Introdução
A linguagem de programação Java

Prof. André Aparecido da Silva

O que é o Java ?

“O Java é uma linguagem de programação de propósito geral, concorrente, com base em classes e orientada a objetos. Foi projetada para ser simples o bastante para que a maioria dos programadores se torne fluente com ela.

O que é o Java ?

A linguagem Java tem relação com C e C++, mas é organizada de maneira diferente, com vários aspectos de C e C++ omitidos e algumas idéias de outras linguagens incluídas.” (Gosling et al., 2000, pág. 1)

Pequeno Histórico (1/4)

- 1991, projeto Green da *Sun Microsystem*, criar uma nova geração de computadores portáteis inteligentes, capazes de se comunicar de várias maneiras, ampliando suas potencialidades de uso.

Pequeno Histórico (1/4)

- Para tanto, decidiu-se criar também uma nova plataforma para desenvolvimento desses equipamentos, de forma que seu software pudesse ser portado para os mais diferentes tipos de equipamentos (tecnologia para aplicativos embutidos em dispositivos).

Tecnologia Java (1/2)

- **Java Platform, Standard Edition (Java SE)**

  (Aplicação padrão)
- **Java Platform, Enterprise Edition (Java EE)**

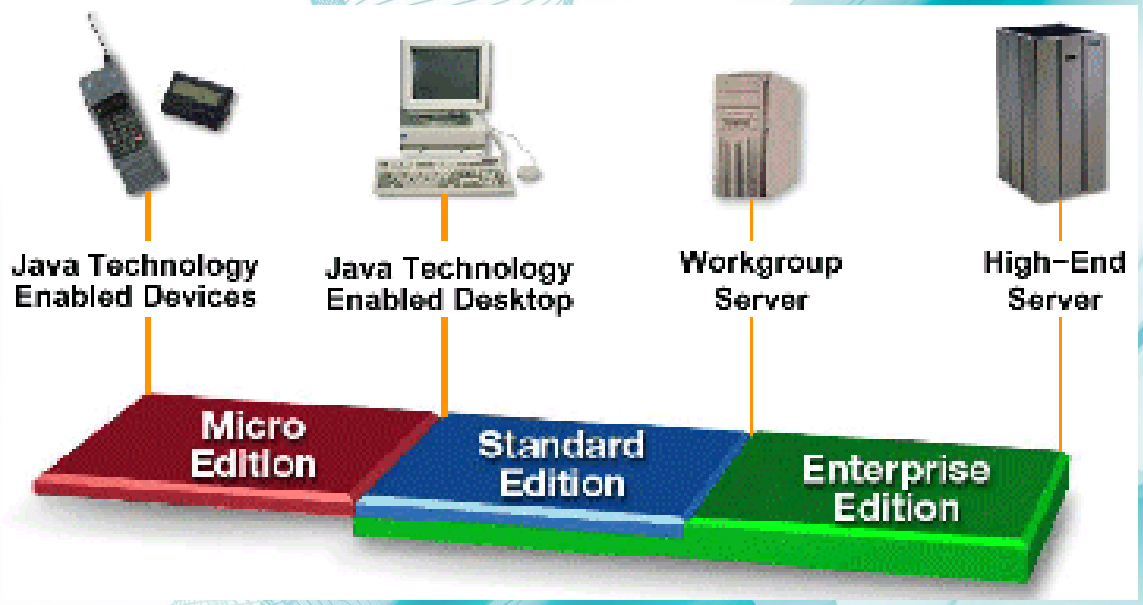
  (Aplicação empresarial)
- **Java Platform, Micro Edition (Java ME)**
  - (Aplicação para micro dispositivos, celulares, tv, controles remotos, celulares, sistemas embarcados, entre outros)

O que são estes sistemas embarcados ou embutidos?

Sistemas embarcados

São **sistemas** computacionais completos e independentes, mais simples que um computador de propósito geral (desktops), encarregados de executar apenas uma função determinada - tarefas pré-determinadas, com requisitos específicos - na qual executam geralmente repetidas vezes.

