolução das LPs

- **Terceira Geração**

- Imperativas: FORTRAN, Cobol, Basic, Algol, ADA, Pascal, **C**
- Lógicas e Funcionais: LISP, ML, Prolog

- **Quarta Geração**

- Geradores de Relatórios, Linguagens de Consultas: **SQL**, CSP

1.2. Evolução das LPs

- **Quinta Geração**
 - LOO : Smalltalk, **Java**, Eiffel, Simula 67
- **Sexta Geração ?**
 - Web e Linguagens Dinâmicas : **Python**, **JavaScript**, Ruby

1.2. Evolução das LPs

- **Quinta Geração**
 - LOO : Smalltalk, **Java**, Eiffel, Simula 67
- **Sexta Geração ?**
 - Web e Linguagens Dinâmicas : **Python**, **JavaScript**, Ruby

1.3. O que são paradigmas de programação?

- Os paradigmas são **modelagens de escrita de código** que podem ser aplicados a várias linguagens, desde que estas permitam.
- É possível ainda aplicar mais de um paradigma a uma mesma solução em uma linguagem previamente escolhida.

1.4. Atividade

- ◎ Pesquise sobre os paradigmas de programação abaixo e suas características:
 - Imperativo
 - Funcional
 - Lógico

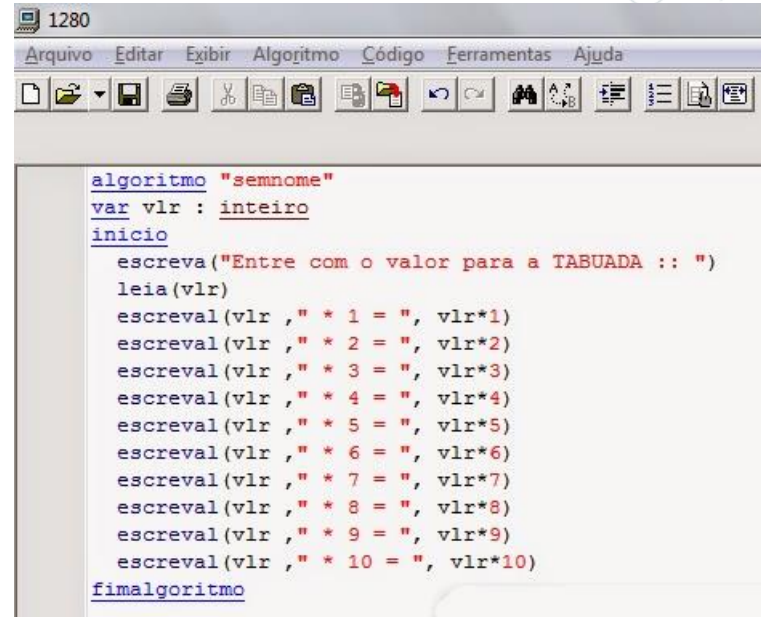
1.5. Pergunta:

- © Afinal, qual paradigma nós estamos utilizando?



1.6. Paradigma Procedural (Imperativo)

- ⦿ Nesse tipo de construção, as instruções devem ser passadas ao computador na sequência em que devem ser executadas.



The screenshot shows a software interface with a menu bar (Arquivo, Editar, Exibir, Algoritmo, Código, Ferramentas, Ajuda) and a toolbar. The main text area contains the following code:

```
algoritmo "semnome"
var vlr : inteiro
inicio
  escreva("Entre com o valor para a TABUADA :: ")
  leia(vlr)
  escreval(vlr, " * 1 = ", vlr*1)
  escreval(vlr, " * 2 = ", vlr*2)
  escreval(vlr, " * 3 = ", vlr*3)
  escreval(vlr, " * 4 = ", vlr*4)
  escreval(vlr, " * 5 = ", vlr*5)
  escreval(vlr, " * 6 = ", vlr*6)
  escreval(vlr, " * 7 = ", vlr*7)
  escreval(vlr, " * 8 = ", vlr*8)
  escreval(vlr, " * 9 = ", vlr*9)
  escreval(vlr, " * 10 = ", vlr*10)
finalgoritmo
```

