
ORACLE
Academy

Java Foundations

3-2

Dados Numéricos

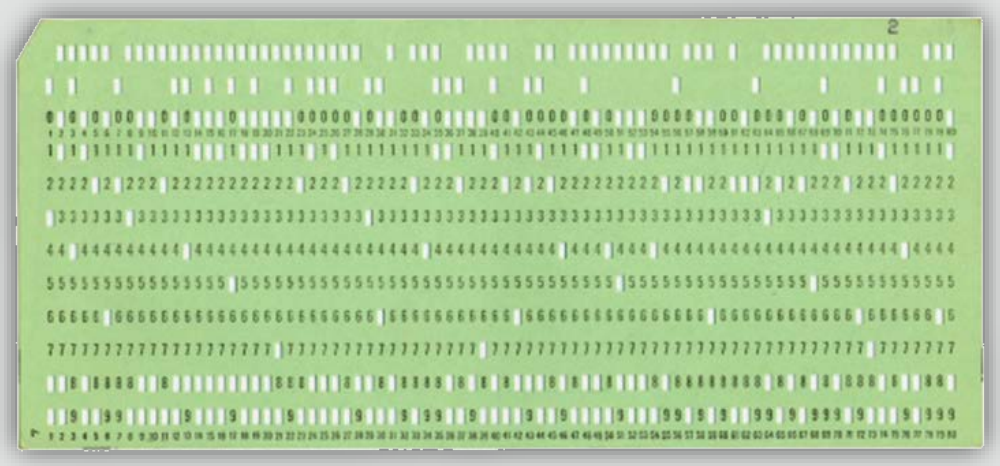
ORACLE Academy

Objetivos

- Esta lição abrange os seguintes objetivos:
  - Diferenciar tipos de dados inteiros (byte, short, int, long)
  - Diferenciar tipos de dados flutuantes (float, double)
  - Manipular e fazer cálculos matemáticos com dados numéricos
  - Usar parênteses e a ordem das operações

Um Pouco sobre os Dados

- Nos primórdios da computação, os dados eram armazenados em cartões perfurados



- Cada slot tinhas dois estados possíveis:
  - Perfurado
  - Não perfurado

Lendo Dados de um Cartão Perfurado

- Uma Pianola lê cartões perfurados
- Uma coluna representa uma tecla no piano
- O cartão perfurado rola pelo piano, acionando as teclas
- Cada slot tem dois estados possíveis, com dois resultados possíveis:

Rolagem de um piano dos anos 1800

Estado	Resultado
Perfurado	Tocar uma nota
Não perfurado	Não tocar uma nota

Um Pouco Sobre a Computação Moderna

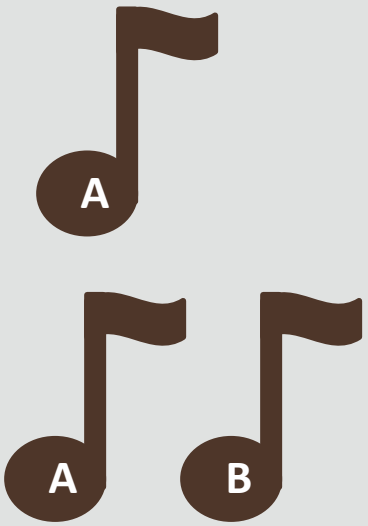
- O processamento moderno de dados continua a representar dois estados:
  - Isto é interpretado como um código binário: **10011101**
  - **1** ou **0** individual denomina-se um bit

	Pianola	**Computação Moderna**
Bit	Orifício perfurado/não perfurado	1/0
Os bits são instruções para…	Componentes mecânicos	O processador
Meio físico	Mecânico	Eletromagnetismo
Os bits armazenam dados sobre…	Teclas do piano	Números

Vamos analisar mais de perto

Bits de Dados

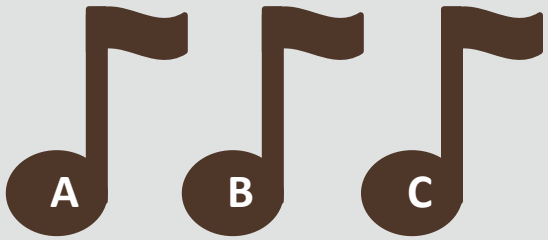
- A tecla de uma Pianola é representada por um bit
  - 0: Não tocar
  - 1: Tocar
- Duas teclas requerem dois bits
  - Existem quatro combinações possíveis de teclas
  - Podemos calcular isso como 2^2



	Tecla A	Tecla B
Silêncio	0	0
Somente B	0	1
Somente A	1	0
A e B	1	1

Bits Maiores de Dados

- Três teclas requerem três bits
  - Existem oito combinações possíveis de teclas
  - Podemos calcular isso como 2^3
- Oito teclas requerem oito bits
  - Existem 256 combinações possíveis
  - Podemos calcular isso como 2^8