Tecnologia Java (2/2)



Características Importantes

- Orientada a objetos
- Independente de plataforma (interpretada)
- Performance
- Sem ponteiros (*garbage collector*)
- Permite *multithreading*
- Segura
- Robusta
- Simples

Java é Orientada a Objetos

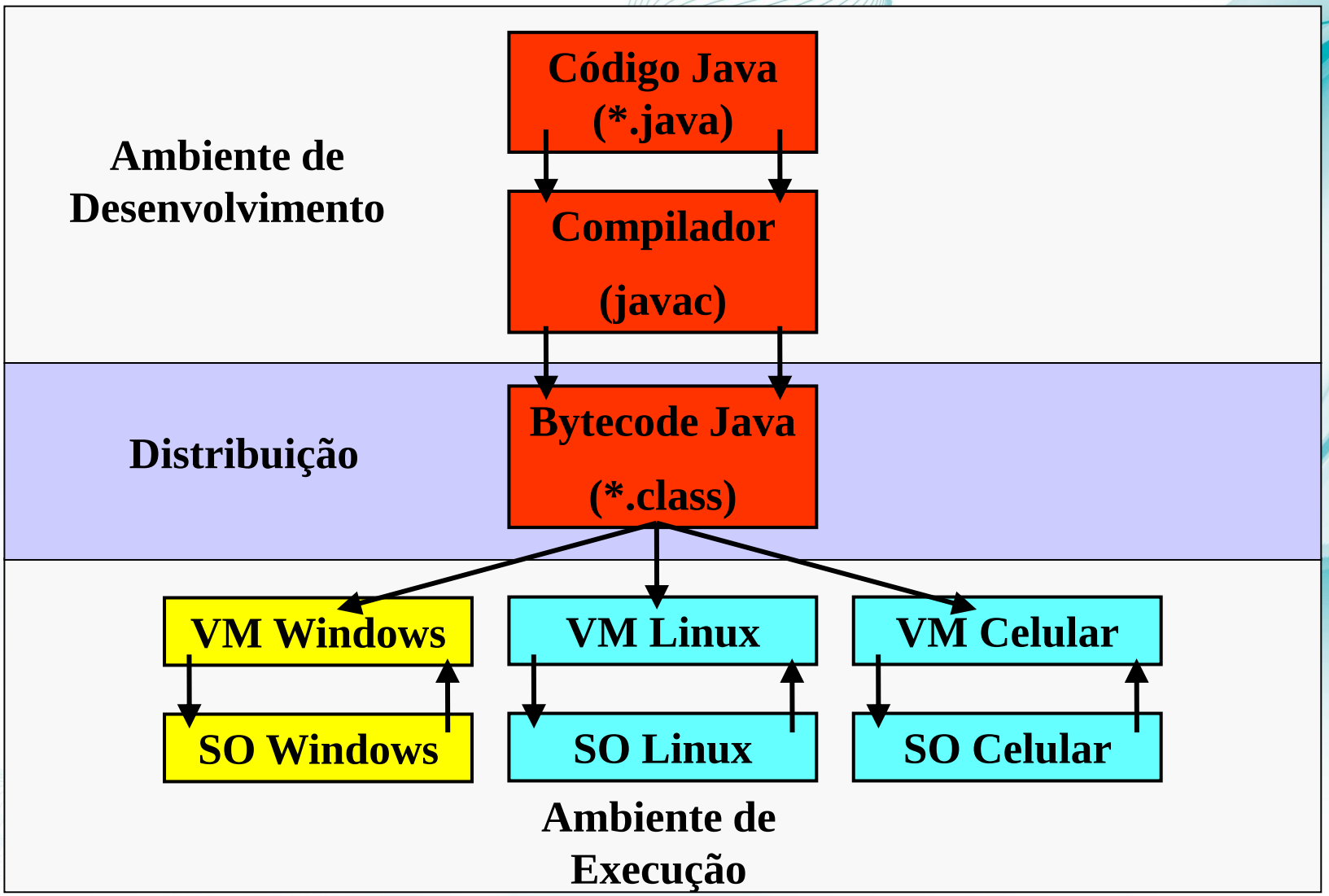
- Java é uma linguagem puramente orientada a objetos, pois, com exceção de seus tipos primitivos de dados (número, caractere e booleano), tudo em Java são classes ou instâncias (exemplares) de uma classe.

