



# FUNDAMENTOS DE JAVA

Prof. Mário Meireles Teixeira

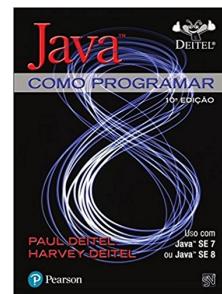
DEINF – UFMA

mario@deinf.ufma.br



## Tópicos da Disciplina

- Fundamentos de Java
- Introdução a Orientação a Objetos
- Vetores e Listas em Java
- Herança e Polimorfismo
- Tratamento de Exceções
- Manipulação de Arquivos
- Classes Abstratas e Interfaces
- Generics, Set, Map
- Desenvolvimento de projetos de programação



Fundamentos de Java



# Introdução

Fundamentos de Java

# Tecnologia Java

- **Java** = linguagem de programação de alto nível + plataforma de execução de aplicações
- **Java como linguagem de programação:**
  - Aplicativos locais (programas Java) e distribuídos (sockets, RMI)
  - Aplicativos que executam em browsers (Applets)
  - Aplicativos corporativos de grande porte e Páginas Web dinâmicas (Servlets, JSP, JSF, Web Services → WebApps)
  - Aplicativos para dispositivos móveis (Java ME, Android)
- **Java como ambiente de execução:**
  - Ambiente neutro (JRE – **Java Runtime Environment**) para diferentes plataformas: SO's, browsers, celulares, tablets...

Fundamentos de Java

5

# Histórico

- Criada pela Sun em 1995; adquirida pela Oracle em 2010
- Vantagens em relação a suas predecessoras:
  - Independente de plataforma (**portabilidade** do código)
  - Fortemente tipada
  - Gerenciamento dinâmico de memória (coleta de lixo)
  - Robusta
  - Segura
  - Código compilado para **bytecodes** e executado em máquina virtual
  - Suporte a diferentes dispositivos

Fundamentos de Java

6

## Edições de Java

- Java possui diversas plataformas para desenvolvimento e execução de aplicações ([java.com](http://java.com))
- As principais APIs Java são distribuídas pela Oracle:
  - **Java Standard Edition (Java SE)** – desktops e servidores
    - <http://www.oracle.com/technetwork/java/javase>
  - **Java Enterprise Edition (Java EE)** – aplicações corporativas e de Internet
    - <http://www.oracle.com/technetwork/java/javaee>
  - **Java Micro Edition (Java ME)** – dispositivos móveis e embarcados (IoT)
    - <http://www.oracle.com/technetwork/java/javame>
  - **Android SDK**: API de propósito específico, durante muitos anos foi o padrão para desenvolvimento de aplicativos Android (API do [Google](#), não da Oracle)

Fundamentos de Java

7

## Compilação e Interpretação

- **Compilação**
  - geração de código executável; dependente de plataforma
  - tradução lenta X execução rápida
  - Ex: FORTRAN, C, C++
- **Interpretação pura**
  - sem geração de código
  - execução lenta; independente de plataforma
  - Ex: Pascal, PHP, JavaScript
- **Híbrida**
  - geração de código intermediário; independente de plataforma
  - tradução rápida X execução não tão rápida
  - Ex: Java, C#, Python

