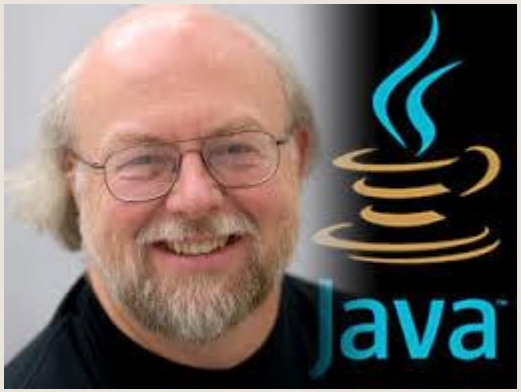
Linguagem Java
Introdução



James Gosling, criador da linguagem Java

Prof. José Augusto Cintra
http://www.josecintra.com/blog

Por que usar Java?

- Java é multiplataforma, ou seja, um programa escrito em Java pode ser executado em qualquer plataforma sem necessidade de alterações no código fonte.

- Java é uma arquitetura aberta, extensível, com várias implementações, o que a torna independente do fornecedor.

- Robusta e segura

- Java pode ser baixada gratuitamente.

Por que usar Java?

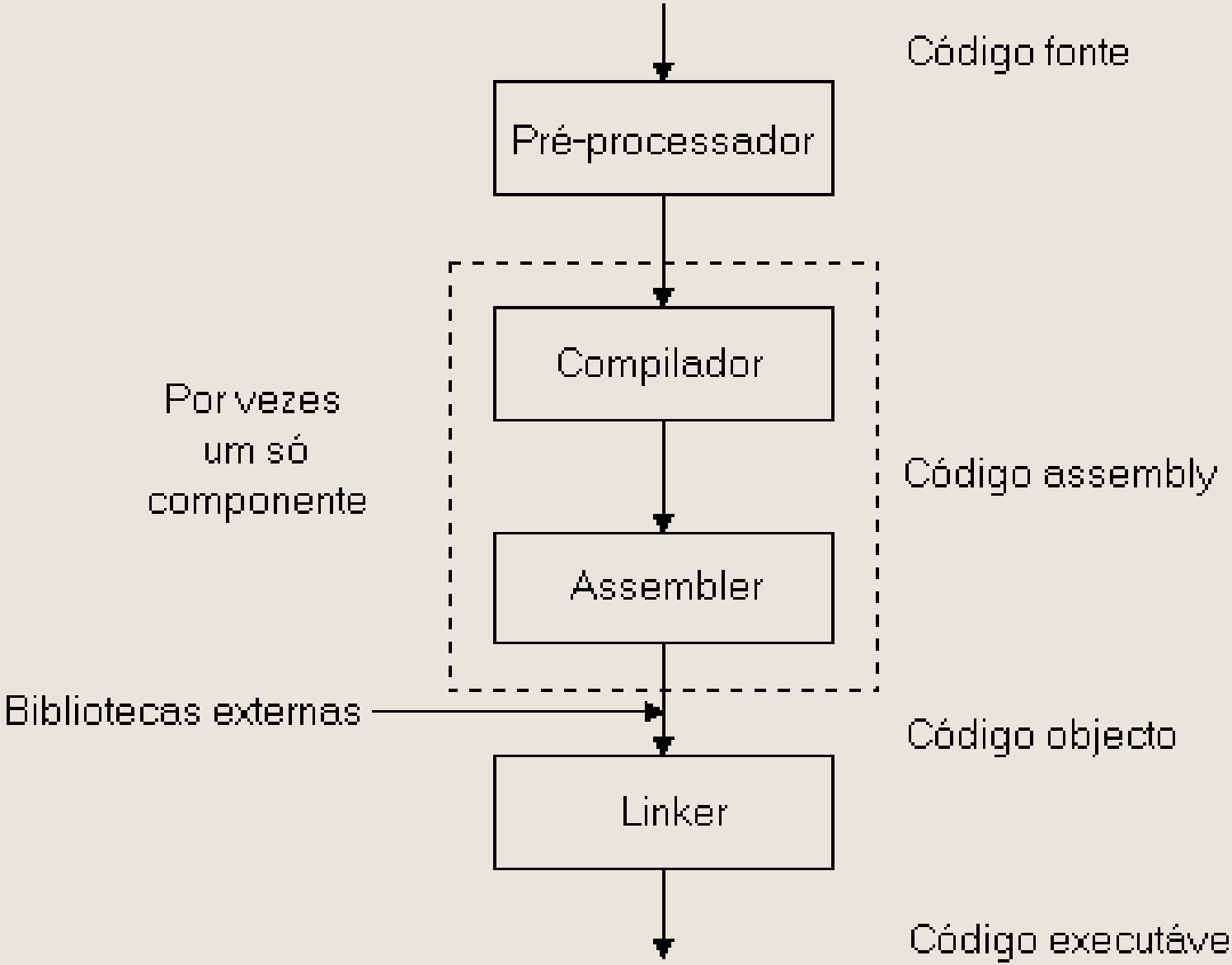
- Linguagem Independente de plataforma que pode ser utilizada em vários produtos eletrônicos, tais como torradeiras e refrigeradores.