Java é Independente de Plataforma

Máquina Virtual Java (1/2)

- JVM – *Java Virtual Machine*
- *Bytecode* não é código de máquina, e sim um padrão binário (consiste de 1's e 0's) que não é "específico" para um processador.

Máquina Virtual Java (2/2)



Portável, Vantagens e Desvantagens

Vantagens

- *Bytecode* = independente de plataforma, ou arquitetura
- um programa Java pode ser executado em qualquer arquitetura (Windows, Linux, Unix, Mac) que suporte a JVM (interpretador e ambiente de execução)
- importante para aplicações distribuídas, especialmente em redes heterogêneas
- evita o desenvolvimento de versões do mesmo software específicas para cada arquitetura

Desvantagem

- linguagem interpretada = queda na performance

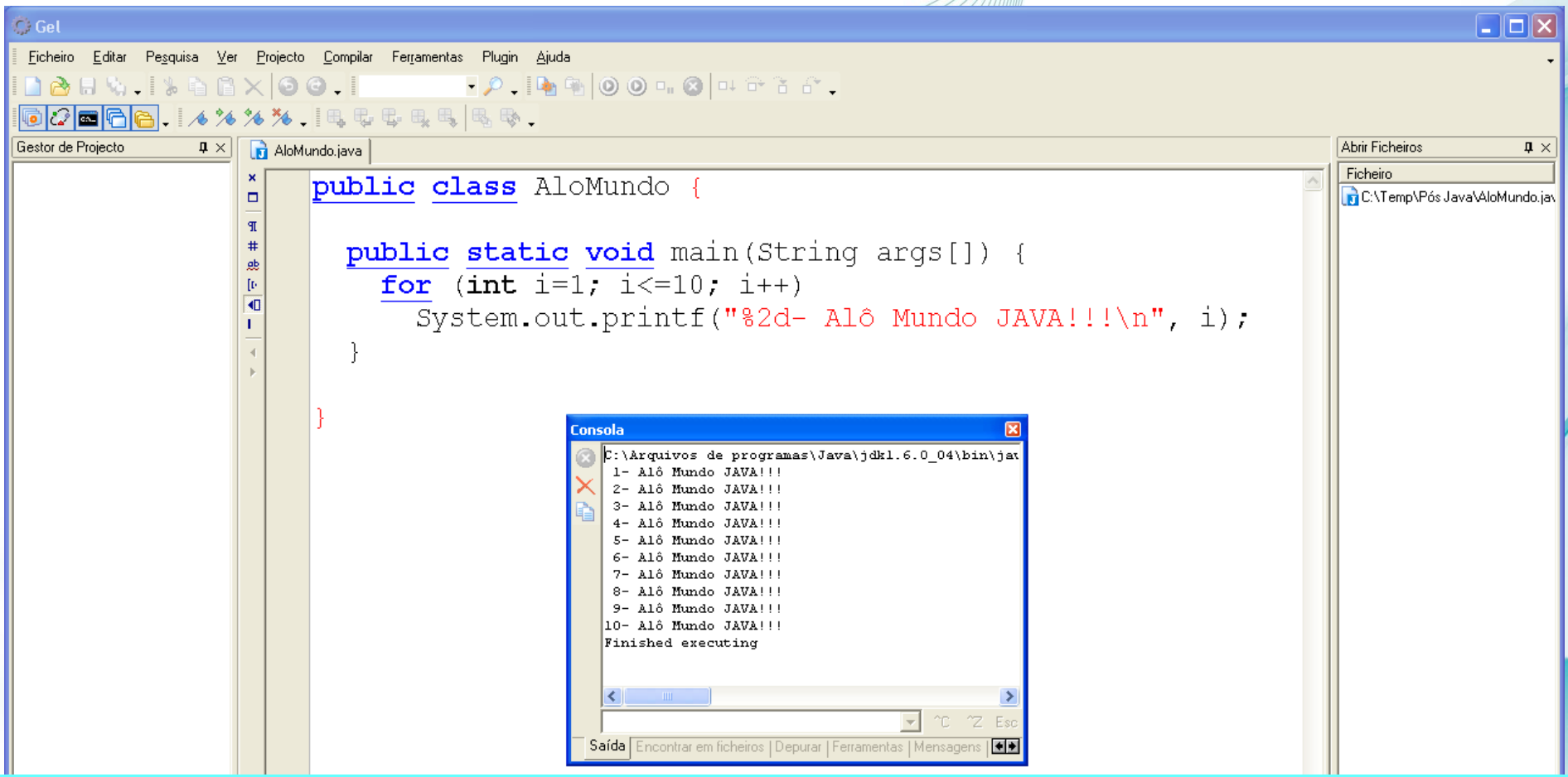
Java não Possui Ponteiros

- Java não permite a manipulação direta de endereços de memória nem exige que os objetos criados sejam explicitamente destruídos, livrando os programadores de uma tarefa complexa.

Java Permite Multithreading

Java é Seguro

Primeira Aplicação Java usando o ambiente Gel.



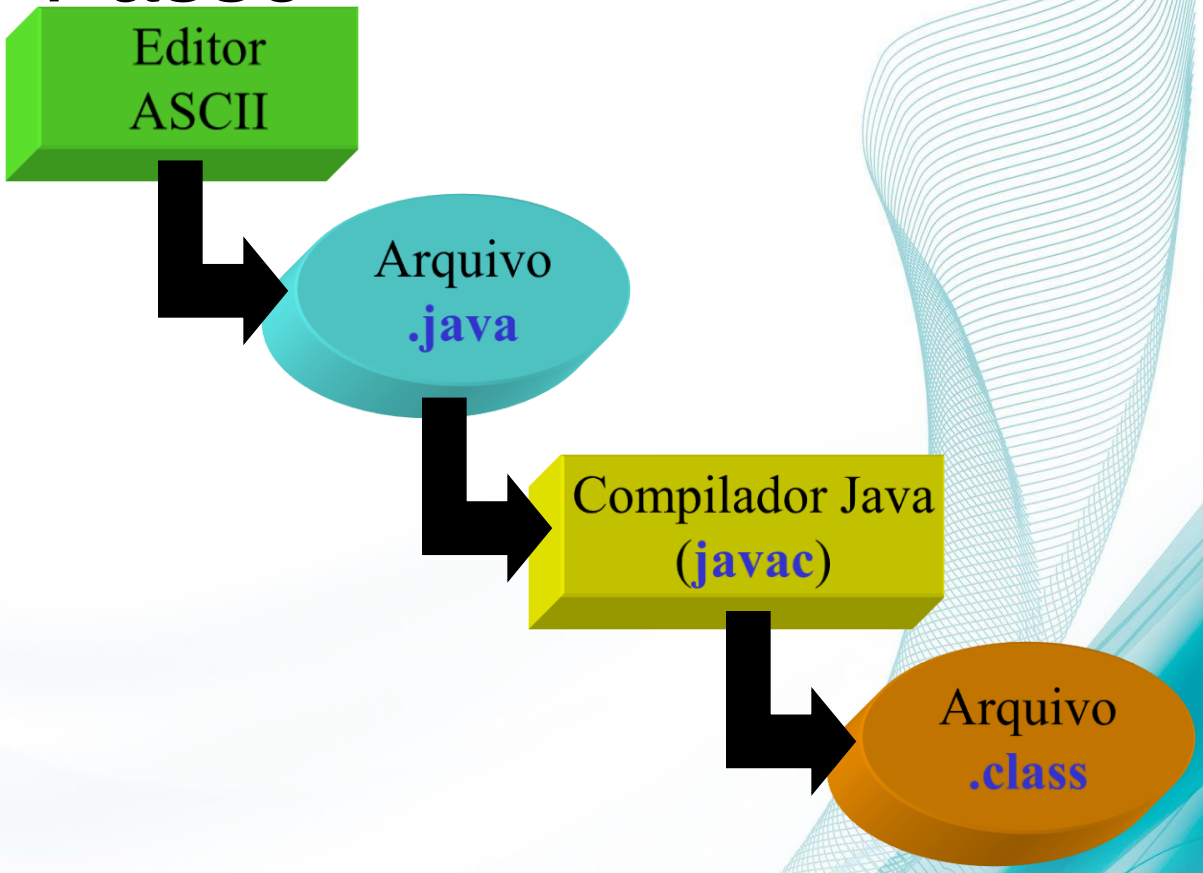
1) Ficheiro | Novo... | Ficheiro Java
2) Salvar a classe: AloMundo
3) Implementação do código fonte
4) Compilar | Compilar Ficheiro Crtl+F9
5) Compilar | Executar Ficheiro

