Universidade Federal do Agreste de Pernambuco

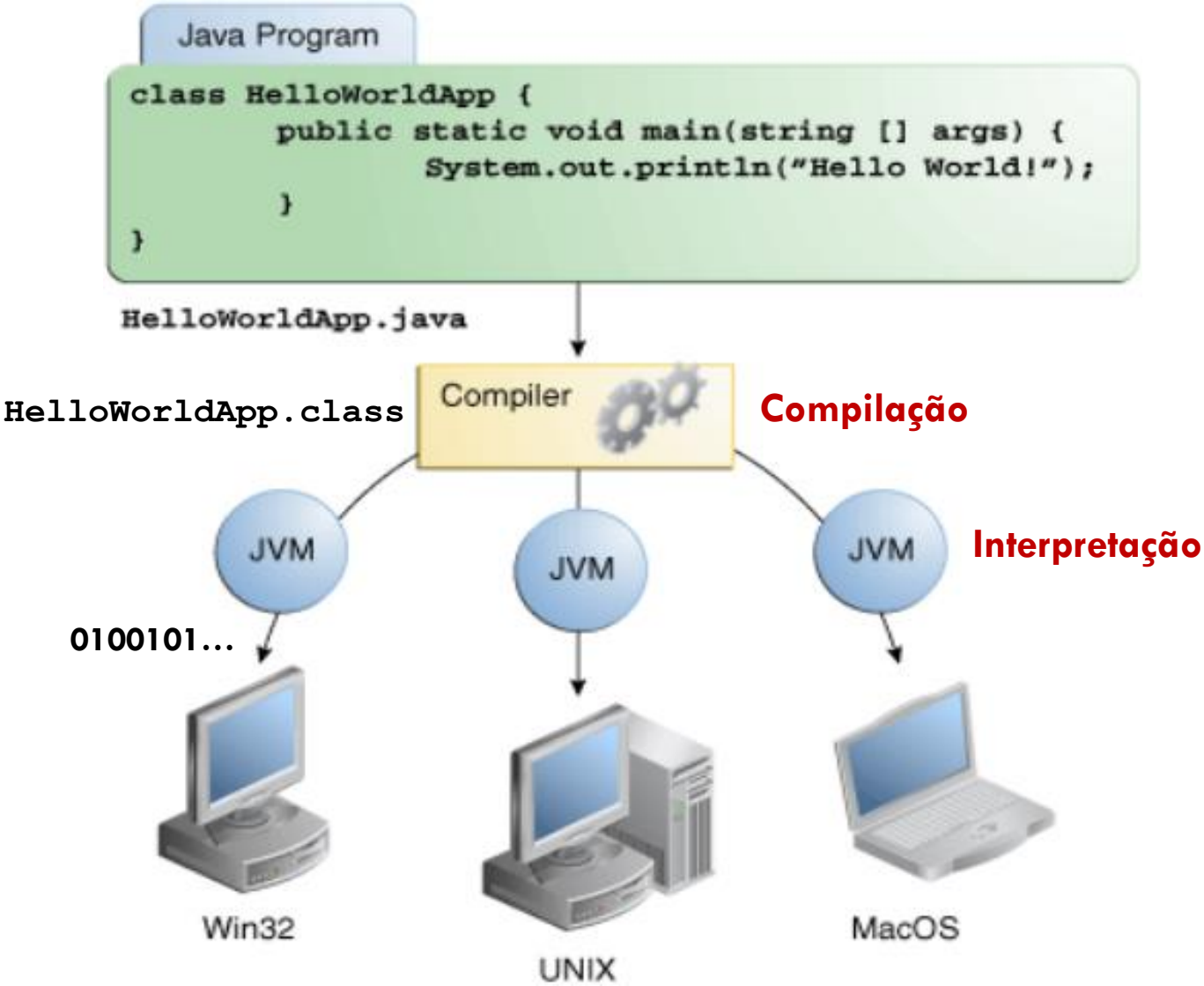
INTRODUÇÃO A PROGRAMAÇÃO EM JAVA

Thaís Alves Burity Rocha

Agenda

- Compilação e interpretação
- Plataformas
- Ferramentas
- Sintaxe
  - Tipos primitivos de dados
  - Operadores
  - Estruturas de controle
  - Estruturas de repetição