1.6. Paradigma Procedural (Imperativo)



- © Esse tipo de programação é recomendada em projetos nos quais **não se espera que haja mudanças significativas** ao longo do tempo (programa estático) ou quando **não existiram muitos elementos compartilhados.**

1.6. Paradigma Procedural (Imperativo)

- © Esse paradigma tem a vantagem de **ser eficiente e de permitir uma modelagem tal qual o mundo real**, além de ser bem estabelecido e bastante flexível.
- © Por outro lado, o **código fonte gerado é de difícil legibilidade**.



1.7. Código Fonte

- © É aquele escrito pelo programador em linguagem de programação.

```
import java.io.*;
public class Calculadora {

    public static void main (String []
args){
    int a,b,c;
        a = 5;
        b = 3;
        c = a++ + ++b;

    System.out.println("Valores = " + a + b + c);
    }
}
```

```
cont = 0
n = int(input('Digite um valor: '))
if n > 1:
    for i in range(1, 11):
        if n % i == 0:
            cont += 1
    if cont > 2:
        print(f'Não é primo, ele é divisível {cont} vezes')
    else:
        print(f'É primo, ele é divisível apenas {cont} vezes')
else:
    print('Não é primo')
```


1.7. Código Fonte

- © É aquele escrito pelo programador em linguagem de programação.
- © **O computador não consegue entender esses comandos.**

```
import java.io.*;
public class Calculadora {

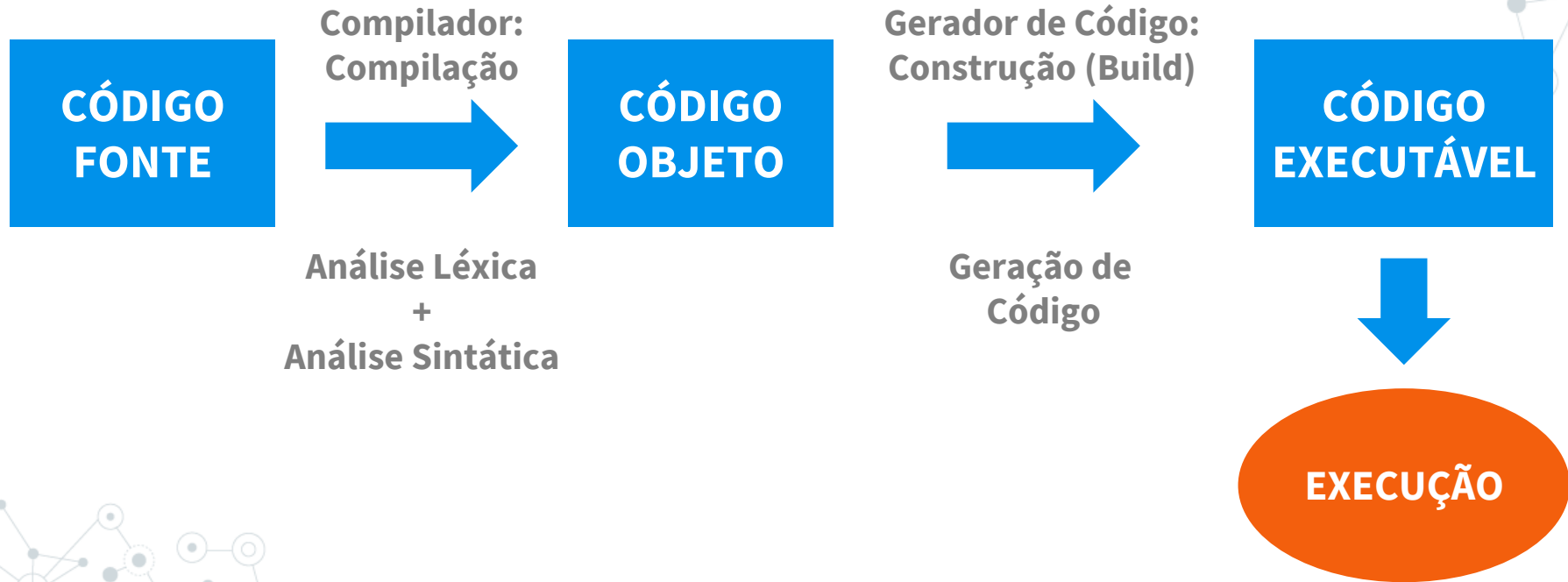
    public static void main (String []
args){
    int a,b,c;
        a = 5;
        b = 3;
        c = a++ + ++b;

    System.out.println("Valores = " + a + b + c);
    }
}
```

