Capítulo – 10

10.1 Operações matemáticas

A linguagem Java possui uma classe chamada `Math` que contém diversos métodos especializados em realizar cálculos matemáticos sem a necessidade de o programador ter que "reinventar a roda" ou elaborar métodos que já existam nas classes do Java.

A forma de utilizar a classe `Math` é:

Estes argumentos geralmente são separados por vírgula

```
Math.nome_do_metodo(argumentos ou lista de argumentos)
```

A classe `Math` já está embutida em `java.lang`, portanto já está disponível no java, não havendo a necessidade de importar ou instanciar a classe como fizemos com a classe Scanner vista anteriormente. Isto se deve ao fato de que os métodos da classe `Math` são métodos estáticos onde aprenderemos mais a este respeito nos próximos capítulos.

10.2 Obtendo constantes matemáticas.

A classe `Math` traz em si a definição de duas constantes matemáticas, uma delas é o valor de pi que é uma constante (π = 3.1415926535897932846) .

Outro valor é o da base para logaritmos naturais (2.7182818284590452354).

Veja abaixo a classe `Math` em funcionamento:

```
3 public class PieLogaritmos {
4
5     public static void main(String[] args) {
6         System.out.println("Valor de PI "+ Math.PI);
7         System.out.println("Valor da base para logaritmos "+Math.E);
8     }
9
10 }
11
```

```
Console
<terminated> PieLogaritmos [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (26/03/2015 18
Valor de PI 3.141592653589793
Valor da base para logaritmos 2.718281828459045
```

Devemos notar que na frente de `Math` temos o ponto e a palavra PI com letras maiúsculas, isso nos indica de PI é uma constante como se estivéssemos declarando:

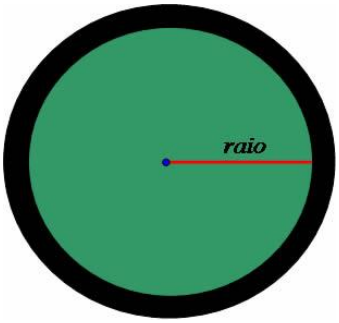
```
final double  PI = 3.141592653589793;
```

Ou seja a variável PI é do tipo final seu valor não pode ser alterado. Note que no código abaixo foi gerado um erro pelo compilador, pois a variável PI não pode ter um novo valor atribuído a ela.

```
3 public class PieLogaritmos {
4
5     public static void main(String[] args) {
6         final double PI = Math.PI;
7         PI = 898.25;
8
9         System.out.println("Valor de PI "+ Math.PI);
10        System.out.println("Valor da base para logaritmos "+Math.E);
11    }
12
13 }
```

10.3 Cálculo comprimento de um círculo.

Perto da casa do Sr. José foi inaugurada uma praça com pista de lazer e corrida no formato circular. Tentando descobrir uma forma de medir quantos metros irá percorrer numa volta completa na pista circular, ele descobriu uma interessante forma de calcular essa distância. Sr. José desvendou que para a medição ele precisaria determinar a medida do raio da praça, que é a distância entre o centro da praça e a pista de corrida.



Em suas pesquisas ele descobriu que precisaria multiplicar a medida do raio por 2 e por um número chamado de pi (símbolo: π). O número pi está presente em todos os cálculos envolvendo formatos circulares e seu valor único é igual a 3,14. Portanto, para determinarmos o comprimento dessa praça e de todas de formato circular realizamos o seguinte cálculo:

$$C = 2 * \pi * r$$

C: comprimento

π: 3,14

r: medida do raio

A medida do raio dessa praça é de 50 metros, então:

C = 2 * 3,14 * 50

C = 314 metros

Portanto, uma volta completa nessa praça corresponde à distância de 314 metros.

