

# **INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE**

## **PROGRAMAÇÃO ESTRUTURADA E ORIENTADA A OBJETOS**

**Docente: Éberton da Silva Marinho**

**e-mail: [ebertonsm@gmail.com](mailto:ebertonsm@gmail.com)**

**[eberton.marinho@gmail.com](mailto:eberton.marinho@gmail.com)**

25/02/2019

# SUMÁRIO

- Tipos de Dados Java
- Variáveis
- Operadores
- Processo Lógico de Programação
- Estruturas de desvio condicional



# JAVA

- Java é uma linguagem **FORTEMENTE TIPADA**.
  - Linguagem onde os dados são bem definidos.
  - A manipulação de dados se dá com o conhecimento do escopo de seus possíveis estados já na especificação dos programas dentro de um conjunto bem definido



# TIPOS DE DADOS

- Tipos de dados são modelos que representam conjuntos de valores que podemos manipular no programa
- Tipos Primitivos
  - São os tipos de dados predefinidos pela linguagem de programação Java
- Tipos definidos pelo usuário
  - São os tipos de dados que o usuário define através de classes



# TIPOS DE DADOS PRIMITIVOS

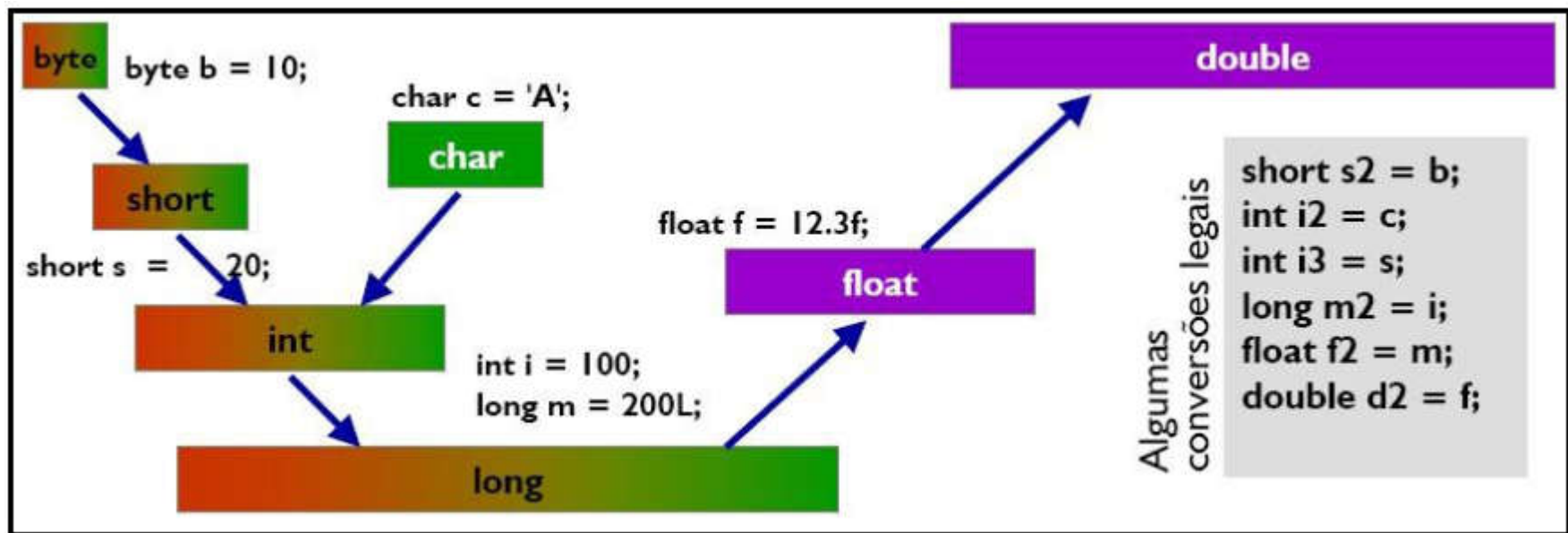
Tipo	Descrição	Mínimo	Máximo
boolean	Ocupa 1 bit. Assume os valores true ou false.		
char	Caractere em notação Unicode de 16 bits. Serve para armazenar dados alfanuméricos. Também pode ser usado como um dado inteiro.	0	65535
byte	Inteiro de 8 bits em notação de complemento de dois.	-128	127
short	Inteiro de 16 bits em notação de complemento de dois.	-32768	32767
int	Inteiro de 32 bits em notação de complemento de dois.	2147483648	2147483647
long	Inteiro de 64 bits em notação de complemento de dois.	$-2^{63}$	$2^{63}-1$
float	Representa números em notação de ponto flutuante normalizada em precisão simples de 32 bits em conformidade com a norma IEEE 754-1985. 4 bytes de tamanho e 23 dígitos binários de precisão.	1.40239846e-46	3.40282347e+38
double	Representa números em notação de ponto flutuante normalizada em precisão dupla de 64 bits em conformidade com a norma IEEE 754-1985. 8 bytes de tamanho e 52 dígitos binários de precisão.	4.94065645841246544e-324	1.7976931348623157e+308