- Orientação a objetos com forte suporte a técnicas adequadas de engenharia de software

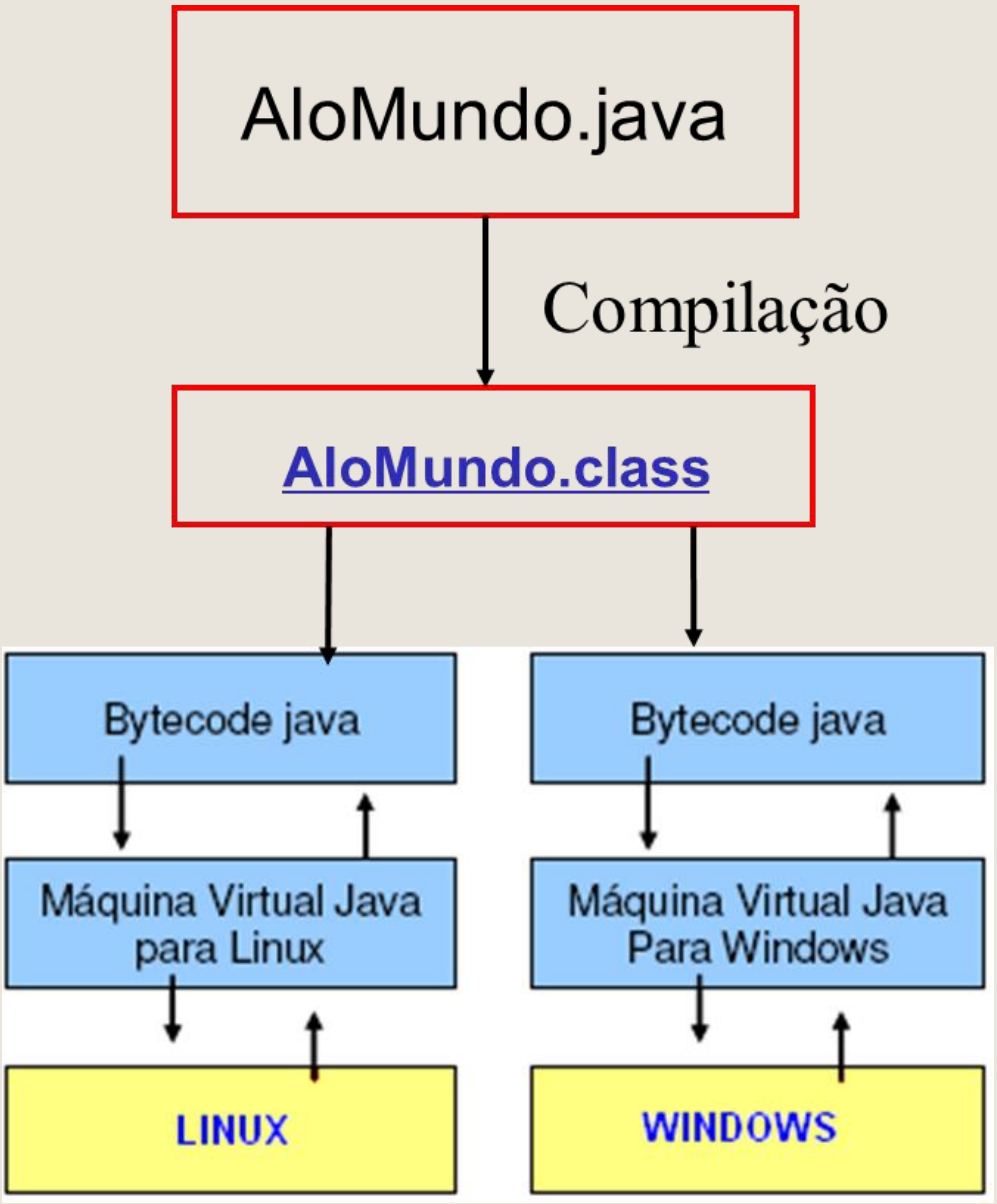
- Sintaxe simples baseada na linguagem C

