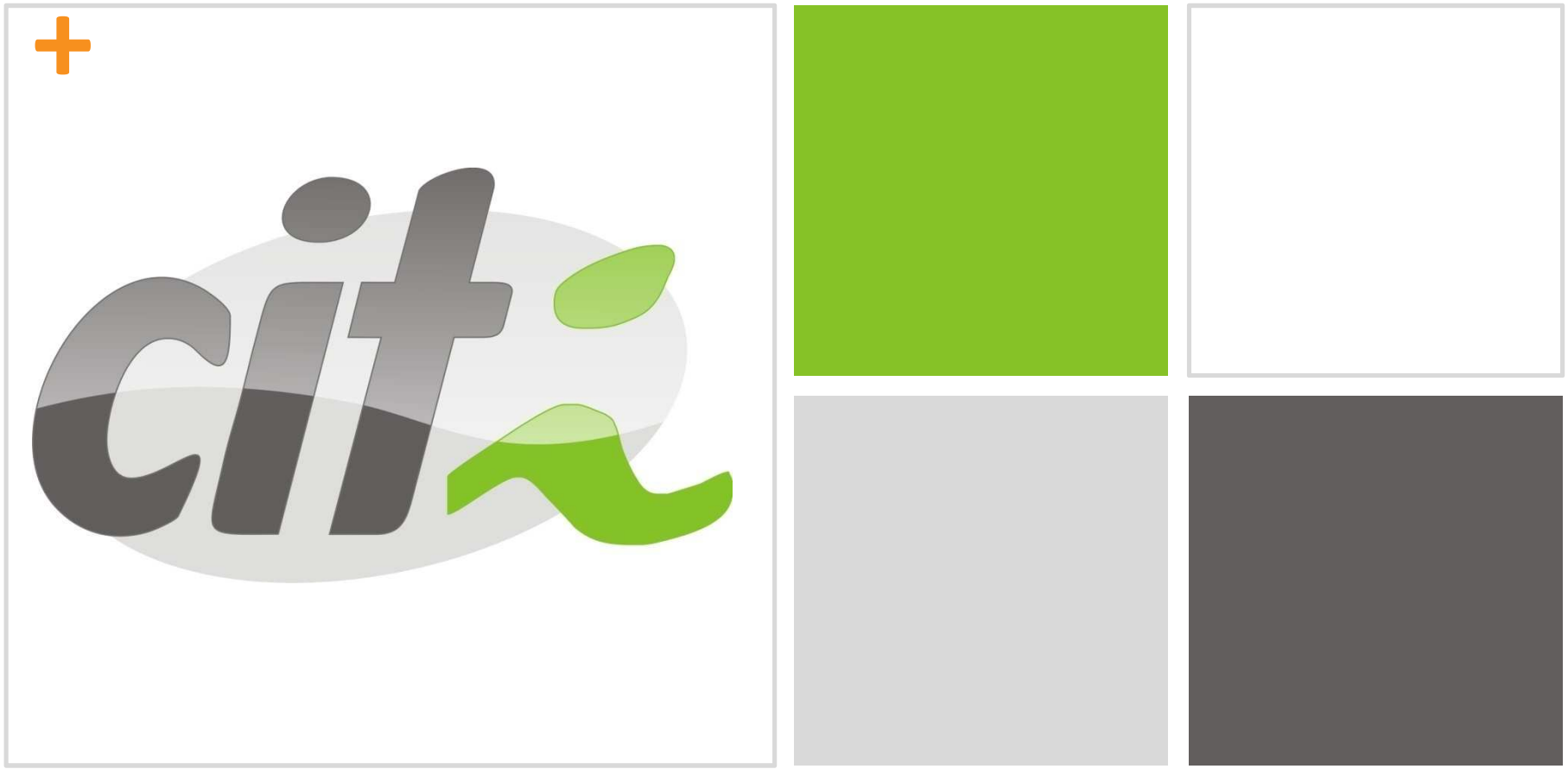


Java Básico

Igor Ebrahim (ies@cin.ufpe.br)

