transforme-se

serasa experian

WISEHANDS
Tecnologia e Acessibilidade

JAVA
AULA 4 - OPERAÇÕES MATEMÁTICAS E DE STRING

serasa experian

WISEHANDS
Tecnologia e Acessibilidade

Operações matemáticas

A linguagem Java possui uma classe chamada Math que contém diversos métodos especializados em realizar cálculos matemáticos. Observe a seguinte sintaxe:

Math.<nome_do_método>(argumentos)

Os métodos da classe Math são estáticos (vistos mais à frente) e por isso seguem a notação “Classe.nome do método”. Não é necessário importar a classe Math em um programa para poder utilizar seus recursos, pois ela já faz parte do pacote java.lang, disponível com o Java.

Método ceil

O método ceil tem como função realizar o arredondamento de um número (do tipo float ou double) para o seu próximo inteiro, por exemplo: o próximo inteiro de 1.8 é 2, de 5.5 é 6, de 4.1 é 5, e assim por diante. Sua sintaxe é a seguinte:

Math.ceil(valor)

```
1   double valorArrendodado = Math.ceil(8.456);
2
3   System.out.println("Valor Arrendodado: " + valorArrendodado);
```

```
Valor Arrendodado: 9.0
```

Método ceil

O método ceil tem como função realizar o arredondamento de um número (do tipo float ou double) para o seu próximo inteiro, por exemplo: o próximo inteiro de 1.8 é 2, de 5.5 é 6, de 4.1 é 5, e assim por diante. Sua sintaxe é a seguinte:

Math.ceil(valor)

```
1   double valorArrendodado = Math.ceil(8.456);
2
3   System.out.println("Valor Arrendodado: " + valorArrendodado);
```

```
Valor Arrendodado: 9.0
```

Método ceil

Exemplo: Quantos ônibus serão necessários para transportar um número X de passageiros que será digitado por um usuário?

```
1    double valorArrendodado = Math.ceil(8.456);
2
3    System.out.println("Valor Arrendodado: " + valorArrendodado);
```

```
Valor Arrendodado: 9.0
```

Método floor

Assim como ceil, o método floor também é utilizado para arredondar um número, mas para o seu inteiro anterior, por exemplo: o inteiro anterior de 1.1 é 1, de 2.9 é 2 e de 6.54 é 6. Sua sintaxe é a mesma do método ceil:

Math.floor(valor)

```
1  float nota = Float.parseFloat(JOptionPane.showInputDialog("Nota da prova: "));
2  JOptionPane.showMessageDialog(null, "Nota original: " + nota +
3     "\nArredondada para baixo: " + Math.floor(nota));
```

O exemplo apresenta o exemplo de um professor exigente. A nota da prova que o aluno tirou será arredondada para baixo. Na linha 5 o usuário digita o valor da nota que é convertido para um tipo float e na linha 7 esse valor é arredondado para baixo por meio do método floor.

Métodos round, max, min, sqrt, pow e abs

Método	Sintaxe	Descrição
round	Math.round(<valor>)	Recebe um valor numérico e retorna esse valor arredondado. Para valores decimais <0.5 arredonda para baixo, para valores >=0.5 arredonda para cima. Exemplos: Math.round(2.35) → 2, Math.round(2.59) → 3
max	Math.max(<valor1>,<valor2>)	Recebe dois valores numéricos e retorna o maior deles. Exemplo: Math.max(10,20) → 20
min	Math.min(<valor1>,<valor2>)	Recebe dois valores numéricos e retorna o menor deles. Exemplo: Math.max(10,20) → 10
sqrt	Math.sqrt(<valor>)	Recebe um valor numérico e retorna sua raiz quadrada. Exemplo: Math.max(25) → 25
pow	Math.pow(<valor1>,<valor2>)	Recebe dois valores numéricos (o operando e o expoente) e eleva o primeiro valor ao segundo. Exemplo: Math.max(10,2) → 100
abs	Math.abs(<valor>)	Recebe um valor numérico e retorna seu valor absoluto, desconsiderando o sinal. Exemplo: Math.max(-2) → 2

Método random

O método random da classe Math é utilizado para gerar valores de forma aleatória. Toda vez que o método random é chamado, sorteia-se um valor do tipo double entre 0.0 e 1.0 (o valor 1 nunca é sorteado).

Nem sempre essa faixa de valores é suficiente numa aplicação real. Por exemplo, para simular o sorteio de números entre 0 e 99 para um jogo de loteria qualquer, torna-se necessário o sorteio de números inteiros aleatórios no intervalo de 0 a 99.