# TIPOS DE DADOS PRIMITIVOS

- Apesar de Java ser fortemente tipada, alguns tipos primitivos diferentes podem ser manipulados. Para isso, conversões entre tipos tem que ser feitas
  - Um byte pode ser convertido em um short, int, long, float ou double
  - Um short pode ser convertido em um int, long, float ou double
  - Um char pode ser convertido em um int, long, float ou double
  - Um int pode ser convertido em um long, float ou double
  - Um long pode ser convertido em um float ou double
  - Um float pode ser convertido em um double



# TIPOS DE DADOS PRIMITIVOS



# TIPOS DE DADOS PRIMITIVOS

- Exemplo de código





## TIPOS DE DADOS CHAR

- Permite a representação de caracteres individuais no formato UNICODE.
- Alguns caracteres não possuem representação visual

Representação	Significado
\n	Pula linha
\r	Retorno de carro
\b	Retrocesso
\t	Tabulação
\f	Nova página
\'	Apóstrofo
\"	Aspas
\\	Barra invertida



# VARIÁVEIS

- Uma variável em programação é uma referência a uma posição da memória onde podemos guardar alguma informação;
- Para referenciar essa posição, as linguagens de programação utilizam apelidos com nomes mais sugestivos à aquilo que elas armazenam;
- Toda variável possui um nome, um tipo e um conteúdo
- Variáveis podem ser declaradas em todo o escopo do programa, código que estiver entre ‘{’ e ‘}’



# NOMENCLATURA

- **Identificadores válidos:** Definem as regras para que o compilador identifique o nome como válido.
  1. Devem iniciar com uma letra, cifrão (\$) ou sublinhado/underscore (\_);
  2. Após o primeiro caractere podem ter qualquer combinação de letras, caracteres e números;
  3. Não possuem limite de tamanho;
  4. Não podem ser palavras reservadas;
  5. Identificadores são case-sensitive isto é, “Nome” e “nome” são identificadores diferentes.



# NOMENCLATURA

- O nome das variáveis não pode ter outros símbolos gráficos, operadores ou espaços em branco;
- Exemplos válidos

Identificadores válidos	Identificadores inválidos
_codigo	5ident
\$turma	-idade
\$\$_5A	%valor