Fundamentos de Java

8

# Modelo de execução

```
package hello;
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Compilador

javac

Bytecode  
(representação  
intermediária)

Tradução Just-in-  
Time (JIT)

Interpretador (JVM)  
java

Código nativo  
(de máquina)

- JDK = Java Development Kit
  - javac, java, javadoc,...
- JRE = Java Runtime Environment
  - java (somente execução)

Fundamentos de Java

9

# Modelo de execução

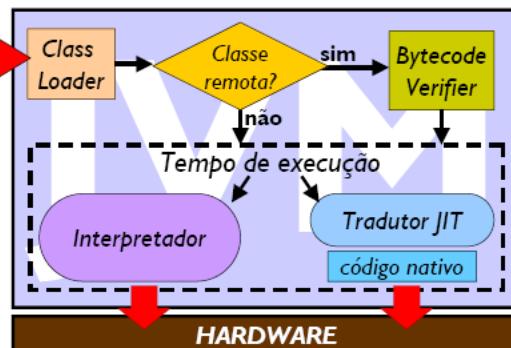
HelloWorld.java

javac

bytecode  
HelloWorld.class

java

**Tradutor JIT:**  
Just-In-Time Compiler.  
Gera código nativo a  
partir de bytecodes para  
maior performance



Fundamentos de Java

10

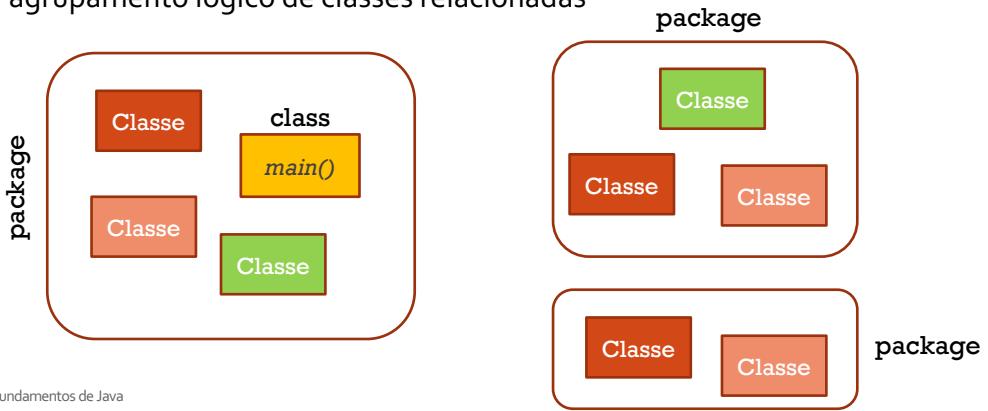


# Estrutura de uma aplicação Java

Fundamentos de Java

## Aplicação Java

- Uma aplicação Java é composta por uma ou mais classes. Um pacote é um agrupamento lógico de classes relacionadas



12

Fundamentos de Java

## Ambientes de Desenvolvimento (IDEs)



[netbeans.org](http://netbeans.org)



[www.eclipse.org](http://www.eclipse.org)



[www.bluej.org](http://www.bluej.org)

Fundamentos de Java

13

## Primeiro Programa

Bemvindo.java

```
class Bemvindo{
    public static void main(String[] args) {
        System.out.println("Bem vindo!");
    }
}
```

- Atenção a maiúsculas e minúsculas
- Arquivo **.java** tem que ter o mesmo nome de uma de suas classes

Método **main()**: aqui se inicia a execução do programa

Fundamentos de Java

14

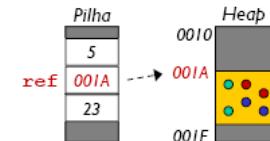
## Instanciando um objeto

```
class Bemvindo {
    void imprime() {
        System.out.println("Bem vindo!");
    }

    public static void main(String[] args) {
        Bemvindo obj = new Bemvindo();
        obj.imprime();
    }
}
```

Chamada de método

Criação de objeto



`obj` = referência para um objeto

Fundamentos de Java

15

## Usando um método estático

Não recomendável

```
class Bemvindo {
    static void imprime() {
        System.out.println("Bem vindo!");
    }

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

Fundamentos de Java

16

## Duas classes: composição

Bemvindo.java

```
class Bemvindo{
    void imprime(){
        System.out.println("Bem vindo!");
    }
    public static void main(String[] args){
        Bemvindo obj = new Bemvindo();
        Welcome outro = new Welcome();
        obj.imprime();
        outro.imprime();
    }
}
```

Welcome.java

```
class Welcome{
    void imprime(){
        System.out.println("Welcome!");
    }
}
```

```
class Bemvindo3{
    imprime()
    main(...)
}
```

```
class Welcome{
    imprime()
}
```

