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

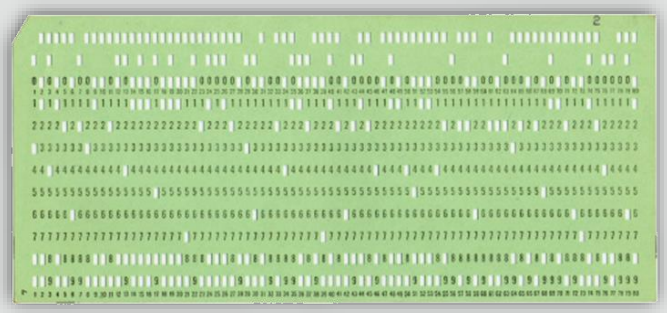
Tópicos

- **Um Pouco sobre os Dados**
- Trabalhando com Valores Inteiros
- Trabalhando com Pontos Flutuantes
- A Ordem das Operações

ORACLE Academy

Um Pouco sobre os Dados

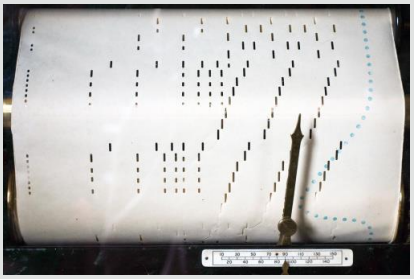
- Nos primórdios da computação, os dados eram armazenados em cartões perfurados



- 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

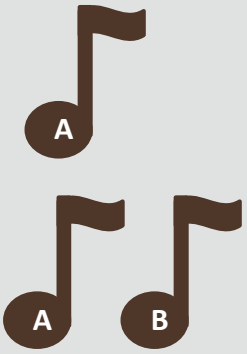
	Pianola	Computação Moderna
Bit	Orifício perfurado/não perfurado	1/0
Os bits são instruções para…	Componentes mecânicos	O processador
Meio físico	Mecânico	Eletromagnetismo
Os bits armazenam dados sobre…	Teclas do piano	Números

Vamos analisar mais de perto

ORACLE Academy

Bits de Dados

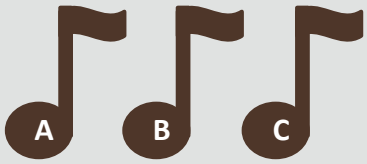
- A tecla de uma Pianola é representada por um bit
  - 0: Não tocar
  - 1: Tocar
- Duas teclas requerem dois bits
  - Existem quatro combinações possíveis de teclas
  - Podemos calcular isso como 22



	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 23
- Oito teclas requerem oito bits
  - Existem 256 combinações possíveis
  - Podemos calcular isso como 28



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

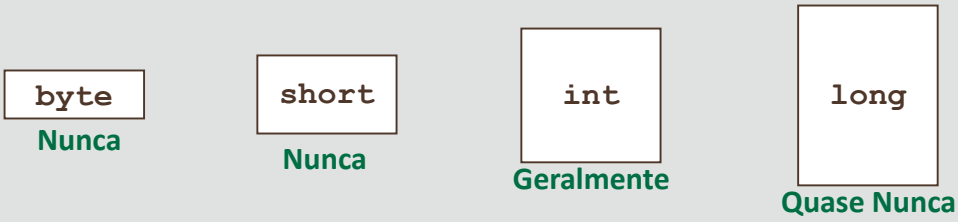
JFo 3-2
Dados Numéricos

11

Existem quatro tipos primitivos integrais (número inteiro) na linguagem de programação Java. Você já está usando o tipo de dados int. Então, vamos focar nos outros três. Os tipos integrais são usados para armazenar números que não têm partes decimais. Eles são mostrados aqui em ordem de tamanho.