Módulo 2

Sintaxe

Sintaxe e Semântica

- Sintaxe: *parte da estrutura gramatical de uma língua que contém as regras relativas à combinação das palavras em unidades maiores (como as orações), e as relações existentes entre as palavras dentro dessas unidades;*
- Semântica: *estudo da linguagem humana do ponto de vista do significado das palavras e dos enunciados;*

Identificadores

- Podem ser:
  - Variáveis
  - Métodos
  - Atributos
  - Rótulos
  - …
- Regras:
  - Devem iniciar por uma letra, um _ (*underscore*) ou $ (cifrão)
  - O restante dos caracteres podem ser letras, algarismos, underscore ou cifrão
  - *Case sensitive*
    - maiúsculas são diferenciadas de minúsculas

Identificadores

- Boas práticas:
  - Evitem nomes sem significado (ou sem sentido)
  - Evitem siglas cujo significado não seja claro
  - Sigam o padrão para nomenclatura

Palavras reservadas

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

*	Não usado
**	Adic. em 1.2
***	Adic. em 1.4
****	Adic. em 5.0

true
false
null

} Não são palavras reservadas

Tipos Primitivos

boolean	true ou false
char	caracteres (16 bits Unicode)
byte	inteiro (8 bits)
short	inteiro (16 bits)
int	inteiro (32 bits)
long	inteiro (64 bits)
float	ponto flutuante (32 bits)
double	ponto flutuante (64 bits)

Declaração de Variáveis

- Padrão de codificação para nome de variáveis:
  - Começam com letras minúsculas
  - Se houver mais do que uma palavra, a seguinte começa com letra maiúscula

```
double saldo = 100.5;
float saldo2 = 100.5f;

int quantidadeMaxima = 10;

boolean achou = true;

char c = ‘a’;
```

String

- String representa uma cadeia ordenada de caracteres;
- Em java, String é uma classe e não um tipo primitivo;

```
String s = “Olá Mundo!”
```

- Note que String começa com uma letra maiúscula diferentemente dos tipos primitivos;

Operadores

- Tipos de Operadores:
  - Aritméticos
  - Concatenação
  - Relacionais
  - Lógicos
  - Atribuição
  - Unários
  - Condicional (ternário)

Operadores Aritméticos

+ - * / %

```
5 * 2                    Resultado: 10
```

Multiplicação

```
5 % 2                    Resultado: 1
```

Resto de divisão inteira

Operador de Concatenação

Aplicado a Strings

```
Strin nomeCompleto = nome + sobrenome
```

```
float saldo = 10.0f;
String mensagem = “Saldo da conta é: “ + saldo;
```

A concatenação também faz uma conversão implícita para String

Operadores Relacionais

>	Maior que	==	Igual
<	Menor que	!=	Diferente
>=	Maior que ou igual		
<=	Menor que ou igual		

```
int i = 10;
int j = 100;
boolean resultado = i < j;
```

```
boolean resultado = 21 < 7;
```

Operadores Lógicos (Short-circuit)

&& E lógico

|| OU lógico

```
boolean v = true;
boolean f = false;
boolean resultado = v && f;
```

```
boolean v = true;
boolean f = false;
boolean resultado = v || f;
```

Operadores de Atribuição

`=` `+=` `-=` `*=` `/=`

```
int x = 0;
int y = 1;
x += 1; // x = x + 1
y *= x + 3; // y = y * (x + 3)
```

Operadores Unários

++ -- - !