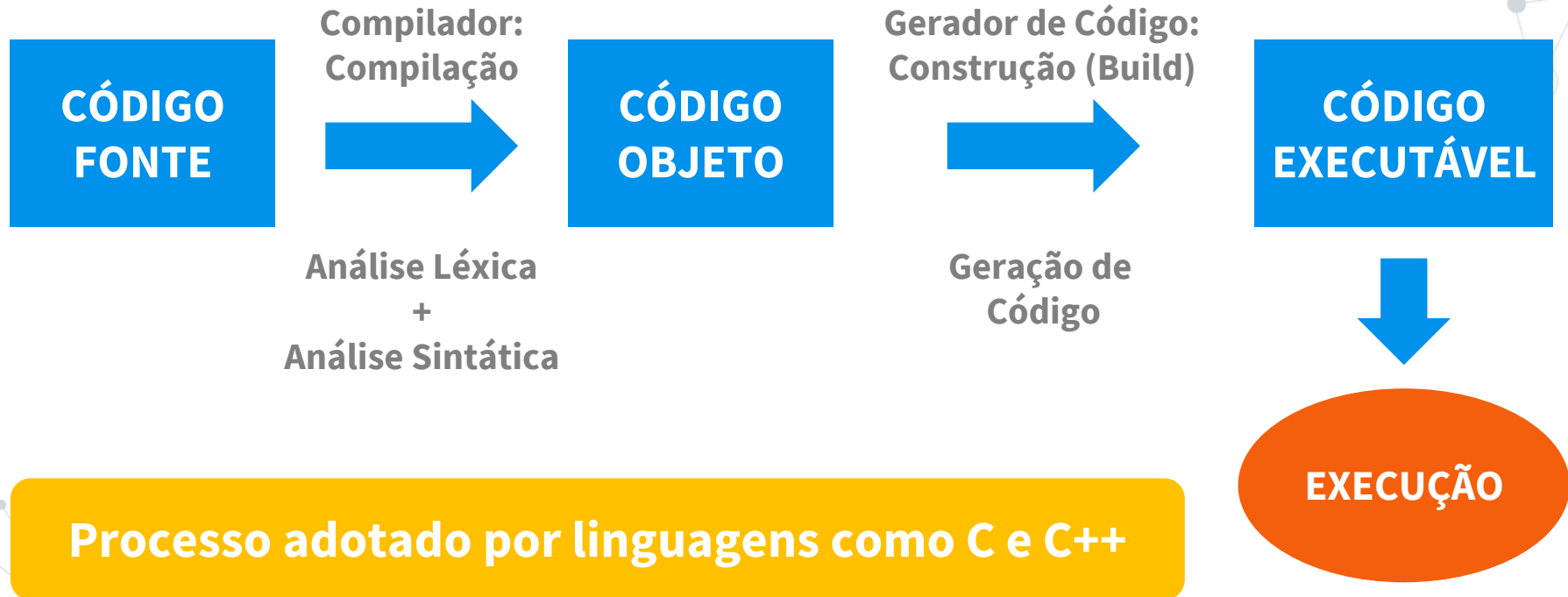
```
cont = 0
n = int(input('Digite um valor: '))
if n > 1:
    for i in range(1, 11):
        if n % i == 0:
            cont += 1

    if cont > 2:
        print(f'Não é primo, ele é divisível {cont} vezes')
    else:
        print(f'É primo, ele é divisível apenas {cont} vezes')
else:
    print('Não é primo')
```