A Criação de Aplicações Java – 1° Passo

O “primeiro passo” consiste da edição do programa por meio de um editor de textos capaz de salvar arquivos no formato ASCII (por exemplo, o bloco de notas (*Notepad*) do Microsoft Windows).

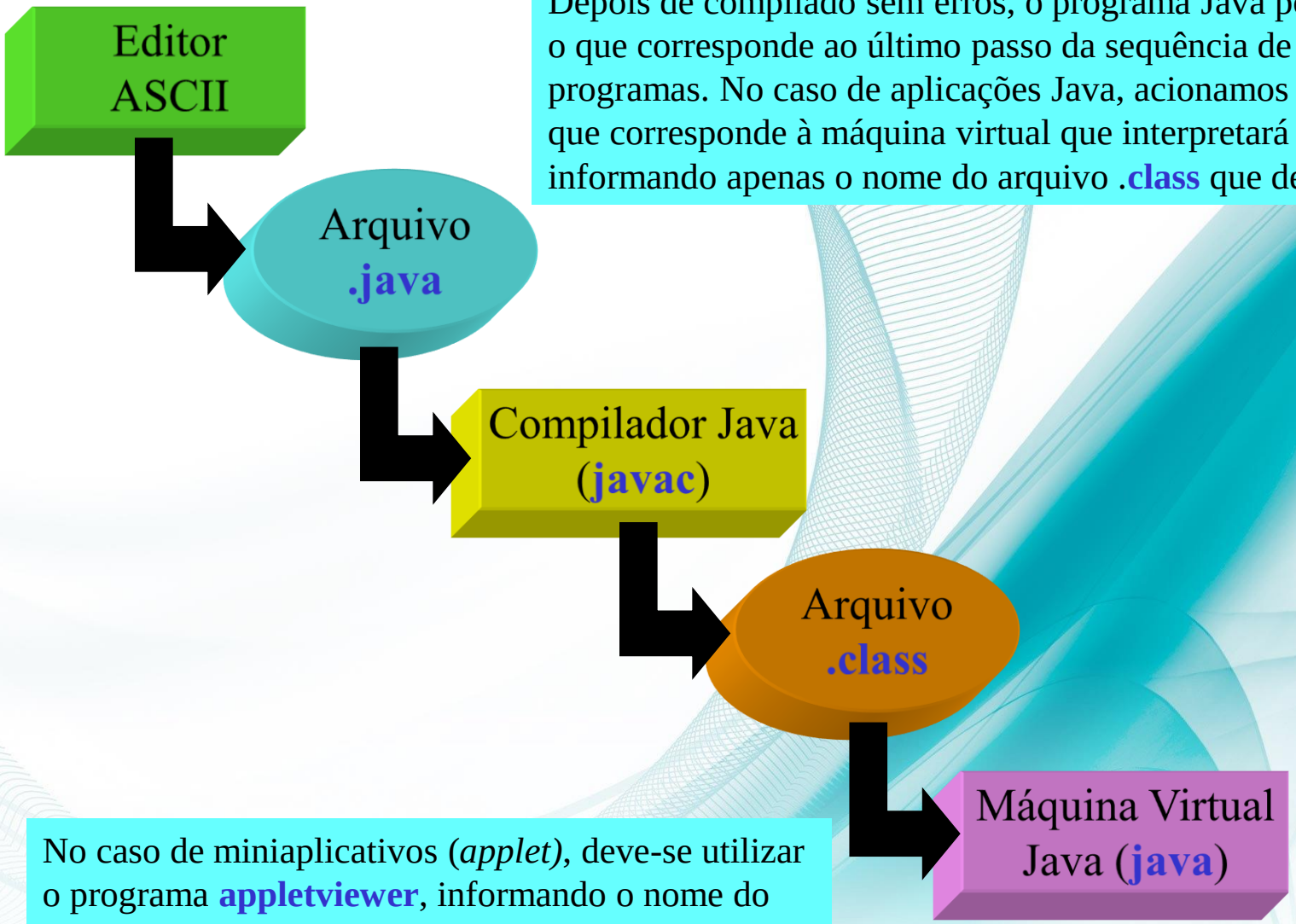
Após a edição de qualquer programa Java, os arquivos devem, obrigatoriamente ser salvos com a extensão **.java**. Tais arquivos de programa são chamados de arquivos-fonte de programa ou apenas arquivos-fonte.