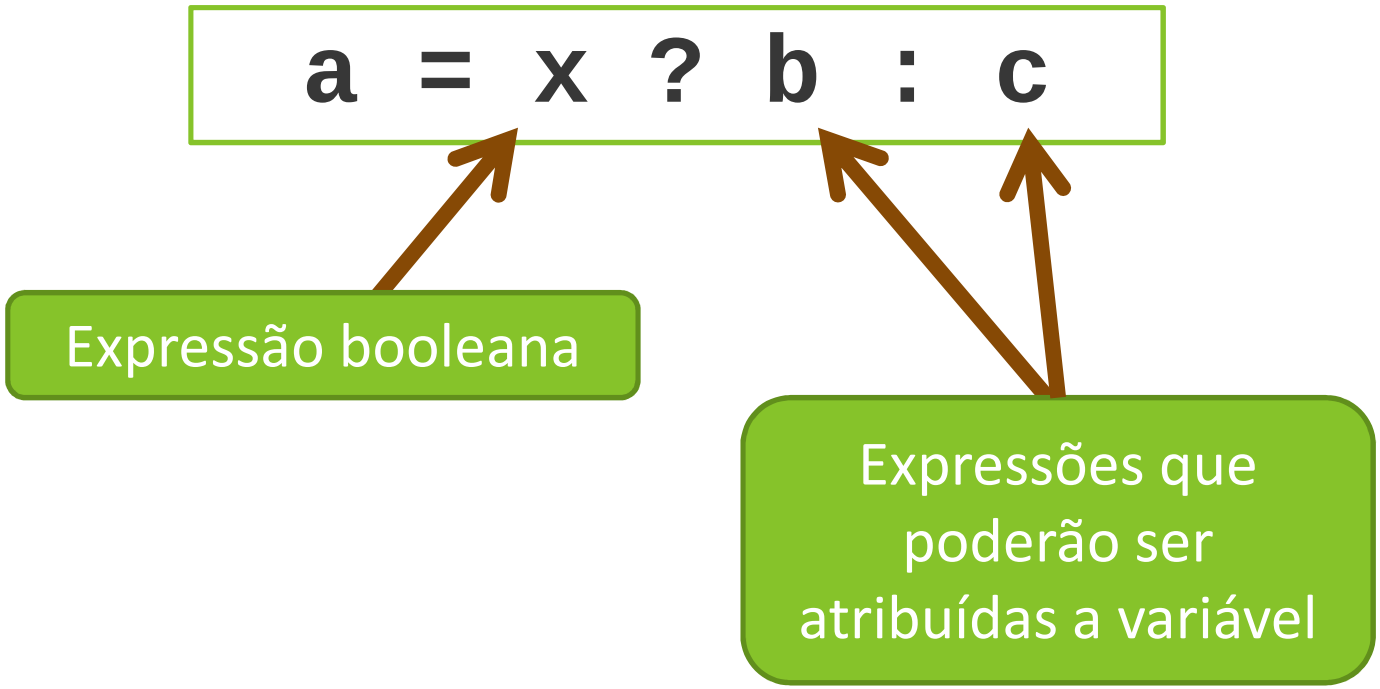
```
int x = 0;
int y = x++;
y = ++x;
y = x--;
y = --x;
```

```
int x = 0;
int y = -x;

boolean b1 = true;
boolean b2 = !b1;
```

Operador Condicional (ternário)

? :



Ordem de Avaliação dos Operadores

- Do maior para o menor:
  - Unário Posfixado — expr++ expr--
  - Unário Prefixado — ++expr --expr +expr -expr ~ !
  - Multiplicativos — * / %
  - Aditivos — + -
  - Shift — << >> >>>
  - Relacionais — < > <= >= instanceof
  - Igualdade — == !=
  - Bit-a-bit AND — &
  - Bit-a-bit exclusive OR — ^
  - Bit-a-bit inclusive OR — |
  - Lógico AND — &&
  - Lógico OR — ||
  - Ternário — ? :
  - Atribuição — = += -= *= /= %= &= ^= |= <<= >>= >>>=

Associatividade

- Operadores de mesma precedência, avalia-se primeiro o operado mais a esquerda:

  `a + b + c` equivale a `(a + b) + c`

  - Com exceção da atribuição!

    `a = b = c` equivale a `a = (b = c)`

- Precedência e associatividade podem ser redefinidos através de parênteses:

  `a * (b + c)` ou `a + (b + c)`

Conversão de Tipos

- Pode ser conveniente converter um tipo de dado para outro;
- Por exemplo, em uma situação particular podemos querer tratar um inteiro como um ponto flutuante para obter maior precisão;
- Essas conversões não mudam o tipo da variável ou o valor armazenado, elas apenas convertem o valor como parte da computação;
- Entretanto, conversões devem ser feitas com cuidado para evitar perda de informação;

Conversão de Tipos

- Conversão Ampliada são mais seguras porque elas tendem a ir de um tipo “menor” para um “maior”;
- Conversão Reduzida podem perder informação porque vão de um tipo “maior” para um “menor”;
- Em Java, conversão de tipo pode ocorrer de três maneiras:
  - Conversão atribuída;
  - Promoção;
  - Casting.