1.8. Compilação



1.8. Compilação



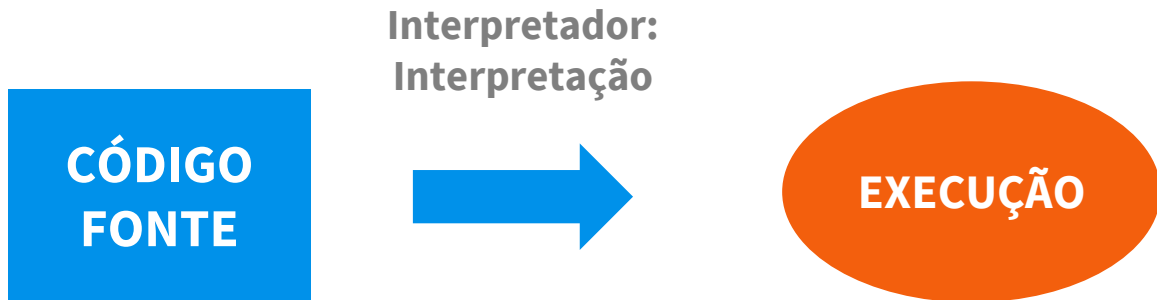
1.9. Interpretação



Análise Léxica + Análise Sintática + Geração de Código

SOB DEMANDA

1.9. Interpretação

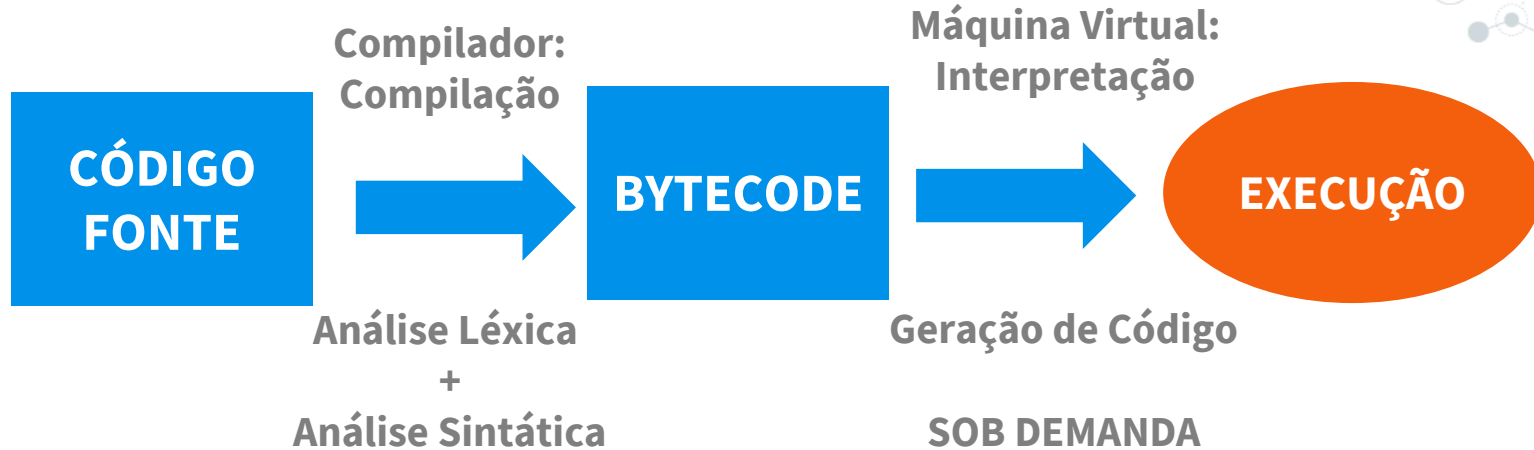


Análise Léxica + Análise Sintática + Geração de Código

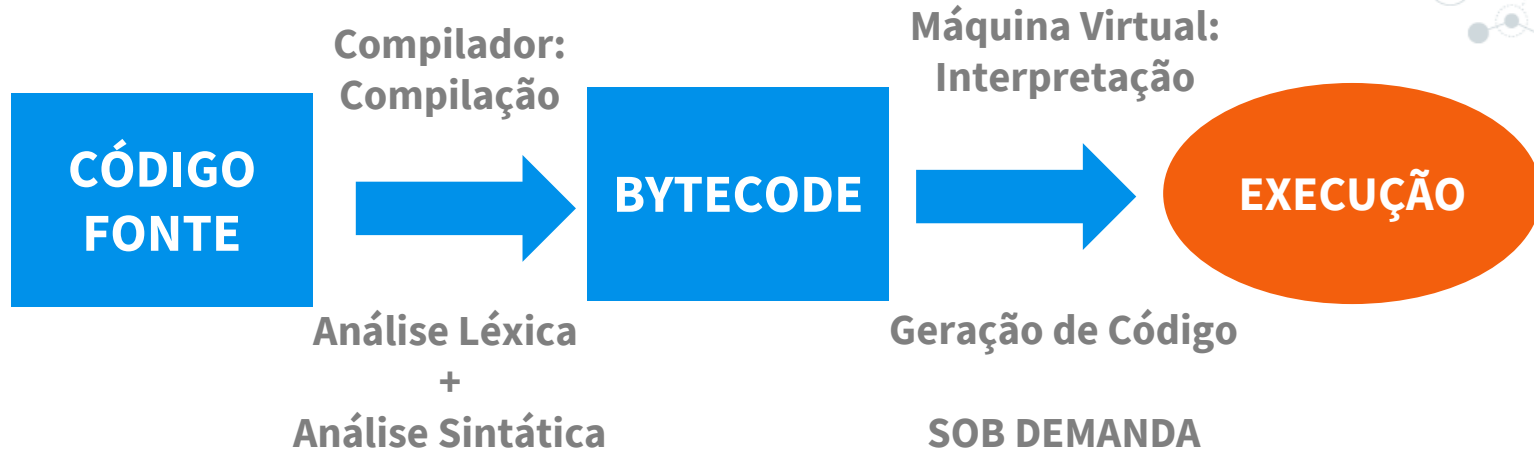
SOB DEMANDA

Processo adotado por linguagens como PHP, Python, JavaScript, Ruby...

1.10. Abordagem Híbrida



1.10. Abordagem Híbrida



Processo adotado por linguagens como Java (JVM), C# (Microsoft .NET Framework)

1.11. Vantagens

◎ **Compilação:**

- Velocidade do programa;
- Auxílio do compilador antes da execução.

◎ **Interpretação:**

- Flexibilidade de manutenção;
- Expressividade e simplicidade da linguagem;
- Código fonte não precisa ser recompilado para rodar em plataformas diferentes.

1.11. Vantagens

Abordagem Híbrida

◎ Compilação:

- Velocidade do programa;
- **Auxílio do compilador antes da execução.**

◎ Interpretação:

- Flexibilidade de manutenção;
- Expressividade e simplicidade da linguagem;
- **Código fonte não precisa ser recompilado para rodar em plataformas diferentes.**

CÓDIGO FONTE

**CÓDIGO
EXECUTÁVEL**
(ESPECÍFICA PARA O S.O.)

**SISTEMA
OPERACIONAL**

HARDWARE

C, C++

CÓDIGO FONTE

INTERPRETADOR
(ESPECÍFICA PARA O S.O.)