- Exemplos inválidos. Por quê?
  - 1x, Total Geral, numero-mínimo, void



# NOMENCLATURA

## ○ Palavras reservadas

<i>abstract</i>	<i>continue</i>	<i>finally</i>	<i>interface</i>	<i>public</i>	<i>throw</i>
<i>boolean</i>	<i>default</i>	<i>float</i>	<i>long</i>	<i>return</i>	<i>throws</i>
<i>break</i>	<i>do</i>	<i>for</i>	<i>native</i>	<i>short</i>	<i>transient</i>
<i>byte</i>	<i>double</i>	<i>if</i>	<i>new</i>	<i>static</i>	<i>true</i>
<i>case</i>	<i>else</i>	<i>implements</i>	<i>null</i>	<i>super</i>	<i>try</i>
<i>catch</i>	<i>extends</i>	<i>import</i>	<i>package</i>	<i>switch</i>	<i>void</i>
<i>char</i>	<i>false</i>	<i>instanceof</i>	<i>private</i>	<i>synchronized</i>	<i>while</i>
<i>class</i>	<i>final</i>	<i>int</i>	<i>protected</i>	<i>this</i>	
<i>const</i>	<i>future</i>	<i>generic</i>	<i>goto</i>	<i>inner</i>	<i>operator</i>
<i>outer</i>	<i>rest</i>	<i>var</i>	<i>volatile</i>		



# CONVENÇÃO DE NOMENCLATURA DA SUN

1. Classes e interfaces: A primeira letra deve ser maiúscula e, caso o nome seja formado por mais de uma palavra, as demais palavras devem ter sua primeira letra maiúscula também (CamelCase);
2. Métodos: A primeira letra deve ser minúscula e após devemos aplicar o camelCase;
3. Variáveis: Da mesma forma que métodos;
4. Constantes: Todas as letras do nome devem ser maiúsculas e caso seja formada por mais de uma palavra separada por underscore.



# CONVENÇÃO DE NOMENCLATURA DA SUN - EXEMPLOS

Classes	Métodos	Variáveis	Constantes
Carro	desligar	motor	COMBUSTIVEL
CursoJavaIniciante	iniciarModulo	quantidadeModulos	NOME_CURSO
Hotel	reservarSuiteMaster	nomeReservaSuite	TAXA_SERVICO



# EXEMPLOS DE DECLARAÇÃO DE VARIÁVEIS

```
int i;  
char letra1, letra2;  
float num1;  
double salario;
```





# ATRIBUIÇÃO A VARIÁVEIS

- Para que o conteúdo seja atribuído às variáveis, utiliza-se o operador '='
- Exemplos

```
int x;
```

```
x = 1;
```

```
int contador = 0;
```

```
double salario1 = 2000.0, salario2 =  
3500.0;
```

```
int a = b = c = d = 0;
```



# COMENTÁRIOS

- Trechos que explicam o código
- Há três formas de comentário no código Java
  - // Comentário de uma linha
  - // Tudo após as duas barras é considerado comentário
  - /\* comentário de  
    múltiplas linhas \*/
  - /\*\* comentário de documentação  
    \* que também pode ser de múltiplas linhas  
    \*/
- Geralmente o comentário de documentação é posicionado imediatamente antes do elemento a ser documentado



# OPERADORES

- Operadores aritméticos
- Operadores relacionais
- Operadores lógicos



# OPERADORES ARITMÉTICOS

Operador	Significado	Exemplo
+	Adição	$a + b$
-	Subtração	$a - b$
*	Multiplicação	$a * b$
/	Divisão	$a / b$
%	Resto da divisão inteira	$a \% b$
-	Sinal negativo	-a
+	Sinal positivo	+a
++	Incremento unitário	++a ou a++
--	Decremento unitário	--a ou a--



# OPERADORES RELACIONAIS

Operador	Significado	Exemplo
==	Igual	a == b
!=	Diferente	a != b
>	Maior que	a > b
>=	Maior ou igual	a >= b
<	Menor que	a < b
<=	Menor ou igual a	a <= b



# OPERADORES LÓGICOS

Operador	Significado	Exemplo
&&	E lógico (and)	a && b
	Ou Lógico (or)	a    b
!	Negação (not)	!a



# PRECEDÊNCIA DE AVALIAÇÃO DE OPERADORES

- Especifica a ordem de avaliação dos operadores em uma expressão

Nível	Operadores
1	. (seletor) [ ] ( )
2	++ -- ~ instanceof new clone – (unário)
3	* / %
4	+ -
5	<< >> >>>
6	< > <= >=
7	== !=