Conversão Atribuída

- Conversão atribuída ocorre quando um valor de um tipo é atribuído a uma variável de outro tipo;

```
int i = 2
float f = i;
```

- Apenas conversões ampliadas podem ser feitas dessa maneira;
- Note que o valor e o tipo de i não mudam;

Promoção

- Promoção ocorre automaticamente quando operadores em uma expressão convertem seus operandos:

```
float soma = 10.0f;
int cont = 2;
float result = soma / cont;
```

- “O valor de cont é convertido” para ponto flutuante para realizar o cálculo;

Casting

- Casting é a mais poderosa, porém perigosa, técnica de conversão;
- Tanto conversão ampliada e reduzida podem ser feitas;
- O tipo para o qual queremos converter é posto entre parênteses em frente ao valor a ser convertido:

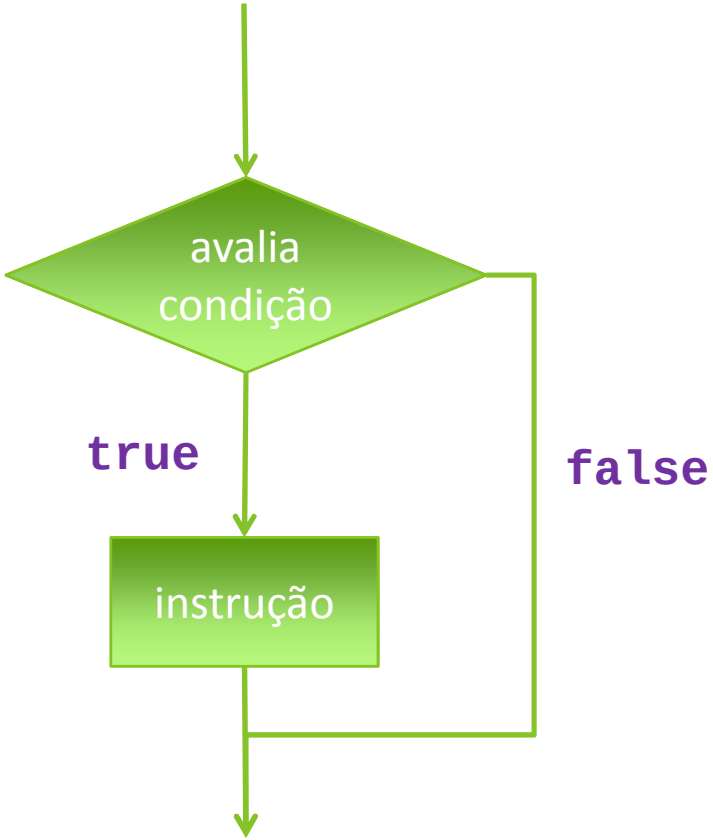
```
int total = 10;
int cont = 3;
float result = (float) total / cont;
```

- O valor de result será um ponto flutuante;

Estruturas de controle

- Servem para alterar o fluxo normal de um programa com a possibilidade de escolhas e repetições. São elas:
  - if
  - if-else
  - if-else-if
  - switch
  - while
  - do-while
  - for