Fundamentos de Java

17



## Tipos primitivos em Java

Fundamentos de Java

## Identificadores

- São palavras utilizadas para nomear variáveis, métodos e classes
- Em Java, os nomes devem começar por letra ou sublinhado \_
  - Não podem começar com dígitos
  - Diferenciam letras maiúsculas e minúsculas
- Identificadores válidos:
  - Nomes de pacotes, atributos, variáveis, métodos: `conta, nomeAluno, calcHorasExtras, _var1, rendaMedia`
  - Nomes de classes: `NotaFiscal, Conta, AlunoGrad`

Fundamentos de Java

19

## Palavras reservadas em Java

<code>abstract</code>	<code>boolean</code>	<code>break</code>	<code>byte</code>	<code>case</code>
<code>catch</code>	<code>char</code>	<code>class</code>	<code>continue</code>	<code>default</code>
<code>do</code>	<code>double</code>	<code>else</code>	<code>extends</code>	
<code>final</code>	<code>finally</code>	<code>float</code>	<code>for</code>	<code>if</code>
<code>implements</code>	<code>import</code>	<code>instanceof</code>	<code>int</code>	<code>interface</code>
<code>long</code>	<code>native</code>	<code>new</code>	<code>return</code>	<code>package</code>
<code>private</code>	<code>protected</code>	<code>public</code>	<code>synchronized</code>	<code>short</code>
<code>static</code>	<code>super</code>	<code>switch</code>	<code>this</code>	<code>try</code>
<code>throw</code>	<code>throws</code>	<code>transient</code>	<code>void</code>	<code>volatile</code>
<code>while</code>	<code>enum</code>			

- Todas as palavras reservadas são escritas com letras minúsculas
- `true, false` e `null` são literais reservados

Fundamentos de Java

20

## Tipos de dados primitivos

- Java, assim como C e C++, é uma linguagem **fortemente tipada**:
  - todas as variáveis devem ter um tipo previamente declarado (assumem valores default se não inicializadas)
- Diferentemente de C e C++, os tipos primitivos em Java são **portáteis** entre todas as plataformas

Fundamentos de Java

21

## Tipos primitivos em Java

Tipo	Tamanho	Mínimo	Máximo	Default	'Wrapper'
<code>boolean</code>	—	—	—	<code>false</code>	<code>Boolean</code>
<code>char</code>	16-bit	Unicode 0	Unicode $2^{16} - 1$	<code>\u0000</code>	<code>Character</code>
<code>byte</code>	8-bit	-128	+127	0	<code>Byte</code>
<code>short</code>	16-bit	$-2^{15}$	$+2^{15} - 1$	0	<code>Short</code>
<code>int</code>	32-bit	$-2^{31}$	$+2^{31} - 1$	0	<code>Integer</code>
<code>long</code>	64-bit	$-2^{63}$	$+2^{63} - 1$	0	<code>Long</code>
<code>float</code>	32-bit	IEEE754	IEEE754	0.0	<code>Float</code>
<code>double</code>	64-bit	IEEE754	IEEE754	0.0	<code>Double</code>
<code>void</code>	—	—	—	—	<code>Void</code>