- **byte:** se você precisasse armazenar idades de pessoas, uma variável do tipo byte funcionaria porque os tipos de byte podem aceitar valores nessa faixa.
- **short:** um tipo short (curto) conterá 16 bits de dados.
- **long:** quando você especificar um valor literal para um tipo long, coloque um L maiúsculo à direita do valor para informar explicitamente que esse tipo é longo. O compilador considera as literais de números inteiros como sendo do tipo int, a menos que você especifique o contrário usando um L para indicar o tipo longo.
- Você pode expressar qualquer um dos tipos integrais como binário (0s e 1s). Por exemplo, uma expressão binária do número 2 é mostrada como um valor permitido do tipo integral de byte. O valor binário é 0b10. Observe que esse valor começa com 0b (ou seja, zero seguido da letra *B* minúscula ou maiúscula). A letra indica ao compilador que um valor binário virá a seguir.

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

ORACLE Academy

Exemplos de valores literais permitidos:

- byte = 2, -114, 0b10 (número binário)
- short = 2, -32699
- int (tipo padrão para literais integrais) = 2, 147334778, 123_456_678
- long = 2, –2036854775808L, 1

Tópicos

- Um Pouco sobre os Dados
- **Trabalhando com Valores Inteiros**
- Trabalhando com Pontos Flutuantes
- A Ordem das Operações

ORACLE Academy

Encontre o valor de x

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

System.out.println();
```

- 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();
```

- 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

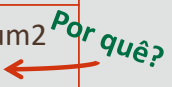
```
int y = 20;
int x = y;
y++;

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

- Saída:

```
x  20
y  21
```

Operadores Matemáticos Padrão

Finalidade	Operador	Exemplo	Comentários
Adição	+	sum = num1 + num2;	Se num1 é 10 e num2 é 2, sum é 12
Subtração	–	diff = num1 – num2;	Se num1 é 10 e num2 é 2, diff é 8
Multiplicação	*	prod = num1 * num2;	Se num1 é 10 e num2 é 2, prod é 20
Divisão	/	quot = num1 / num2;	Se num1 é 31 e num2 é 6, quot é 5 O resto é descartado A divisão por 0 retorna um erro



A tabela no slide assume que todos os operandos e variáveis resultantes sejam números inteiros (int). Combinar duplos e inteiros pode alterar os resultados. Por exemplo, no caso da divisão, se o quociente e o dividendo (ou se todos os três) forem duplos, o quociente mostrará o resto decimal:

```
double quot, num1;
num1 = 31;
int num2 = 5;
quot = num1 / num2;
```

Resposta: quot = 6,2

Combinando Operadores para Fazer Atribuições

Finalidade	Operador	Exemplos int a = 6, b = 2;	Resultado
Somar a e atribuir	+=	a += b	a = 8
Subtrair de e atribuir	-=	a -= b	a = 4
Multiplicar por e atribuir	*=	a *= b	a = 12
Dividir por e atribuir	/=	a /= b	a = 3
Obter o resto e atribuir	%=	a %= b	a = 0

A tabela a seguir mostra vários recursos muito úteis. Você pode combinar qualquer operador com o sinal de igualdade para abreviar seu código. Por exemplo:

```
a = a + b;
```

pode ser expresso como:

```
a += b;
```

Operador de Módulo

Finalidade	Operador	Exemplo	Comentários

Resto	%
	 módulo
		num1 = 31;
		 num2 = 6;
		
		 mod = num1 % num2;
		
		 mod é 1
			Resto encontra o resto do primeiro número dividido pelo segundo
			 5 R 1
			 6 | 31
			 30
			 ----- 	
			 1
			 Resto sempre fornece uma resposta com o mesmo sinal do primeiro operando

ORACLE Academy

JFo 3-2
Dados Numéricos

19

Os programas fazem vários cálculos matemáticos, desde cálculos simples a complexos. Os operadores aritméticos permitem que você especifique como os valores numéricos dentro das variáveis devem ser calculados ou combinados. Os operadores matemáticos padrão (também denominados *operadores binários*) usados na linguagem de programação Java são mostrados nas tabelas desta seção.

Observação: O sinal % também é conhecido como módulo.

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

- A maneira longa:
  - age = age + 1;
  - ou
  - count = count – 1;

- A maneira curta:
  - age++;
  - ou
  - count--;

Um requisito comum nos programas é somar ou subtrair 1 do valor de uma variável. Você pode fazer isso usando o operador + da seguinte maneira:

– age = age + 1;

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--)	

ORACLE Academy

Você usou operadores de acréscimo e decréscimo antes, colocando-os depois da variável que você deseja afetar. Mas você sabia que esses operadores podem vir antes (pré-acréscimo e pré-decréscimo) ou depois (pós-acréscimo e pós-decréscimo) de uma variável?

Quando você coloca o operador ++ ou -- antes de uma variável, o valor é alterado imediatamente. Quando você colocar o operador depois da variável, ele não será alterado depois que a expressão for calculada.

- No primeiro exemplo de código no slide, o id é inicializado para 6. Na linha seguinte, você vê `newId = ++id`. Como o operador precede o id, esse acréscimo é calculado imediatamente, e o valor atribuído a newId é 7.
- No segundo exemplo de código, o operador ++ segue o id. O id foi aumentado depois da atribuição. Portanto, newId é 6.
- Esses mesmos comportamentos aplicam-se a um operador de decréscimo (--) para seu posicionamento antes ou depois da 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
```

ORACLE Academy

O exemplo do slide mostra o uso básico dos operadores de acréscimo e decréscimo:

```
int count=15;
int a, b, c, d;
a = count++;
b = count;
c = ++count;
d = count;
System.out.println(a + ", " + b + ", " + c + ", " + d);
```

O resultado desse fragmento de código é:

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

Qual é o resultado do código a seguir?

```
int i = 16;
System.out.println(++i + " " + i++ + " " + i);
```

Exercício 1, Parte 1

- Importe e edite o projeto `Chickens01`
- 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
```

Tópicos

- Um Pouco sobre os Dados
- Trabalhando com Valores Inteiros
- **Trabalhando com Pontos Flutuantes**
- A Ordem das Operações

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?

ORACLE Academy

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:

Observe o F

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

ORACLE Academy

JFo 3-2
Dados Numéricos

Existem dois tipos de números de pontos flutuantes: float e double. Mais uma vez, focaremos no novo tipo de dados aqui: float Os tipos de pontos flutuantes são usados para armazenar números com valores à direita da vírgula decimal, como 12,24 ou 3,14159.

- float é usado para armazenar números de pontos flutuantes menores. Uma variável flutuante pode conter 32 bits.
- Os valores de pontos flutuantes são considerados como sendo do tipo double, a menos que você informe explicitamente um tipo float, e não um tipo double. Você faz isso colocando um F maiúsculo (float) à direita do valor.
- Aqui estão exemplos de valores literais permitidos:
  - float = 99F, -327456,99.01F, 4.2E6F (notação de engenharia para 4,2 * 106)
  - double = -1111, 2,1E12, 99970132745699.999

Observação: use o tipo double quando for necessário um intervalo ou uma precisão maior.

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 vantagem do Java é que ele trata o gerenciamento da memória para você. Abordaremos melhor esse assunto mais adiante.

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

- Importe e edite o projeto `Chickens02`
- 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
```

Tópicos

- Um Pouco sobre os Dados
- Trabalhando com Valores Inteiros
- Trabalhando com Pontos Flutuantes
- **A Ordem das Operações**

ORACLE Academy

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
```


No Exercício 2, se não usasse parênteses, você poderia ter criado um valor inteiro com o número total de ovos coletados nos três dias e, então, definido dailyAverage = threeDayTotal/3.0. Ou poderia ter dividido o total de cada dia por 3 individualmente e feito a soma.

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?

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

ORACLE Academy

Em uma instrução matemática complexa com vários operadores na mesma linha, como o computador escolhe qual operador ele deve usar primeiro? Para fazer operações matemáticas consistentes, a linguagem de programação Java segue as regras matemáticas padrão de precedência do operador.

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