A Criação de Aplicações Java – 2° Passo



O “segundo passo” é a compilação do programa, que deve ser feita pelo programa **javac** fornecido com o Kit. Não encontrando erros, o compilador **javac** transformará o arquivo-fonte em um ou mais arquivos de extensão .**class**. Cada arquivo .**class** contém *bytecodes*, formato intermediário da plataforma Java, equivalentes do programa editado. Existindo erros, os mesmos devem ser localizados e corrigidos, repetindo-se, para isso, os passos 1 e 2.

A Criação de Aplicações Java – 3° Passo



Depois de compilado sem erros, o programa Java pode ser executado, o que corresponde ao último passo da sequência de criação de programas. No caso de aplicações Java, acionamos o programa **java**, que corresponde à máquina virtual que interpretará os *bytecodes*, informando apenas o nome do arquivo **.class** que desejamos executar.

No caso de miniaplicativos (*applet*), deve-se utilizar o programa **appletviewer**, informando o nome do arquivo HTML que incorpora o *applet*.

Primeira Aplicação Java

Com o Java adequadamente instalado, podemos continuar com a construção de nossas primeiras aplicações Java. Utilizando o editor ASCII de sua preferência, digite exatamente o programa exemplificado a seguir, isto é, "observando" as letras minúsculas e maiúsculas indicadas. Não se preocupe no momento com o significado de cada uma de suas partes.
Crie um diretório apropriado para salvar o arquivo editado e garanta que seu nome seja **AloMundo.java**, respeitando mais uma vez a questão das letras minúsculas e maiúsculas. Os compiladores Java exigem que a extensão dos arquivos-fonte seja sempre **.java**.

```
public class AloMundo {

   public static void main(String args[]) {
      System.out.println("Alo Mundo !");
   }

}
```