Nível	Operadores
8	&
9	^
10	
11	&&
12	
13	? :
14	=
15	,



# PROCESSO DE PROGRAMAÇÃO

- Para o programador resolver um problema, é necessário obedecer um passo a passo lógico onde ele consiga chegar a uma solução
  1. Ler e entender o problema;
  2. Definir as entradas e as saídas do programa;
  3. Dadas as entradas, resolver o problema em um ambiente fora da programação. Neste momento também são definidas as variáveis necessárias no processo de resolução;
  4. Escolher um caso de teste e solucione o problema sem o computador. Encontre as saídas com base nas entradas teste;





# PROCESSO DE PROGRAMAÇÃO

5. Traduzir a solução do problema em código. A tradução deve ser:
  1. Declarar todas as variáveis do programa
  2. Inicializar TODAS as variáveis
  3. Escrever o código que solucione o problema
  4. Escrever o código que imprimir a saída do programa
6. Executar o programa com o caso de teste como entrada e verificar se a saída é igual ao do item 4.



## PROBLEMA

- Crie um programa em Java que solucione uma equação do segundo grau na forma:

$$ax^2 + bx + c = 0$$



## RESOLUÇÃO DO PROBLEMA

- Item 2: as entradas do programa devem ser os valores de  $a$ ,  $b$  e  $c$  na equação do segundo grau.
- Item 3: a resolução de uma equação do segundo grau pode ser realizada pela fórmula de Bhaskara
  - $\text{delta} = b^2 - 4ac$
  - Se  $\text{delta} \geq 0$ 
    - $x1 = \frac{-b + \sqrt{\text{delta}}}{2a}$
    - $x2 = \frac{-b - \sqrt{\text{delta}}}{2a}$
  - Variáveis  $\text{delta}$ ,  $x1$  e  $x2$



# RESOLUÇÃO DO PROBLEMA

- Item 4: caso de teste
  - Entradas:  $a = 1$ ,  $b = 5$  e  $c = 4$
  - Saídas:  $x_1 = -1$  e  $x_2 = -4$
- Item 5:
  1. Declarar todas as variáveis do programa
  2. Inicializar TODAS as variáveis
  3. Escrever o código que solucione o problema
  4. Escrever o código que imprimir a saída do programa
- Item 6: Executar o programa com o caso de teste do item 4 e comparar as saídas.