**SISTEMA
OPERACIONAL**

HARDWARE

PHP, JAVASCRIPT

CÓDIGO FONTE

BYTECODE
(CÓDIGO PRÉCOMPILADO)

**MÁQUINA
VIRTUAL**
(ESPECÍFICA PARA O S.O.)

**SISTEMA
OPERACIONAL**

HARDWARE

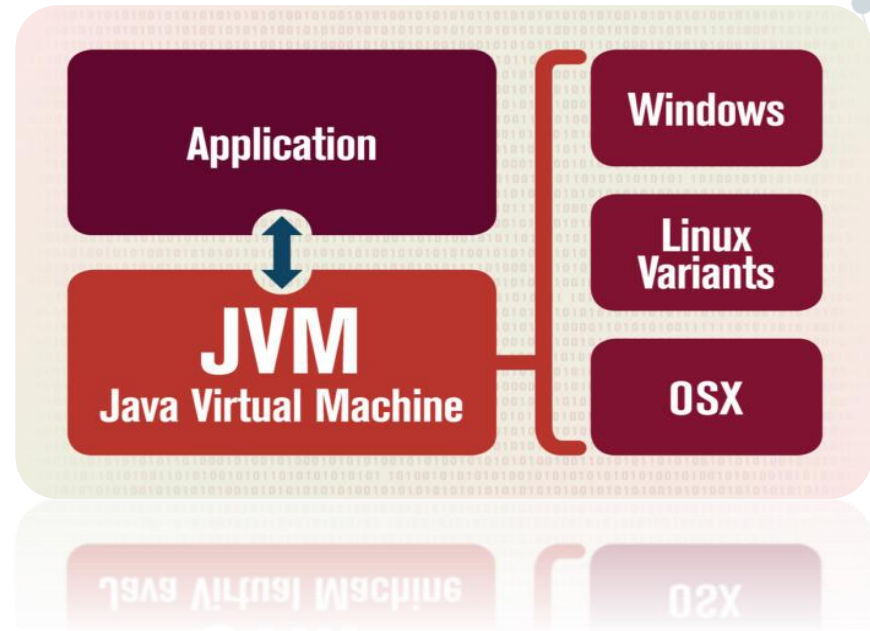
JAVA, C#

1.12. JVM – Java Virtual Machine

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

1.12. JVM – Java Virtual Machine

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



1.13. O que é Java?

- ◎ **Linguagem de programação;**
- ◎ **Plataforma de desenvolvimento e execução**
 - Bibliotecas (API);
 - Ambientes de execução.



1.14. Histórico

- ◎ **Problemas resolvidos e motivos do sucesso:**
 - Ponteiros / gerenciamento de memória;
 - Portabilidade: Reescrever código ao mudar SO;
 - Utilização em dispositivos diversos;
 - Custo;
- ◎ **Criada pela Sun Microsystems na década de 90;
Adquirida pela Oracle em 2010.**

1.15. Aspectos Notáveis

- ◎ Código compilado para bytecode e executado na JVM;
- ◎ Portável, robusta e segura;
- ◎ Roda em vários dispositivos;
- ◎ Domina o mercado desde a década de 90;
- ◎ Padrão Android por muitos anos.



1.16. Edições

◎ **Java ME – Java Micro Edition**

- Dispositivos embarcados, móveis e IoT.

◎ **Java SE – Java Standard Edition**

- Desktop e servidores.

◎ **Java EE – Java Enterprise Edition**

Aplicações corporativas.

1.17. JDK – Java Development Kit

- ◎ Para criar applets e aplicações Java, você precisa de ferramentas de desenvolvimento como o JDK.
- ◎ O JDK inclui o Java Runtime Environment, o compilador Java e as APIs Java.