- Java é uma das linguagens de desenvolvimento de softwares mais utilizadas no Mundo

Processo de compilação em C

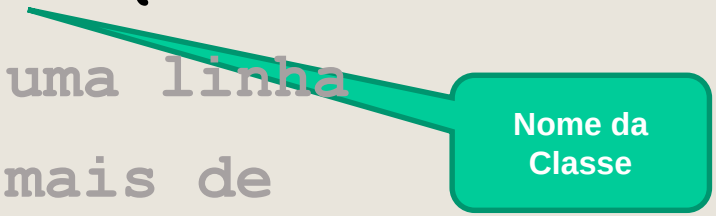


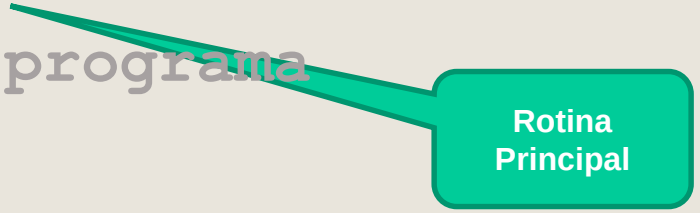
Processo de compilação Java



Estrutura do Programa em Java

```
public class AloMundo {
   // Comentário de uma linha
   /* Comentário de mais de
      uma linha */
   /** Comentário de documentação */
   public static void main (String[] args) {
      // Código fonte do programa
   }
}
```





Saída de Dados

Para saída dos dados podemos usar um dos comandos:

System.out.print()

System.out.println()

Saída de Dados

```
public class AloMundo {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        System.out.println("Alo Mundo");
    }

}
```

System.out

É a saída padrão do sistema

A mensagem

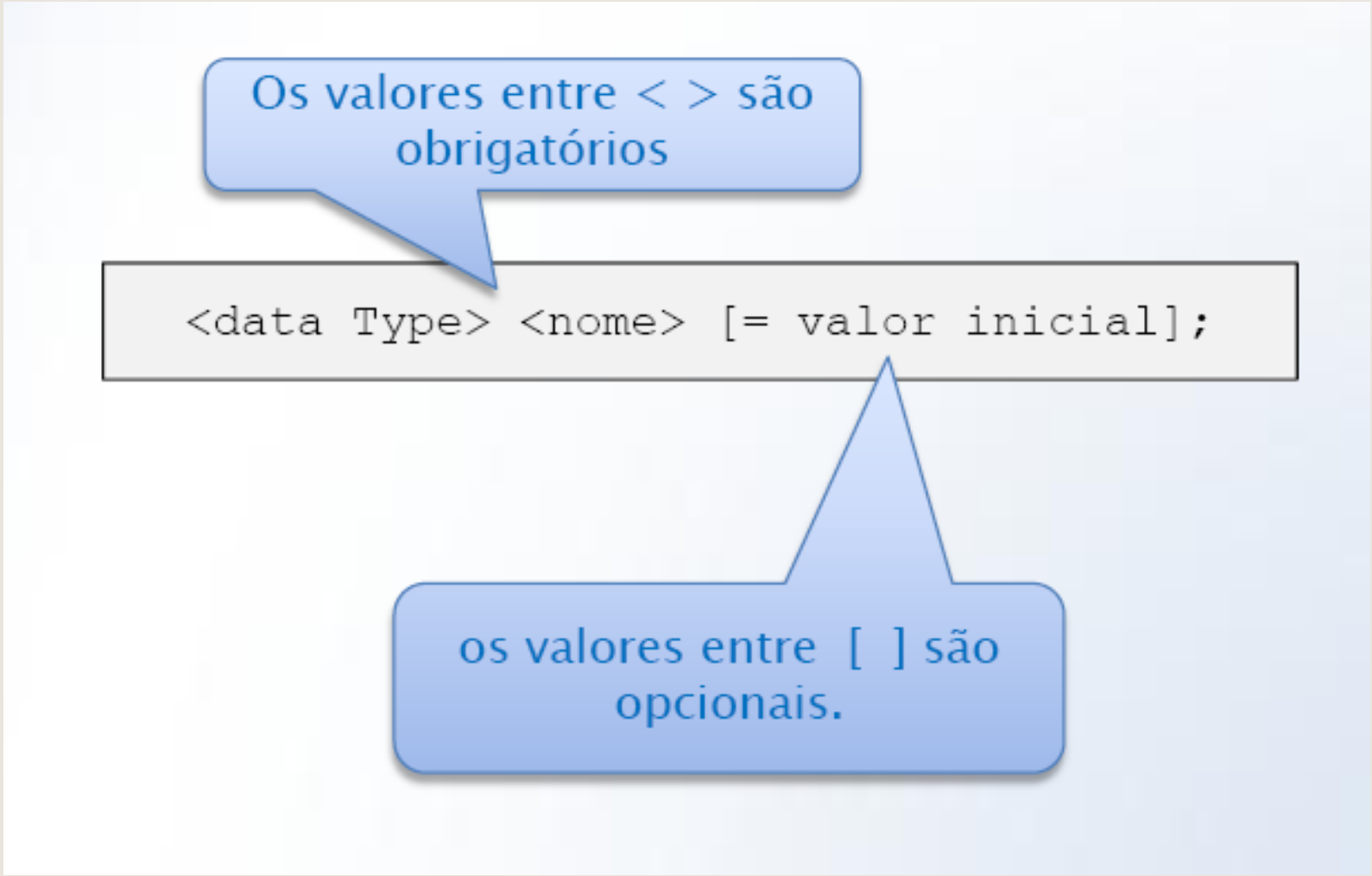
(Expressão)