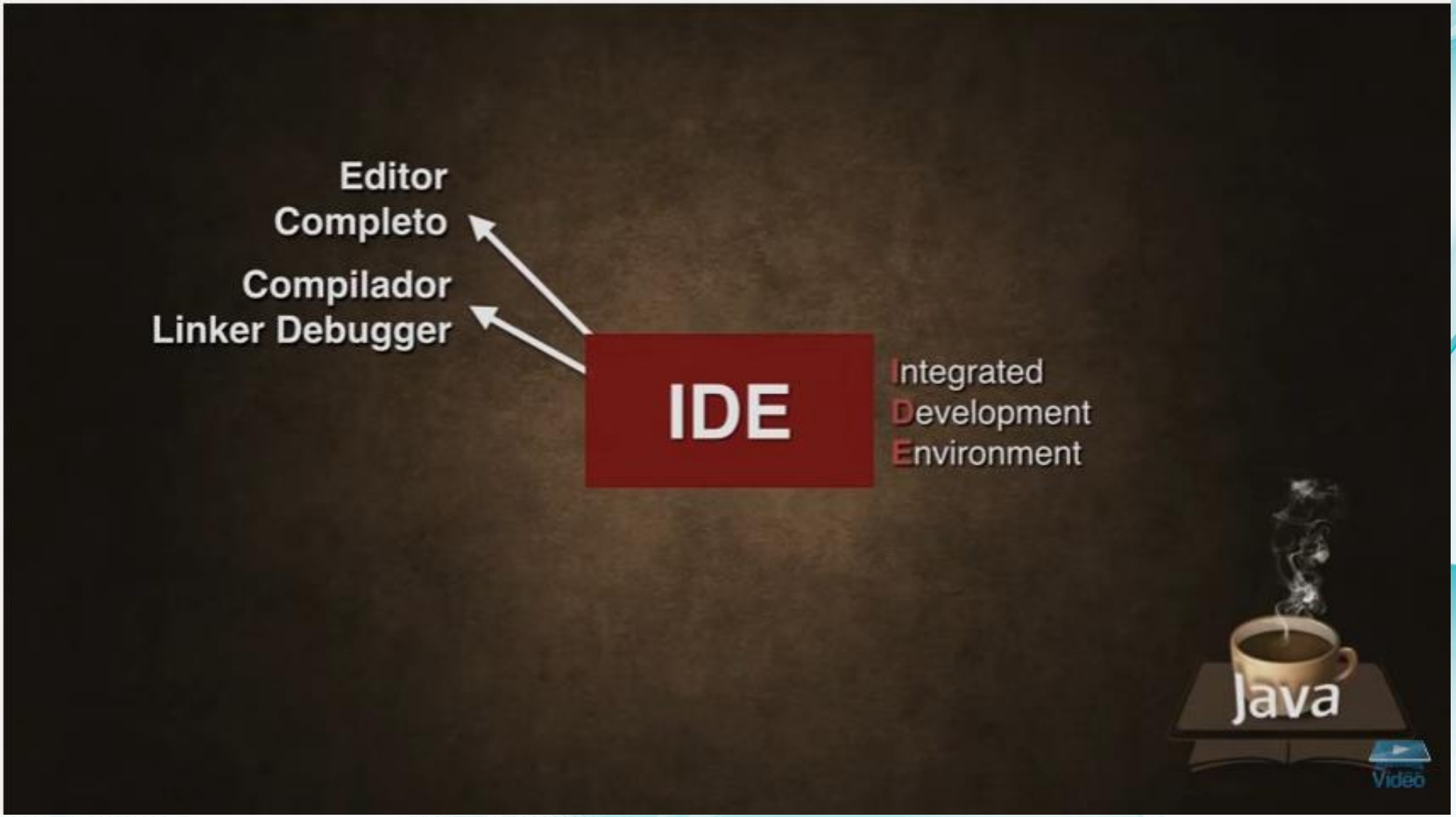
Método **main()** (1/2)

public static void main(String args[])

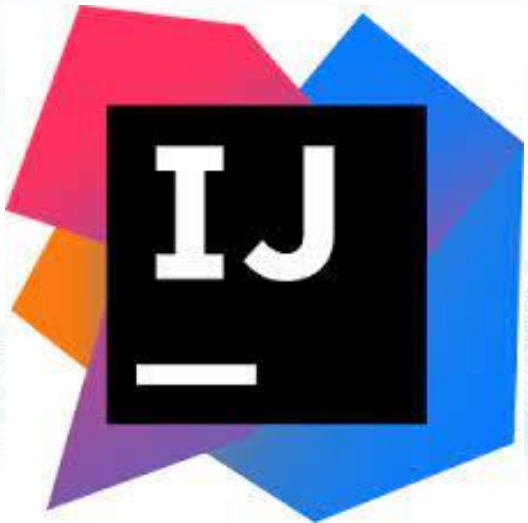
- **main()** é um método público (**public**)
- **static** – uma *keyword* que informa ao compilador que o método **main()** pode ser utilizado no contexto da classe, não exigindo uma instância para sua execução.
- **void** – indica que o método main() não retorna nada. Isso é importante porque a linguagem Java executa uma verificação de tipos cuidadosa, que inclui verificar se os métodos chamados retornam os tipos com os quais foram declarados.
- **String args[]** – declaração de um array de strings. Estes são os argumentos digitados na linha e comandos após o nome da classe: **java AloMundo arg1 arg2 ...**

Método **main()** (2/2)

- Uma aplicação Java pode conter várias classes:
  - cada classe pública dá origem a um arquivo ***.class**
  - uma delas deverá conter o método **main()** que representa o ponto inicial da execução da aplicação.
- Outras linguagens de programação, especialmente C e C++ também utilizam a declaração **main()** como o ponto inicial da execução de uma aplicação.