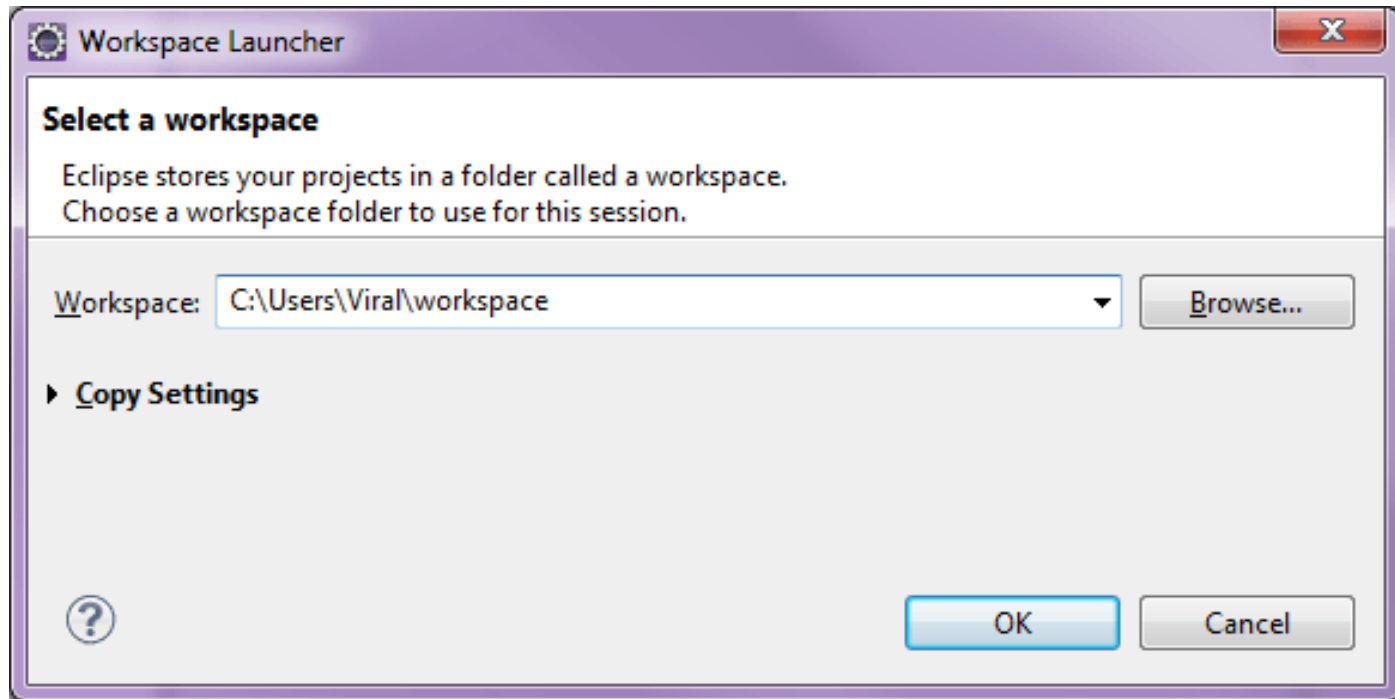


1.18. IDE - Ambiente de Desenvolvimento Integrado

- © A IDE Eclipse é um Ambiente de Desenvolvimento Integrado de código aberto e gratuito, que reúne ferramentas para apoiar o desenvolvimento de softwares em diversas linguagens de programação.



1.19. Workspace Eclipse



1.19. Mudando a perspectiva

© Window -> Perspective -> Open Perspective -> Java

1.20. Restaurando layout padrão

⦿ Window -> Perspective -> Reset Perspective

1.21. Mostrar o console

⦿ Window -> Show View -> Console

1.22. Criando um projeto

© File -> New -> Java Project

1.23. Criando uma classe

- ⦿ Botão direito na pasta “src” -> New -> Class
- ⦿ Package: deixe em branco
- ⦿ Nome da classe simples
- ⦿ Deixa o pacote em branco
- ⦿ Marca o Public Static Void Main

1.24. Mudando o tamanho da fonte

⦿ Ctrl +

⦿ Ctrl -

1.25. Nosso primeiro programa em java



> Hello, world!_

Obrigado!

Dúvidas?

Entre em contato:

heraldo.junior@ifsertao-pe.edu.br



**INSTITUTO
FEDERAL**

Sertão Pernambucano

Campus
Salgueiro