Tecla A	Tecla B	Tecla C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Bits e Bytes

- Oito bits denominam-se um byte
- Um byte Java pode armazenar 256 valores possíveis. Os valores possíveis variam de -128 a 127
  - 128 valores abaixo de 0
  - 127 valores acima de 0
  - 1 valor igual a 0

```
byte x = 127;
```

```
byte z = 128;    //Muito alto
```

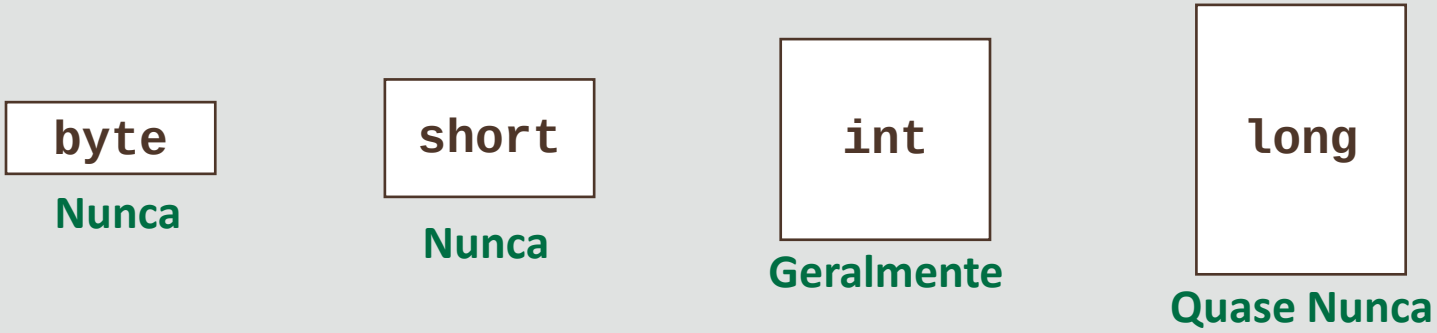
Alguns Novos Tipos Primitivos Integrais

Tipo	Comprimento	Número de Valores Possíveis	Valor Mínimo	Valor Máximo
Byte	8 bits	2^8 ou… 256	-2^7 ou… –128	2^7–1 ou… 127
short	16 bits	2^{16} ou… 65,535	-2^{15} ou… –32,768	2^{15}–1 ou… 32,767
int	32 bits	2^{32} ou… 4.294.967.296	-2^{31} ou… –2,147,483,648	2^{31}–1 ou… 2,147,483,647
long	64 bits	2^{64} ou… 18.446.744.073.709. 551.616	-2^{63} ou… –9.223.372.036, 854.775.808L	2^{63}–1 ou… 9.223.372.036, 854.775.807L

Observe o L

ORACLE Academy

Quando Usarei Cada Tipo de Dados?



- Os tipos byte e short são usados para reduzir o consumo de memória nos dispositivos mais antigos ou menores
- Mas os desktops modernos contêm uma memória muito grande
- Desses quatro tipos, usaremos principalmente o tipo int neste curso

Encontre o valor de x

```
int x = 20;
x = 25;
x = 5 + 3;

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

- x é sempre igual a 20 …
  - Até você atribuir outro valor a x
- É possível atribuir a x um valor calculado

Valores de x: ~~20~~ ~~25~~ 8

Encontre o valor de x

```
int x = 20;
x = 25;
x = 5 + 3;
x = x + 1;
x += 1;
x++;
System.out.println(x);
```

- É possível atribuir a x um novo valor com base em seu valor atual:
  - O Java fornece o operador += abreviado para fazer isso
  - Somar 1 a uma variável é tão comum que o Java fornece o operador ++ abreviado

Valores de x: ~~20~~ ~~25~~ ~~8~~ ~~9~~ ~~49~~ 11

Encontre o valor de x novamente

- É possível atribuir a x o valor de outra variável:
  - Mudar y não muda x
  - y e x são variáveis separadas

```
int y = 20;
int x = y;
y++;

System.out.println(x);
System.out.println(y);
```

- Saída:

```
x  20
y  21
```

Operadores Matemáticos Padrão

Finalidade	Operador	Exemplo	Comentários

Adição	+	`int sum = 0;`
		`int num1 = 10;`
		`int num2 = 2;`
		`sum = num1 + num2;`
			Se num1 é 10 e num2 é 2, sum é 12

Subtração	–	`int diff = 0;`
		`int num1 = 10;`
		`int num2 = 2`
		`diff = num1 – num2;`
			Se num1 é 10 e num2 é 2, diff é 8

Operadores Matemáticos Padrão

Finalidade	Operador	Exemplo	Comentários

Multiplicação	*	`int prod = 0;`
		`int num1 = 10;`
		`int num2 = 2;`
		`prod = num1 * num2;`
			Se num1 é 10 e num2 é 2, prod é 20

Divisão	/	`int quot = 0;`
		`int num1 = 31;`
		`int num2 = 2;`
		`quot = num1 / num2;`
			Se num1 é 31 e num2 é 6, quot é 5
			
			O resto é descartado
			
			Note:
			A divisão por 0 retorna um erro

Por quê?

Como os tipos de dados `int` são apenas números inteiros, a parte decimal restante será descartada. Você verá como alterar esse comportamento mais adiante nesta lição.

Como usar operadores abreviados para fazer atribuições

- Um operador abreviado é uma maneira mais curta de expressar algo que já está disponível na linguagem de programação Java

Finalidade	Operador	Exemplos de operadores abreviados	Construção equivalente	Resultado

Somar a e atribuir	`+=`	`int a = 6;`
		`int b = 2;`
		`a += b;`
			`int a = 6;`
			`int b = 2;`
			`a = a + b;`
				`a = 8`

Subtrair de e atribuir	`-=`	`int a = 6;`
		`int b = 2;`
		`a -= b;`
			`int a = 6;`
			`int b = 2;`
			`a = a - b;`
				`a = 4`

Como usar operadores abreviados para fazer atribuições

Finalidade	Operador	Exemplos de operadores abreviados	Construção equivalente	Resultado

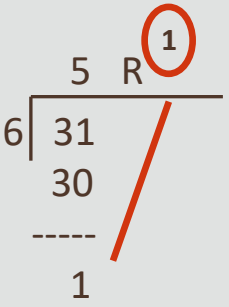
Multiplicar por e atribuir	*=	`int a = 6;`
		`int b = 2;`
		`a *= b;`
			`int a = 6;`
			`int b = 2;`
			`a  = a * b;`
				a = 12

Dividir por e atribuir	/=	`int a = 6;`
		`int b = 2;`
		`a  /= b;`
			`int a = 6;`
			`int b = 2;`
			`a  = a / b;`
				a = 3

Obter o resto e atribuir	%=	`int a = 6;`
		`int b = 2;`
		`a %= b;`
			`int a = 6;`
			`int b = 2;`
			`a  = a % b;`
				a = 0

Operador de Módulo

Finalidade	Operador	Exemplo	Comentários

Resto	% **(modulo)**	`num1 = 31;`
		`num2 = 6;`
		
		`mod = num1 % num2;`
		
		mod = 1
			Módulo encontra o resto do primeiro número dividido pelo segundo
			 
			Módulo sempre fornece uma resposta com o mesmo sinal do primeiro operando

Operadores de Acréscimo e Decréscimo (++ e --)

- A maneira longa:
  - `age = age + 1;`

  ou
  - `count = count – 1;`

- A maneira curta:
  - `age++;`

  ou
  - `count--;`

Mais sobre Operadores de Acréscimo e Decréscimo

Operador	Finalidade	Exemplo

++	Pré-acréscimo (++variável)	`int id = 6;`
		`int newId = ++id;`
		id é 7, newId é 7

	Pós-acréscimo (variável++)	`int id = 6;`
		`int newId = id++;`
		id é 7, newId é 6

--	Pré-decréscimo (--variável)	(O mesmo princípio se aplica)
	Pós-decréscimo (variável--)	

Operadores de Acréscimo e Decréscimo (++ e —)

```
1  int count=15;
2  int a, b, c, d;
3  a = count++;
4  b = count;
5  c = ++count;
6  d = count;
7  System.out.println(a + ", " + b + ", " + c + ", " + d);
```

- Saída:

```
15, 16, 17, 17
```

Exercício 1, Parte 1

- Crie um novo projeto e adicione o arquivo `Chickens01.java` a ele
- Leia esta história e calcule/imprima o totalEggs coletados entre segunda-feira e quarta-feira:
  - As galinhas do Fazendeiro Brown sempre colocam eggsPerChicken ovos ao meio-dia, e ele recolhe no mesmo dia
  - Na segunda-feira, o Fazendeiro Brown tem chickenCount galinhas
  - Na terça-feira de manhã, o Fazendeiro Brown ganha 1 galinha
  - Na quarta-feira de manhã, um animal come metade das galinhas!
  - Quantos ovos o Fazendeiro Brown recolheu se ele começa com...
    - eggsPerChicken = 5, chickenCount = 3
    - eggsPerChicken = 4, chickenCount = 8

Exercício 1, Parte 2

- Seu programa deve produzir a seguinte saída:

```
45 Primeiro cenário
84 Segundo cenário
```

Indução ao Erro na Divisão de Inteiros

- O animal comeu metade das galinhas
- Quando dividimos nove galinhas por dois, o Java considera 9/2 como 4
  - Mas 9/2 = 4,5
  - O Java não deveria arredondar para 5?
  - O que está acontecendo aqui?

Divisão em Java

- Os números inteiros em Java não são arredondados
- Os números inteiros em Java são truncados, o que significa que quaisquer números depois do ponto decimal serão removidos

```
int x = 9/2;
System.out.println(x);  //imprime 4
```

- Precisamos de outros tipos de dados se tivermos cenários que exijam uma precisão do ponto flutuante!

Tipos Primitivos de Ponto Flutuante

Tipo	Comprimento Flutuante	Quando usarei isso?
`float`	32 bits	Nunca
`double`	64 bits	Frequentemente

Dobra a precisão de um flutuante

- Exemplo:
  - `public float pi  = 3.141592F;`
  - `public double pi = 3.141592;`

Observe o F

Indução ao Erro em double

- O problema original:

```
int x = 9/2;
System.out.println(x);  //imprime 4
```

- double x deveria corrigir esse problema?

```
double x = 9/2;
System.out.println(x);  //imprime 4.0
```

  - Não?!?!
  - Por que não?

Indução ao Erro em double

```
double x = 9/2;
System.out.println(x);  //imprime 4.0\
```

- O Java soluciona a expressão, trunca o 0,5 e transforma a resposta em um double
- A expressão contém só valores inteiros. O Java não alocará a memória adicional necessária aos doubles até ele realmente precisar fazer isso.
  - Solução: inclua double na expressão

```
double x = 9/2.0;
System.out.println(x);  //imprime 4.5
```

Uma Observação Final

- Declare uma variável com a palavra-chave final para fazer com que o valor dela seja inalterado (imutável)

```
final double PI = 3.141592;
PI = 3.0;          //Não Permitido
```

- O Java impedirá se você tentar alterar o valor de uma variável final
- Convenções de nomenclatura da variável final:
  - Capitalize todas as letras
  - Use um caractere de sublinhado para separar as palavras
    - MINIMUM_AGE
    - SPEED_OF_LIGHT

Exercício 2, Parte 1

- Crie um novo projeto e adicione o arquivo `Chickens02.java` a ele
- Leia esta história e calcule/imprima os valores obrigatórios:
  - Na segunda-feira, o Fazendeiro Fred recolhe 100 ovos
  - Na terça-feira, ele recolhe 121 ovos
  - Na quarta-feira, ele recolhe 117 ovos
  - Qual é a dailyAverage (média diária) de ovos recolhidos?
  - Quantos ovos poderiam ser esperados em uma monthlyAverage (média mensal) de 30 dias?
  - Se um ovo pode ser vendido com um lucro de US$ 0,18, qual é o monthlyProfit (lucro mensal) total dos ovos?

Exercício 2, Parte 2

- Seu programa deve produzir a seguinte saída:

```
Daily Average:   112.66666666666667
Monthly Average: 3380.0
Profit:          $608.4
```

Parênteses em Expressões Matemáticas

- Esta expressão sem parênteses...

```
int x = 10 +20 +30 / 3;                    //x=40
```

- é como escrever essa expressão com parênteses:

```
int x = 10 +20 +(30 / 3);                  //x=40
```

- Se você quiser encontrar uma média, use parênteses como estes:

```
int x = (10 +20 +30) / 3;                  //x=20
```

Precedência do Operador

- Veja aqui um exemplo da necessidade de regras de precedência:

```
int x = 25 - 5 * 4 / 2 - 10 + 4;
```

- Esta resposta é 34 ou 9?

- Adicione parênteses para impor precedência

Regras de Precedência

- Operadores dentro de um par de parênteses
- Operadores de acréscimo e decréscimo (**++** ou **--**)
- Operadores de multiplicação e divisão, calculados da esquerda para a direita
- Operadores de adição e subtração, calculados da esquerda para a direita
- Se operadores da mesma precedência aparecerem sucessivamente, eles serão avaliados da esquerda para a direita

Usando Parênteses

- As expressões são avaliadas com as regras de precedência
- No entanto, você deve usar parênteses para fornecer a estrutura pretendida
- Exemplos:

```
int x = (((25 - 5) * 4) / (2 - 10)) + 4;
int x = ((20 * 4) / (2 - 10)) + 4;
int x = (80 / (2 - 10)) + 4;
int x = (80 / -8) + 4;
Int x = -10 + 4;
int x = -6;
```

Resumo

- Nesta lição, você deverá ter aprendido a:
  - Diferenciar tipos de dados inteiros (byte, short, int, long)
  - Diferenciar tipos de dados flutuantes (float, double)
  - Manipular e fazer cálculos matemáticos com dados numéricos
  - Usar parênteses e a ordem das operações


ORACLE
Academy