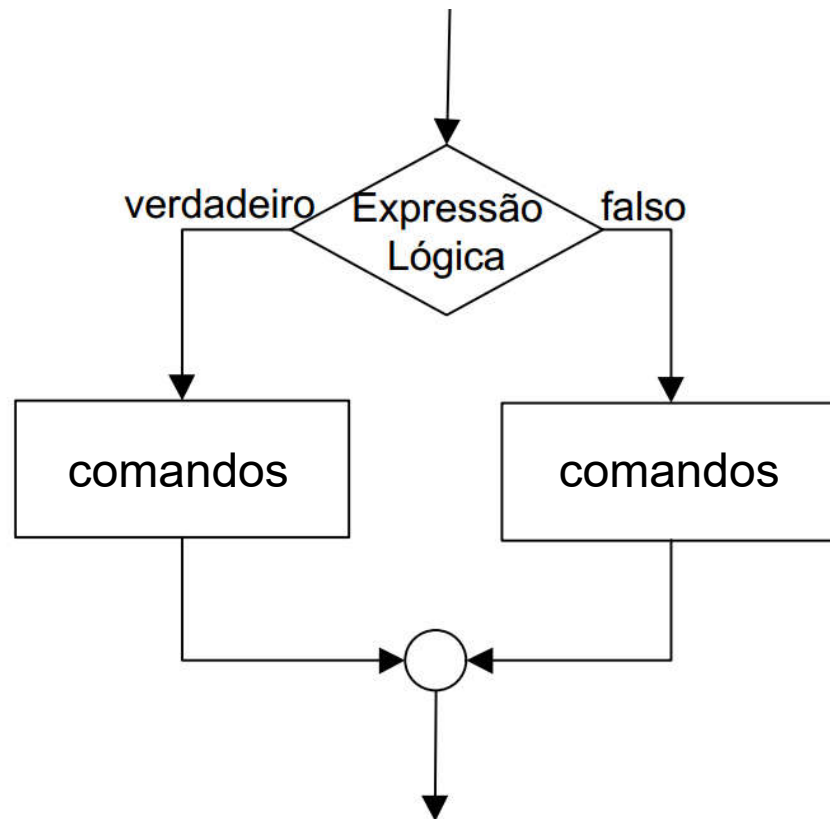


# ESTRUTURAS DE CONTROLE

- Um algoritmo segue um fluxo lógico e bem definido para a resolução de um problema
- As linguagens de programação geralmente possuem estruturas de desvio mediante uma certa circunstância
  - Estruturas de repetição
    - Simples: O programa sabe a quantidade de vezes que o código será repetido
    - Condicional: Quando não se conhece de antemão a quantidade de vezes que o código será executado
  - Estrutura de desvio condicional



# ESTRUTURAS DE DESVIO CONDICIONAL



```
if(expressão lógica) {  
    comandos;  
}else{  
    comandos;  
}
```



# ESTRUTURAS DE DESVIO CONDICIONAL

## ○ Formas de escrita em Java

```
if (expressão)  
    comando1;  
comando2;
```

Bloco de  
comandos

```
if (expressão) {  
    comando1;  
    comando2;  
}
```

```
if (expressão) {  
    comando1;  
}
```

```
else {  
    comando2;  
}
```

```
if (expressão1) {  
    comando1;  
}
```


```
else if (expressão2) {  
    comando2;  
}
```

```
else {  
    comando3;  
}
```



# ESTRUTURAS DE DESVIO CONDICIONAL

Estruturas ifs  
aninhadas



```
if (expressão1) {  
    comando1;  
    if (expressão2) {  
        comando2;  
    }  
}  
  
else if (expressão3) {  
    comando3;  
    if (expressão4) {  
        comando4;  
    }  
}
```





# ESTRUTURAS DE DESVIO CONDICIONAL

- Exercícios



# ESTRUTURAS DE DESVIO CONDICIONAL

- O if é uma estrutura de desvio condicional de propósito amplo, mas há casos onde podemos usar outras estruturas com propósitos semelhantes.

```
switch (expressao) {  
    case valor1:  
        comandos1;  
        break;  
    case valor2:  
        comandos2;  
        break;  
    case valor2:  
        comandos3;  
        break;  
    default:  
        comandos4;  
}
```

O comando break impede que os outros cases sejam executados após a primeira confirmação.

O comando default funciona com o else da estrutura if.



# ESTRUTURAS DE DESVIO CONDICIONAL

- Exercícios



# ESTRUTURAS DE DESVIO CONDICIONAL

- Há uma estrutura condicional especial, que é na verdade um operador que funciona da seguinte forma
  - `int var = (expressão ? valor_verdade: valor_falso);`
- Exemplo
  - Se o valor da variável a for maior que a de b, o valor 50 será atribuído a variável x, senão o valor 100;
  - `int x = (a > b ? 50 : 100);`



# ESTRUTURAS DE DESVIO CONDICIONAL

- Quando usar cada estrutura
  - if: Para propósitos gerais
  - switch: Quando temos uma faixa de valores bem definidas
  - Operador ( ? : ): Quando precisamos de uma resposta rápida mediante a avaliação de uma expressão



## DICAS DO DIA

- Defina variáveis com nomes sugestivos, e que indiquem o que a variável se destina armazenar. Evite nomes muito curtos ou muito abreviados que dificultem o entendimento do código.
- Quando usar estruturas de desvio de fluxo, verifique se as condições cobrem todo o escopo do que se deseja cobrir, e não há sobreposição de faixa de valores.



# DÚVIDAS

- e-mail:

ebertonsm@gmail.com

eberton.marinho@ifrn.edu.br

- Endereço eletrônico da disciplina:

- <http://docente.ifrn.edu.br/ebertonmarinho>