Classes Math, DecimalFormat e Locale

Instituto Federal do Espirito Santo – Ifes
Campus Colatina – Programação Orientada a Objetos I
Professor: Giovany Frossard Teixeira

Classe Math

- A classe Math é responsável pelas funcionalidades de cálculos matemáticos.

- Os métodos da classe Math, que utilizaremos, são todos estáticos. Dessa forma para utilizar esses métodos basta fazer:

  Math.<nome_do_metodo>(<argumentos>)

- A classe Math define duas constantes matemáticas:

  Math.PI = 3.14159265358979323846

  Math.E = 2.718282828459045223 54

Método ceil

- Método que dado um número do tipo *double* retorna o seu próximo *inteiro*. Sintaxe:

  Math.ceil(< valor double >)

Ex:

```
public class Ceil {

    public static void main(String[] args) {

        double a = 6.2, b = 6.6, c = -6.8;

        System.out.println(Math.ceil(a)); // resposta 7.0

        System.out.println(Math.ceil(b)); // resposta 7.0

        System.out.println(Math.ceil(c)); // resposta -6.0

    }

}
```

Método floor

- Método que dado um número do tipo *double* retorna o seu *inteiro* anterior. Sintaxe:

```
Math.floor(< valor double >)
```

Ex:

```
public class Floor{
    public static void main(String[] args) {
        double a = 6.2, b = 6.6, c = -6.8;
        System.out.println(Math.floor(a));  // resposta 6.0
        System.out.println(Math.floor(b));  // resposta 6.0
        System.out.println(Math.floor(c));  // resposta -7.0
    }
}
```

Método max

- Retorna o maior entre dois números. Sintaxe:

```
Math.max(<valor1>, <valor2>)
```

Ex:

```
public class Max {
    public static void main(String[] args) {
        int a = 20, b = 30;
        double c = -5.9, d = - 4.5;
        System.out.println(Math.max(a, b));
        System.out.println(Math.max(c, d));
    }
}
```

Método min

- Retorna o menor entre dois números. Sintaxe:

  Math.min(<valor1>, <valor2>)

Ex:

```
public class Min {
    public static void main(String[] args) {
        int a = 20, b = 30;
        double c = -5.9, d = - 4.5;
        System.out.println(Math.min(a, b));
        System.out.println(Math.min(c, d));
    }
}
```

Método sqrt

- Retorna a raíz quadrada de um número.

  Math.sqrt(< valor do tipo double >)

Ex:

```
public class Sqrt {
  public static void main(String[] args) {
    double a = 900;
    System.out.println("A raíz quadrada de 900 é: " + Math.sqrt(a));
  }
}
```

Método pow

- Retorna a potência de um número.

  Math.pow(< valor da base >, < valor da potência >)

Ex:

```
public class Pow {
    public static void main(String[] args) {
        double a = 30, b = 2;
        System.out.println("O quadrado de 30 é: " + Math.pow(a, b));
    }
}
```

Método random

- Retorna um número randômico entre 0 e 1 (com intervalo aberto em 1, ou seja, 1 não pode ser sorteado).

```
Math.random()
```

- Obs. Importante notar os parenteses antes da conversão para inteiro, isso porque a conversão para inteiro trunca o valor double. Experimente colocar 0.999 ao invés de (Math.random() * 100)

Ex:

```
public class Random {

    public static void main(String[] args) {

        int num = ( int )(Math.random() * 100);

        System.out.println("Randomico entre 0 e
        99 " + num);

    }

}
```

Classe DecimalFormat

- A classe DecimalFormat é utilizada para definir um modelo de formatação para um número (*pattern*).

- Se quiséssemos imprimir o valor de PI (3.14159265358979323846) com apenas 3 dígitos, 1 antes do separador decimal e 2 depois, ou seja, 3.14. Precisaríamos da classe DecimalFormat para fazê-lo.

Classe DecimalFormat - Principais caracteres especiais

Caracteres	Significado
0	Mostra o dígito, se ele não existe, coloca zero e, seu lugar
#	Mostra o dígito, se zero à esquerda é ignorado
.	Separador decimal ou monetário
,	Separador de grupo
substrings	Conjunto de caracteres fixo, pode ser colocado antes ou depois do *pattern*.

- **As substrings são válidas se diferentes de quaisquer caracteres especiais, para mais informações sobre outros caracteres especiais consulte a documentação do Java.**

Exemplo

```
package decimalFormat;

import java.text.DecimalFormat;

public class DecimalFormat01 {

  public static void main(String[] args) {

        DecimalFormat df = new DecimalFormat();

        df.applyPattern("0.00");

        System.out.println("O valor de PI eh:" +
        df.format(Math.PI));



        double estoque = 198564.45;

        df.applyPattern("#,##0,000.00");

        System.out.println("O valor do estoque
         eh:" + df.format(estoque));

        df.applyPattern("Jonas possui R$
          #,##0,meio000.00 teste");

        System.out.println("O valor do estoque  eh:" +
        df.format(estoque));

    }

}
```

- Obs. É interessante notar que se uma substring é colocada no meio do pattern ela é enviada para após a formatação do número (ver a substring “meio”)

A classe Locale

- Cada país ou região adota certos formatos para representação monetária, apresentação de datas, etc. Esses formatos são definidos pelo Sistema Operacional da máquina e ficam armazenados como configurações locais. O separador de decimal 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 está sendo executado. Dessa forma, torna-se possível utilizar recursos do Java, configurando determinados formatos de maneira automática.

Exemplo

```
package locale;

import java.text.DecimalFormat;

import java.util.Locale;

public class Locale01 {

    public static void main(String[] args) {

        DecimalFormat df = new
        DecimalFormat();

        Locale local = Locale.getDefault();

        System.out.println("Configurações do
        Sistema Operacional: ");

        System.out.println("Sigla: " +
        local.getCountry());

        System.out.println("Pais: " +
        local.getDisplayCountry());

        System.out.println("Lingua: " +
        local.getDisplayLanguage());

        System.out.println("Teclado: " +
        local.getDisplayName());

        System.out.println();

        double valor = 1370.25;

        if ( local.getCountry().equals("BR"))

            df.applyPattern("R$ #,##0.00");

        System.out.println("Valor: " +
         df.format(valor));

    }

}
```

Dúvidas?