Tipos de Dados

Tipos de dados primitivos

- Existem oito tipos de dados primitivos.
  - Lógico
    - boolean..... 08 bits
  - Caracter
    - char..... 16 bits
  - Inteiro
    - byte..... 08 bits
    - short..... 16 bits
    - int..... 32 bits
    - long..... 64 bits
  - Ponto flutuante
    - float..... 32 bits
    - double..... 64 bits

Declaração de variáveis



Exemplos

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

    char option;

    option  = 'c';

    int x = 35;

    double f = 3.5;

    boolean r = true;


    System.out.println("Alo Mundo");
```

Convenções para nomes

Embora não seja de uso obrigatório, existe a convenção padrão para atribuir nomes em Java, como:

- Nomes de classes são iniciados por letras maiúsculas;
- Nomes de métodos, atributos e variáveis são iniciados por letras minúsculas;
- Em nomes compostos, cada palavra do nome é iniciada por letra maiúscula, as palavras não são separadas por nenhum símbolo.

Operadores

Operador	Uso	Descrição
+	op1 + op2	Soma op1 e op2
*	op1 * op2	Multiplica op1 por op2
/	op1 / op2	Divide op1 por op2
%	Op1 % op2	Calcula o resto da divisão op1 por op2
-	op1 - op2	Subtrai op2 de op1

Operadores de incremento

Operador	Uso	Descrição
++	op++	Incrementa 1 em op, avalia o valor de op antes de incrementar
++	++op	Incrementa 1 em op, avalia o valor de op depois de incrementar
--	op--	Decrementa 1 em op, avalia o valor de op antes de decrementar
--	--op	Decrementa 1 em op, avalia o valor de op depois de decrementar

Operadores Relacionais

Operador	Uso	Descrição
>	op1 > op2	op1 é maior que op2
>=	op1 >= op2	op1 é maior ou igual a op2
<	op1 < op2	op1 é menor que op2
<=	op1<= op2	op1 é maior ou igual a op2
==	op1 == op2	op1 é igual a op2
!=	op1 != op2	op1 é diferente de op2

Operadores lógicos

Operador	Descrição
&&	AND lógico
&	AND lógico booleano
||	OR lógico
|	OR lógico booleano inclusivo
^	OR lógico booleano exclusivo
!	NOT lógico

Entrada de Dados

Pode ser usada a classe Scanner do pacote java.util

- Retorna entradas no teclado nos tipos primitivos: String, int, double, float, etc
- Métodos
  - next()
  - nextInt()
  - nextDouble()
  - nextFloat()

Exemplo de Entrada de Dados

```
import java.util.Scanner;

public class GetInputFromKeyboard {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        String nome = "";

        System.out.printf("Digite o seu nome " );

        nome = input.next();