Compilação e interpretação



Principais plataformas Java

JSE (Java Standard Edition)	• É a base • Inclui o ambiente de execução e as bibliotecas principais • Aplicações desktop
Jakarta EE (Antes JEE - Java Enterprise Edition)	• Servidores • Aplicações distribuídas • Aplicações web • Microserviços
JME (Java Micro Edition)	• Dispositivos móveis (celular, tablet) • Sistemas embarcados • Internet das Coisas (IoT) • Sensores, Tv, impressoras

Java Card	• Dispositivos embarcados limitados
	  • Cartões SIM (chip do celular)
	  • Chip de cartão de crédito

Ferramentas

- **JRE** (Java Runtime Environment): Conjunto de ferramentas para **executar** um programa Java
  - **JVM (Java Virtual Machine)**
  - Biblioteca de classes fundamentais
- **JDK** (Java Development Toolkit): Conjunto de ferramentas para **desenvolver** programas Java
  - **JRE**
  - **Javac**: Compilador
  - **Jar**: Compacta o programa para distribuição
  - **Javadoc**: Ferramenta para geração de documentação

Variáveis e tipos

- Variável é um **container** em memória
- Em Java, toda variável tem **tipo, tamanho** e **identificador**
- Exemplos
  - int: 32 bits, long: 64 bits

```
int a = 1234;
long b = a;
int c = (int) b;
```

Conversão implícita

Conversão explícita

- Tipo primitivo = Tipo básico da linguagem

Resumo dos tipos primitivos em Java

Tipo	Valor	Tamanho (bits)	Valor mínimo	Valor máximo	Valor default
boolean	true ou false	depende da JVM	n/a	n/a	false
char	inteiro positivo que representa um caractere Unicode	16	0	$2^{16}-1$	\u0000
byte	inteiro	8	-2^{7}	$2^{7}-1$	0
short	inteiro	16	-2^{15}	$2^{15}-1$	0
int	inteiro	32	-2^{31}	$2^{31}-1$	0
long	inteiro	64	-2^{63}	$2^{63}-1$	0
float	ponto flutuante	32	n/a	n/a	0.0
double	ponto flutuante	64	n/a	n/a	0.0

Tipos de dados em programação

- Linguagens como Python, Ruby e Groovy também suportam **tipagem dinâmica**
- O tipo da variável é inferido pelo valor atribuído e pode ser alterado em tempo de execução

```
//v pode armazenar qualquer valor
def v = 3
v = false
v = 10.5
```

- Java suporta **tipagem estática** e desde a versão 10 suporta inferência de tipo de forma limitada

```
var x = 10;
```

String

- Tipo não primitivo usado para representar texto

```
String exemplo = "palavra";

String blocoTexto1 = "linha1\nlinha2\nlinha3";

//A partir de Java 15
String blocoTexto2 = """
linha1
linha2
linha3""";
```

Identificadores

- Não podem conter caractere especial, exceto $ e _
- O identificador não pode ser apenas _
- Não podem começar com números
- Não podem conter operadores matemáticos
- São case-sensitive:

```
int contador;
int CONTADOR;
int cOnTaDoR;
```

- Convenção
  - Inicial minúscula
  - Junção de palavras captalizadas: aulaPoo
  - Nomes curtos e significativos

Operadores aritméticos

- Aplicam-se à tipos numéricos

Operador	Operação
-	Menos unário (inversão de sinal)
+	Adição
-	Subtração
*	Multiplicação
/	Divisão inteira
%	Resto da divisão inteira (módulo)

- Exemplos:

```
2 + 45 → 47
- (5 + 2) → - 7
3 * 4 → 12
1/2 → 0
16 % 5 → 1
int y = 1 * 4 * 4 + 2 * 5 + 9;
```