if

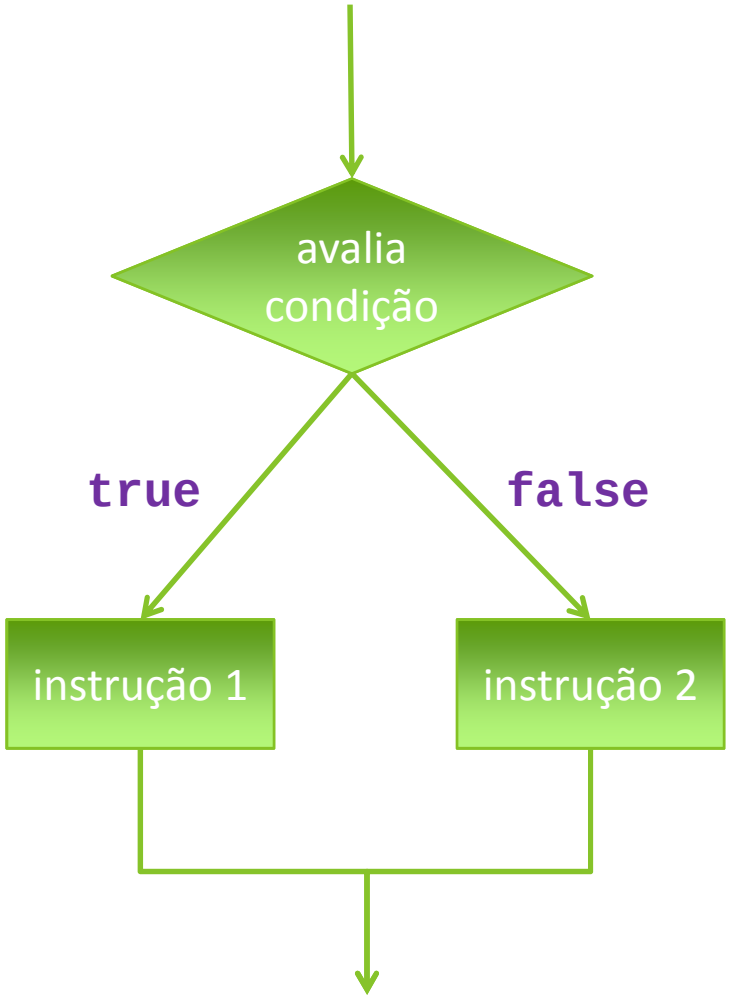


if

```
if (expressao_booleana) {
    // instrução
}
```

```
if (saldo > valor) {
     saldo = saldo – valor;
}
```

if-else



if-else

```
if (expressao_booleana) {
      // instrução 1
} else {
      // instrução 2
}
```

```
if (saldo > valor) {
       saldo = saldo – valor;
} else {
       System.out.println(“Operação negada”);
}
```

if-else-if

```
if (expressao_booleana) {
        // instrução 1
} else if (expressao_booleana) {
        // instrução 2
} else {
        // instrução 3
}
```

```
if (opcao >= 1) {
        facaIsto(0.7);
} else if (opcao < 1 && opcao >= 0) {
        facaIsto(0.4);
} else {
        System.out.println(“Operação negada”);
}
```

switch-case

```
switch (expressao_inteira) {
          case 1:
                    facaIsto(0.2);
                    break;
          case 2:
                    facaIsto(0.4);
                    break;
          default:

          System.out.println(“Erro...”)
    ;
}
```

```
switch (opcao) {
    case ‘s’:
    case ‘S’:
           sair();
           break;
    case ‘n’:
    case ‘N’:
           continuar();
           break;
    default:
           System.out.println(“Opção inválida”);
}
```

Comentários

```
// comentário de uma única linha


/* comentário de
  múltiplas linhas */


/** comentário especial
* “javadoc” usado para geração
* automática de documentação
*/
```

Escopo

- O escopo de uma variável determina onde ela é visível, ou seja, onde ela pode ser usada
- A regra é: dentro das chaves onde a declaração foi feita a variável é visível, fora das chaves, resulta em erro de compilação

Escopo

- Exemplo:

```
public static void main(String[] args) {
     int visivel = 0;
     if(true) {
          int visivelNoIf = 10;
          visivel = visivelNoIf;
     }
     visivelNoIf = visivelNoIf + 1;
}
```

Prática!

1. Faça um programa que leia entrada do usuário e diga se o número digitado é par, se ele é múltiplo de 3 e seus 10 primeiros múltiplos
2. Faça um programa que leia 3 inteiros do usuário e imprima seu produto
3. Faça um programa que leia 3 inteiros do usuário e imprima o maior dos 3

- Par ler entrada do usuário utilize a seguinte linha:

```
Scanner input = new Scanner(System.in);
int numero = input.nextInt();
```