        System.out.printf("\nMeu nome é %s",nome);
    }

}
```

Indica que queremos utilizar a classe Scanner

Cria uma variável do tipo Scanner

Recebe a entrada do usuário

Conversão de Dados

É possível converter String para qualquer tipo primitivo

int - Integer.parseInt(string)

Float - Float.parseFloat(string)

Double - Double.parseFloat(string)

Da mesma forma é possível converter tipos primitivos para String

Exemplo de Conversões

```
import java.util.Scanner;

public class AloMundo {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        String nome = "";

        System.out.printf("Digite o seu nome " );

        nome = input.next();

        System.out.printf("Digite sua idade " );
        int idade = Integer.parseInt( input.next() );

        System.out.printf("\nMeu nome é %s",nome);
        System.out.printf("\nTenho é %d de idade",idade);
    }

}
```

Exemplo

```
// Calcula a área de um círculo dado seu raio
import java.util.*;
public class AreaCirculo {
  public static void main(String[] args) {
    double area, raio;
    Scanner teclado = new Scanner(System.in);
    System.out.print("Informe o raio do círculo: ");

    raio = teclado.nextDouble());
    area = Math.PI * Math.pow(raio, 2);

    System.out.println("Área do círculo = " + area);
  }
}
```

Estruturas de controle

- Controlam à ordem em que instruções, expressões e chamadas de função são executadas ou avaliadas.

- Dividem-se em:

- Estruturas de seleção → Desvia a execução do fluxo de acordo com a avaliação de uma condição (verdadeira ou falsa)
- Estruturas repetição → Repete um bloco de código, permitindo, entre outras coisas, a iteração em uma coleção de dados

Estruturas de seleção

- Java possui as seguintes estruturas de seleção:
  - If / Else
  - switch

Estrutura IF

- Especifica que um comando ou bloco será executado se e somente se uma determinada condição booleana for verdadeira

Exemplo:

```
If (idade < 18) {
   System.out.print("Entrada não Permitida");
}
Else {
   System.out.print("Entrada Permitida");
}
```

Estrutura SWITCH

- **Permite a multiplicidade de escolha**

Exemplo:

```
int op = 3;

switch( op ){

    case 1:
        System.out.println("Você escolheu a opção 1");
        break;
    case 2:
        System.out.println("Você escolheu a opção 2");
        break;
    case 3:
        System.out.println("Você escolheu a opção 3");
        break;
    default:
        System.out.println("Opção inválida");
        break;
```

Estrutura de Repetição

Permite executar um bloco de instruções um número determinado de vezes

- **while**
- **do – while**
- **for**

Estrutura de Repetição

Observações:

- **A expressão lógica é avaliada antes de cada repetição do laço. Enquanto seu resultado for VERDADEIRO, a sequência de comando será executada. Por isso é chamada de condição de parada.**
- **Normalmente é usada uma ou mais variáveis para compor a condição de parada. Essas variáveis são chamadas de variáveis de controle**
- **Para que o laço tenha fim, a condição de parada, em algum momento deve ser atendida, caso contrário, teremos um laço de repetição infinito**

Estrutura WHILE

Os comandos no laço while são executados enquanto uma condição booleana for verdadeira

Exemplo:

```
int x = 1;
while (x <= 10) {
   system.out.println(x);
   x++;
}
```

Valor inicial
x é a variável de controle

Condição de parada → Valor final

Instrução que controla a iteração

Pergunta: O que irá acontecer se inicializarmos o valor da variável x com o valor 11?

Estrutura DO WHILE

Semelhante ao WHILE, a principal diferença é que os comandos são executados pelo menos uma vez

Exemplo:

```
int x = 1;
do {
   system.out.println(x);
   x++;
} while (x <= 10)
```

Pergunta: O que irá acontecer se inicializarmos o valor da variável x com o valor 11?

Estrutura FOR

Semelhante ao WHILE, a principal diferença é que as estruturas de controle ficam todas definidas num mesmo ponto (os parênteses).

Exemplo:

```
int x;
for (x = 1; x <= 10; x++){
   System.out.println(x);
}
```

Valor inicial
x é a variável de controle

Condição de parada → Valor final

Instrução que controla a iteração

FIM

Esta aula foi compilada a partir do trabalho do prof. Bruno Correa.

Outros links úteis:

http://www.wilson.kinghost.net/POO/java_basico.ppt

http://java.icmc.usp.br/resources/ebooks/tutorial_java.ppt

Consulte o blog para materiais complementares e exercícios resolvidos

- **http://www.josecintra.com/blog**