Operadores aritméticos de atribuição

- Aplicam-se à tipos numéricos

Operador	Equivalente à
a += b	a = a + b
a -= b	a = a - b
a *= b	a = a * b
a /= b	a = a / b
a %= b	a = a % b

- Exemplos:

```
int a = 5, b=3;
a+=b; //a➔8  e b➔3
a*=b; //a➔15 e b➔3
```

Incremento e decremento

```
int a=5, b=5;

System.out.println(++a);

System.out.println(b--);

System.out.println(b);
```

Incremento prefixo

Decremento sufixo

Operador de concatenação

```
String nome = “Marcos”;
String sobrenome = “Silva”;
String nomeCompleto = nome + “ ” +  sobrenome;
//Marcos Silva
```

```
int a = 10;
int b = 2;

System.out.println(a+b);                  //12
System.out.println("numero "+a+b);   //numero 102
System.out.println("numero "+(a+b)); //numero 12
System.out.println(a+b+" numero");   //12 numero
```

Operadores de comparação

- Também chamados de operadores relacionais
- Usados apenas com tipos numéricos

Operador	Aplicação	Resultado
>	a > b	**true** se **a** é maior do **b** **false**, caso contrário
>=	a >= b	**true** se **a** é maior que ou igual a **b** **false**, caso contrário
<	a < b	**true** se **a** é menor do **b** **false**, caso contrário
<=	a <= b	**true** se **a** é menor que ou igual a **b** **false**, caso contrário

Operadores de igualdade

- Subtipo de operadores relacionais

Operador	Denominação	Aplicação	Resultado
==	igual a	a == b	**true** se **a** é igual a **b** **false**, caso contrário
!=	diferente de	a != b	**true** se **a** é diferente de **b** **false**, caso contrário