Fundamentos de Java

22

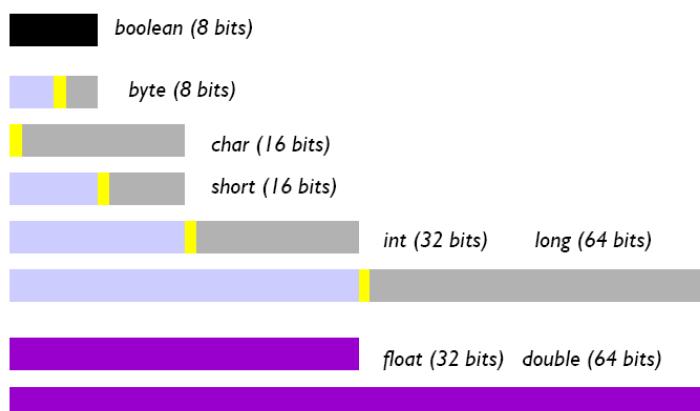
# Exemplos

- *Literais de caractere:*  
`char c = 'a';  
char z = '\u0041'; // em Unicode`
- *Literais inteiros*  
`int i = 10; short s = 15; byte b = 1;  
long hexa = 0x9af0L; int octal = 0633;`
- *Literais de ponto-flutuante*  
`float f = 123.0f;  
double d = 12.3;  
double g = .1e-23;`
- *Literais booleanos*  
`boolean v = true;  
boolean f = false;`
- *Literais de string (não é tipo primitivo - s é uma referência)*  
`String s = "abcde";`
- *Literais de vetor (não é tipo primitivo - v é uma referência)*  
`int[] v = {5, 6};`

Fundamentos de Java

23

# Tipos de dados

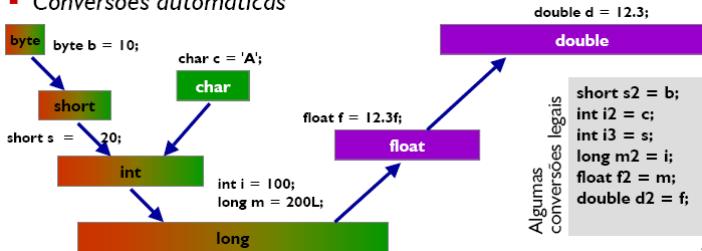


Fundamentos de Java

24

## Conversão entre tipos

- Java converterá um tipo de dados em outro sempre que isto for apropriado
- As conversões ocorrerão automaticamente quando houver garantia de não haver perda de informação
  - Tipos menores em tipos maiores
  - Tipos de menor precisão em tipos de maior precisão
  - Inteiros em ponto-flutuante
- Conversões automáticas



25

## Conversão entre tipos

- É necessário fazer a **conversão explícita (cast)** de um tipo de maior precisão para um de menor precisão:
 

▪ long bigval = 100;	// 100 é do tipo int, OK
▪ int smallval = 1502341L;	// 1502341 é long, illegal
▪ byte b = (byte) 2334;	// cast obrigatório, perda de dados, OK?
▪ boolean flag = 1;	// illegal, use true ou false
▪ float num = 3.23;	// 3.23 é double, illegal
▪ double num = 10 / 2.5;	// resultado é double, OK
▪ float num = (float) 10 / 2.5;	// resultado é double, cast obrigatório
▪ float num = 10 / 2;	// resultado é int, OK, mas é o que se quer?

Fundamentos de Java

26



# Operadores

Fundamentos de Java

## Operadores Aritméticos

- `+` adição
- `-` subtração
- `*` multiplicação
- `/` divisão
- `%` módulo (resto)
  
- Operadores unários
  - `-n` e `+n` (ex: `-23`) (em uma expressão: `13 + -12`)  
■ Melhor usar parênteses: `13 + (-12)`
- Atribuição com operação
  - `+=`, `-=`, `*=`, `/=`, `%=`
  - `x = x + 1` equivale a `x += 1`

Fundamentos de Java

28

## Incremento e Decremento

- *Exemplo*

```
int a = 10;
int b = 5;
```

- *Incrementa ou decrementa antes de usar a variável*

```
int x = ++a; // a contém 11, x contém 11
int y = --b; // b contém 4, y contém 4
```

▪ *A atribuição foi feita DEPOIS!*

- *Incrementa ou decrementa depois de usar a variável*

```
int x = a++; // a contém 11, x contém 10
int y = b--; // b contém 4, y contém 5
```

▪ *A atribuição foi feita ANTES!*

Fundamentos de Java

29

## Operadores Relacionais

- $==$  igual

- $\neq$  diferente

- $<$  menor

- $\leq$  menor ou igual

- $>$  maior

- $\geq$  maior ou igual

- Sempre produzem um resultado booleano

- true ou false

- Comparam os valores de duas variáveis ou de uma variável e uma constante

- Comparam as referências de objetos (apenas == e !=)