Para que esses números possam ser sorteados, é preciso utilizar o operador de multiplicação (*) em conjunto com o método random. Com isso torna-se possível definir o intervalo em que o número será sorteado. O conversor (int) também pode ser usado para truncar a parte do ponto flutuante (a parte depois do ponto decimal) para que um número inteiro seja gerado, da seguinte forma:

```
(int)(Math.random() * 100)
```

Com isso seriam gerados números inteiros entre 0 e 99, atendendo plenamente à necessidade exposta.

Método random

```
1   for(int cartao = 1; cartao <= 4; cartao++){
2       String numerosCartao = "";
3       for(int numCartao = 1; numCartao <= 6; numCartao++){
4           int num = (int) (Math.random() * 100);
5           numerosCartao += num + "    ";
6       }
7       JOptionPane.showMessageDialog(null, "Números do cartão: " + cartao +
8                                     "\n"  + numerosCartao);
9   }
```

Exemplo que gera 4 cartões de loteria com 6 números cada.

Formatação com a classe DecimalFormat

Os cálculos matemáticos, em especial os que envolvem multiplicação e divisão, podem gerar resultados com muitas casas decimais. Isso nem sempre é necessário e esteticamente correto, pois apresentar um resultado com muitas casas decimais não é muito agradável nem legível à maioria dos usuários.

Por exemplo, considere duas variáveis do tipo double: x=1 e y=6. Ao realizar a divisão de x por y, aparece na tela o resultado 0.1666666666666666, que não é o mais adequado para se apresentar na tela. Seria mais conveniente mostrar o resultado formatado com duas ou três casas decimais.

Para realizar a formatação, é necessário definir um modelo de formatação, conhecido pelo nome pattern. Considere pattern o estilo de formatação que será apresentado sobre um valor numérico. Em outras palavras, você terá de informar ao compilador qual estilo de formatação deve ser usado para apresentar o número.

Para definir o pattern são usados caracteres especiais;

Formatação com a classe DecimalFormat

Caractere	Significado
0	Imprime o dígito normalmente, ou, caso ele não exista, coloca 0 em seu lugar. Exemplo: sejam as variáveis int x=4, y=32 e z=154, ao usar o pattern "000", o resultado impresso na tela seria x à 004, y à 032 e z à 154.
#	Imprime o dígito normalmente, desprezando os zeros à esquerda do número. Exemplo: sejam as variáveis double x=0.4 e y=01.34, ao usar o pattern "##.##", o resultado impresso na tela seria x à .4, y à 1.34.
.	Separador decimal ou separador decimal monetário (depende do sistema usado).
-	Sinal de número negativo.

Formatação com a classe DecimalFormat

```java
public static void main(String[] args) {
    DecimalFormat df = new DecimalFormat();
    float pagamento = 2583.75f;
    df.applyPattern("R$ #,##0.00");
    System.out.println("Pagamento: " + df.format(pagamento));
}
```

```
Pagamento: R$ 2.583,75
```

Internacionalização com a classe Locale

Internacionalização com a classe Locale

A linguagem Java é utilizada no mundo todo. Em função disso, um mesmo software feito em Java pode ser utilizado por usuários espalhados pelo globo. Cada país ou região adota certos formatos para representação monetária, apresentação de datas etc., os quais são definidos pelo sistema operacional da máquina e ficam armazenados como configurações locais. O separador de decimais, por exemplo, pode ser um ponto (.) ou uma vírgula (,), dependendo da região.

A classe Locale permite identificar certas propriedades da máquina em que o software é executado. Dessa forma, torna-se possível utilizar os recursos do Java, configurando determinados formatos de maneira automática.

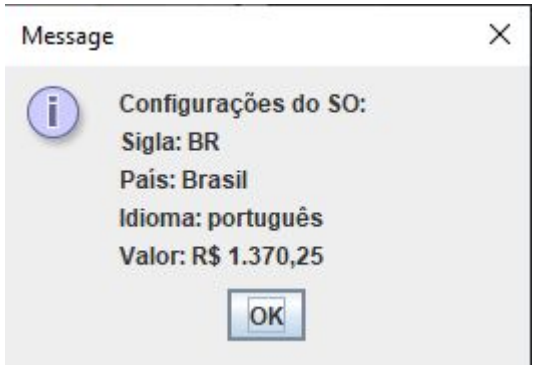
Internacionalização com a classe Locale

A linguagem Java é utilizada no mundo todo. Em função disso, um mesmo software feito em Java pode ser utilizado por usuários espalhados pelo globo. Cada país ou região adota certos formatos para representação monetária, apresentação de datas etc., os quais são definidos pelo sistema operacional da máquina e ficam armazenados como configurações locais. O separador de decimais, por exemplo, pode ser um ponto (.) ou uma vírgula (,), dependendo da região.