Editor
Completo
Compilador
Linker Debugger
IDE
Integrated
Development
Environment
Java
Video

IDEs



Rodando o **código**

O **Java** é uma linguagem híbrida, isto é, ela não é totalmente compilada, como C e C++, nem totalmente interpretada como JavaScript e Lua.

Quando escrevemos um **código** em **Java**, precisamos transformar esse **código** em algo que a máquina virtual do **Java** (JVM) consiga entender e executar o **código**.

```
//imoport as biblioctas necessárioas

public class Esqueleto_Do_Programa
{
    // FAÇA A DECLARAÇÃO DE VARIÁVEIS GLOBAIS E COMPONETES NECESSÁRIOS

    public Esqueleto_Do_Programa()
    {
        //METODO CONSTRUTOR, COLOQUE AQUI TUDO QUE SERÁ EXECUTADO
    }


    public static void main (String args [])
    {

            //CHAMADA DO METODO CONSTRUTOR
    }
}
```

ESTE MÉTODO NÃO APARECE EM TODAS AS CLASSES

PALAVRAS RESERVADAS

abstract	class	extends	implements	null	strictfp
true	assert	const	false	import	package
super	try	boolean	continue	final	instanceof
Private	switch	void	break	default	finally
int	protected	synchronized	volatile	byte	do
flota	interface	public	this	while	case
double	for	long	return	throw	catch
else	goto	native	short	throws	char
enum	if	new	static	transient	

TIPOS DE VARIÁVEIS

As **variáveis** de instância de **tipo** primitivo são inicializadas por padrão, as **variáveis** dos **tipos** byte, char, short, int, long, float e double são inicializadas como 0, e as **variáveis** do **tipo** boolean são inicializadas como false.

Caracteríscas gerais

```
1 public class MyClass{
2 // código vai aqui
3 }
```

Caracteríscas gerais

- Os arquivos devem ser salvos com a extensão .java
- Após compilado será gerado um arquivo com a extensão .java
- As instruções terminam com ponto e virgula;
- O inicio e final de blocos de códigos são definidos por abre e fecha chaves
- Os arquivos e variáveis podem ter qualquer nome que não usem palavras reservadas
- Os arquivos e variáveis não podem podem ter acentuação no seu nomes

Exemplo de código

```
  /** * @(#)Text1.java * * * @author * @version 1.00
2020/8/20 */

public class Calculadora

{
        public Calculadora()
        {
                Operadores_Matematicos   calc1 = new
Operadores_Matematicos ();
        }
        public static void main (String args[])
        { new Calculadora(); }
}
```

Exemplo de código

```
import java.util.*;
public class Delta {

    Scanner read = new Scanner(System.in);
    int a, b, c;
    double delta, x1, x2;
    public Delta()
      {deltaCalculo();}
    private void deltaCalculo() {

        System.out.println("Digite o valor de A, B e C respectivamente: ");
        a = Integer.parseInt(read.next());
        b = Integer.parseInt(read.next());
        c = Integer.parseInt(read.next());
        delta = (b * b) + (-4 * (a * c));
        System.out.println("Delta: " + delta);
        if (delta >= 0)
            {calculaX(); }
        else
        {
            System.out.println("Delta não possui raiz!");
            System.exit(0);
        }
    }
    private void calculaX()
    {
        x1 = (double) ((-(b) + Math.sqrt(delta)) / 2 * a);
        x2 = (double) ((-(b) - Math.sqrt(delta)) / 2 * a);
        System.out.println("x1 = " + x1);
        System.out.println("x2 = " + x2);
    }
    public static void main(String[] args)
        {new Delta();  }
}
```

Exemplos Variáveis

- char turma = ‘a’;
- String Nome = “Ana Carolina Bertoldi”;
- int idade = 42;
- double salario = 4900.32;
- boolean status = true;

Vamos criar a duas classes:

1)A primeira exibe conta e exibe na tela até os números 0 a 1000

2) A segunda testa um número conforme a conjectura de Collatz