30

# Operadores Lógicos

- **&&**      *E (and)*
- **||**          *Ou (or)*
- **!**          *Negação (not)*
  
- *Produzem sempre um valor booleano*
  - *true ou false*
  - *Argumentos precisam ser valores booleanos ou expressões com resultado booleano*
  - *Por exemplo: (3 > x) && !(y <= 10)*
- *Expressão será realizada até que o resultado possa ser determinado de forma não ambígua*
  - *"short-circuit"*
  - *Exemplo: (false && <qualquer coisa>)*
  - *A expressão <qualquer coisa> não será calculada*

31

# Operadores bit-a-bit

- **&** *and*
- **|** *or*
- **^** *xor (ou exclusivo)*
- **~** *not*
  
- *Para operações em baixo nível (bit por bit)*
  - *Operam com inteiros e resultados são números inteiros*
  - *Se argumentos forem booleanos, resultado será igual ao obtido com operadores booleanos, mas sem 'curto-circuito'*
  - *Suportam atribuição conjunta: &=, |= ou ^=*

32



# E/S de dados

Fundamentos de Java

## Imprimindo na saída padrão

```

String nome = "Maria";
int idade = 23;
double altura = 1.67;

System.out.println(nome + " tem " + idade + " anos " + "e mede " +
    altura + "cm de altura.");

System.out.print("\nNome: ");
System.out.println(nome);
System.out.print("Altura: ");
System.out.println(altura);

System.out.printf("\n%s tem %d anos e mede %f de altura.\n",
    nome, idade, altura);

```

Imprime na tela e acrescenta quebra de linha ao final

Saída formatada na tela

Fundamentos de Java

34

## Saída na tela

```
Maria tem 23 anos e mede 1.67cm de altura.
```

```
Nome: Maria
```

```
Altura: 1.67
```

```
Maria tem 23 anos e mede 1.670000 de altura.
```

Fundamentos de Java

35

## Entrada de dados – Scanner()

```
import java.util.*;

public class LeScanner{
    public static void main(String[] args) {
        Scanner ent = new Scanner(System.in);

        System.out.print("Digite um nome: ");
        String nome = ent.next(); Lê uma String
        System.out.print("Digite a idade: ");
        int idade = ent.nextInt(); Lê um inteiro
        System.out.println(nome + " tem " + idade + " anos.");
    }
}
```

Fundamentos de Java

36

## Saída na tela

---

Digite um nome: Marcio

Digite a idade: 28

Marcio tem 28 anos.

Fundamentos de Java

37

## Entrada de dados

---

```

1   int n = ent.nextInt();
2   String nome = ent.next();
3   char sexo = ent.next().charAt(0); // Lê char na 1a posição da String
4   double d = ent.nextDouble();
5   System.out.println(n + " " + nome + " " + sexo + " " + d);

6   ent.nextLine();                // Limpa a quebra de linha pendente
7   String linha = ent.nextLine(); // Lê até o final da linha
8   String[] vet = linha.split(" "); // Separa os tokens da String
9   n = Integer.parseInt(vet[0]);    // Converte de string para int
10  nome = vet[1];
11  sexo = vet[2].charAt(0);
12  d = Double.parseDouble(vet[3]); // Converte para double
13  System.out.println(n + " " + nome + " " + sexo + " " + d);

```

38

## Material Adicional

- Introdução à Programação Java, Parte 1 e 2 (IBM)
  - <https://www.ibm.com/developerworks/br/java/tutorials/j-introtojava1/index.html>
- The Java Tutorials (Oracle oficial)
  - <https://docs.oracle.com/javase/tutorial/>
- Learn Java Programming (Tutorials Point)
  - <http://www.tutorialspoint.com/java/>

Fundamentos de Java