A classe Locale permite identificar certas propriedades da máquina em que o software é executado. Dessa forma, torna-se possível utilizar os recursos do Java, configurando determinados formatos de maneira automática.

Internacionalização com a classe Locale

```
1  DecimalFormat df = new DecimalFormat();
2  Locale local = Locale.getDefault();
3  double valor = 1370.25;
4  if(local.getCountry().equals("BR")) {
5      df.applyPattern("R$ #,##0.00");
6  }
7  JOptionPane.showMessageDialog(null, "Configurações do SO: "
8                                  + "\nSigla: " + local.getCountry()
9                                  + "\nPaís: " + local.getDisplayCountry()
10                                 + "\nIdioma: " + local.getDisplayLanguage()
11                                 + "\nValor: " + df.format(valor));
12
13 System.exit(0);
```



OPERAÇÕES COM STRINGS

Operações com Strings

String é um tipo texto que corresponde à união de um conjunto de caracteres. Em Java, uma variável do tipo string é uma instância da classe String, isto é, gera objetos que possuem propriedades e métodos, diferentemente dos tipos primitivos como int, float, double etc.

Essas strings podem ser manipuladas de várias formas. Por exemplo, é possível verificar seu comprimento, retirar uma parte dela, acessar ou mudar caracteres individuais. As strings constituem uma cadeia de caracteres entre aspas. Exemplo: frase = “Linguagem Java”. Da mesma forma que as operações matemáticas, existem diversos métodos para manipulação de strings, os quais acompanham a seguinte sintaxe:

<Nome-String>.<nome-método>(<argumentos>)

Operações com Strings

String é um tipo texto que corresponde à união de um conjunto de caracteres. Em Java, uma variável do tipo string é uma instância da classe String, isto é, gera objetos que possuem propriedades e métodos, diferentemente dos tipos primitivos como int, float, double etc.

Essas strings podem ser manipuladas de várias formas. Por exemplo, é possível verificar seu comprimento, retirar uma parte dela, acessar ou mudar caracteres individuais. As strings constituem uma cadeia de caracteres entre aspas. Exemplo: frase = “Linguagem Java”. Da mesma forma que as operações matemáticas, existem diversos métodos para manipulação de strings, os quais acompanham a seguinte sintaxe:

<Nome-String>.<nome-método>(<argumentos>)

Método length

O método length é utilizado para retornar o tamanho de uma determinada string, incluindo também os espaços em branco presentes nela. Esse método retorna sempre um valor do tipo int. Veja sua sintaxe:

<Nome-String>.lenght()

```
1  String frase = JOptionPane.showInputDialog("Forneça uma frase: ");
2  int tamanho = frase.length();
3  JOptionPane.showMessageDialog(null, "Frase: " + frase
4                                + "\nQuantidade de caracteres: " + tamanho );
```

Método charAt

O método charAt é usado para retornar um caractere de determinada string de acordo com um índice especificado entre parênteses. Esse índice se refere à posição do caractere na string, sendo 0 (zero) o índice do primeiro caractere, 1 (um) o do segundo e assim por diante. O método charAt é útil quando for necessário verificar a existência de um caractere na string. Por exemplo, suponha que uma determinada string só possa conter números - o método charAt pode ser usado para verificar a existência de dígitos numéricos nessa string. A sintaxe do método charAt é a seguinte:

<Nome-String>.charAt(<indice>)

Método charAt

```
1  String palavra = JOptionPane.showInputDialog("Forneça uma palavra com pelo menos quatro letras: ");
2  JOptionPane.showMessageDialog(null, "Palavra: " + palavra
3                                  + "\nCaractere 1: " + palavra.charAt(0)
4                                  + "\nCaractere 2: " + palavra.charAt(1)
5                                  + "\nCaractere 3: " + palavra.charAt(2)
6                                  + "\nCaractere 4: " + palavra.charAt(3) );
```

Métodos toUpperCase e toLowerCase

Os métodos toUpperCase e toLowerCase são utilizados para transformar todas as letras de uma determinada string em maiúsculas ou minúsculas. O método toUpperCase transforma todos os caracteres de uma string em maiúsculos. O método toLowerCase transforma todos os caracteres de uma string em minúsculos. Sua sintaxe é a seguinte:

<Nome-String>.toUpperCase() ou <Nome-String>.toLowerCase()

```
1  String frase = JOptionPane.showInputDialog("Forneça uma frase: ");
2  JOptionPane.showMessageDialog(null, "Frase: " + frase
3                                    + "\nEm maiusculas: " + frase.toUpperCase()
4                                    + "\nEm minusculas: " + frase.toLowerCase() );
```

transforme-se

O conhecimento é o poder
de transformar o seu futuro.