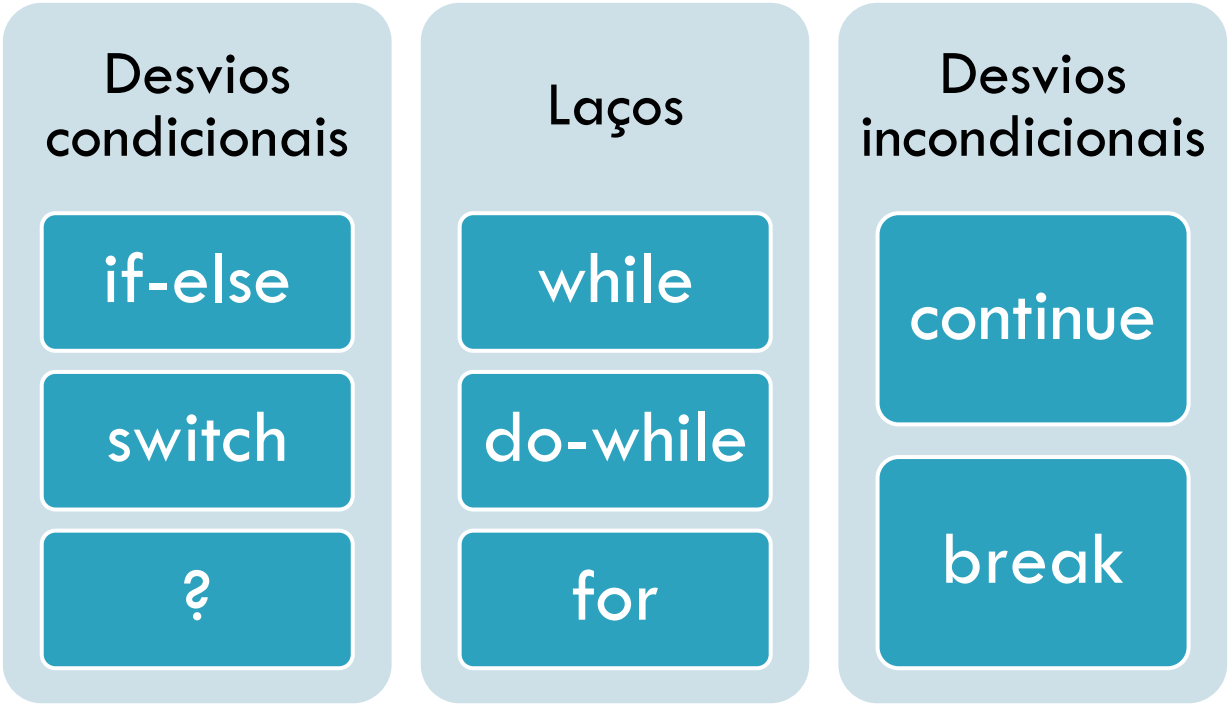
Operadores lógicos

- Aplicam-se apenas à **valores booleanos**

Operador	Símbolo	Descrição
Negação (NOT)	!	Retorna o contrário
Conjunção condicional (AND)	&&	Nem sempre verifica os dois operandos
Disjunção condicional (OR)	||	
Conjunção lógica (AND)	&	Sempre verifica os dois operandos
Disjunção lógica (OR)	|	
Disjunção exclusiva (XOR)	^	

Estruturas de controle

- Os programas são executados de forma sequencial
- Estruturas de controle mudam o fluxo de execução
- Em Java



if-else

```
if ( media < 5 ){
   resultado = "Reprovado";
} else if ( media >= 7 ){
   resultado = "Aprovado";
} else {
   resultado = "Recuperação";
}
```

- O **else** é opcional e não pode ser usado sozinho
- O corpo do **if-else** é limitado por '{' e '}'
  - É possível omitir, desde que só contenha **um** comando
- A condição sempre é limitada por '(' e ')'

switch

Vale para **byte**, **char**, **short**, **int** ou **String**

```
String r;
switch (dia) {
   case "segunda":
   case "terça":
   case "quarta":
   case "quinta":
   case "sexta":
      r = "Dia de semana";
      break;
   case "sábado":
   case "domingo":
      r = "Fim de semana";
      break;
   default:
      r = "Dia inválido!";
}
```

Expressão constante com valor único

Opcional

Executa somente quando todos os cases falham

Operador ternário

```
int x = 5, y;
```

```
if(x>3){
   y = x*2;
} else {
   y = x*4;
}
```

```
y = (x>3) ? x*2 : x*4;
```

Valor se false

Parênteses é opcional

Valor se true

- Pouco usado por comprometer a legibilidade

while & do-while

Teste é feito no **início**

```
int contador = 0;
while (contador < 10) {
   System.out.println(contador);
   contador = contador + 1;
}
```

Se não usar chaves, só executará um comando

```
int contador = 0;
do{
   System.out.println(contador);
   contador = contador + 1;
}while (contador < 10);
```

Teste é feito no **final**, portanto o loop executa **ao menos 1 vez**

for

```
for(int contador = 0 ; contador < 10 ; contador++){
    System.out.println(contador);
}
```

É executado depois do corpo

```
for(;;){
    System.out.println("executou for");
}
```

Loop infinito

```
for(int i=0, j=0; (i<10)&&(j<10) ; i++, j++){
    System.out.println("i vale "+i+", j vale "+j);
}
```

break

- Saída imediata da estrutura de controle e/ou repetição
- Pode ser usado em **switch, while, for** e **do-while**

```
for(int i=1; i<10; i++){
    if(i%2 == 0) break;
    System.out.println("i vale "+i);
}
```

Saída:

```
i vale 1
```

continue

- Interrompe a execução da iteração corrente e avalia a condição de repetição
- Usado apenas em laços: **while, for** e **do-while**

```
for(int i=1; i<10; i++){
    if(i%2 == 0) continue;
    System.out.println("i vale "+i);
}
```

Saída:

```
i vale 1
i vale 3
i vale 5
i vale 7
i vale 9
```