Disponível em:< http://www.escolakids.com/o-comprimento-do-circulo.htm>: acessado em 27/032015

Agora vamos desenvolver uma classe Java para automatizar a resolução deste problema.

```
3 public class ConstantesPI {
4
5         public static void main(String[] args) {
6             float raio = 50f;
7
8             double comprimento = 2 * raio * Math.PI;
9
10            System.out.println(comprimento);
11        }
12 }
13
14
```

```
Console
<terminated> ConstantesPI [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (27/03/2015
314.1592653589793
```

O Sr. José só não sabia que a precisão do Java seria tanto assim!

Com isso temos um resultado muito exato que foram ‘os `314.1592653589793` metros que será percorrido pelo Sr. José. Imagine para ele calcular a Km total da sua corrida diária se tiver que multiplicar este valor por 5 voltas. Ou ainda de se percorrer 10Km e desejar saber quantas voltas foram dadas em volta da praça dividindo 10.000 por `314.1592653589793` seriam incríveis `31,830988618379069569908480881 96` voltas. Aí o Sr. José já não entenderia nada!

Para resolver este problema temos o método `ceil` da classe `Math` visto no próximo tópico.

10.4 Método Ceil (arredondando para cima)

`Math.ceil(<valor>);`

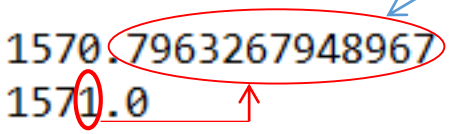
Ao se aplicar o comando `Math.ceil` podemos realizar o arredondamento de valores do tipo `float` ou `double` para seu próximo inteiro, ou seja, no exemplo acima do Sr. José poderíamos aplicar este método para arredondar os valores decimais, simplificando o resultado e tornando a aplicação mais robusta.

Cuidado! O método `ceil` normalmente é utilizado para fornecer a saída de dados mas pode ser utilizado para arredondar o valor dentro de uma variável também.

Exemplo o Sr. José –II a Saga!

```
3 public class MathCeil {
4
5     public static void main(String[] args) {
6         float raio = 50f;  // raio da praça do Sr.José
7         int voltas = 5;
8
9         double comprimento = 2 * raio * Math.PI;
10
11        double totPercorrido = comprimento * voltas ;
12
13        System.out.println("Metros percorridos em 5 voltas na praça sem (Math.ceil): "+ totPercorrido);
14        System.out.println("Metros percorridos em 5 voltas na praça com (Math.ceil): "+ Math.ceil(totPercorrido));
15    }
16
17 }
18
19
```

```
Problems  Javadoc  Declaration  Console
<terminated> MathCeil [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (27/03/2015 08:50:03)
Metros percorridos em 5 voltas na praça sem (Math.ceil): 1570.7963267948967
Metros percorridos em 5 voltas na praça com (Math.ceil): 1571.0
```



Como podemos observar o método `ceil` arredondou para o próximo inteiro, ou seja, se aplicarmos `Math.ceil(5.1)` o Java retornará 6.0.

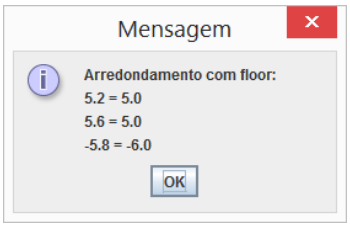
10.5 Método Floor(arredondando para baixo)

Assim como o `ceil`, o método `floor` também é utilizado para arredondar um número, mas para o seu inteiro anterior. A forma de uso é a mesma do `ceil`:

`Math.floor(<valor>)`

O exemplo abaixo mostra o uso de método `floor`.

```java
import javax.swing.JOptionPane;

public class MathFloor {

    public static void main(String[] args) {
        double n1 = 5.2, n2 = 5.6, n3 = -5.8;
        JOptionPane.showMessageDialog(null, "Arredondamento com floor: "+
                        "\n" + n1 + "=" + Math.floor(n1) +
                        "\n" + n1 + "=" + Math.floor(n2) +
                        "\n" + n1 + "=" + Math.floor(n3));
    }
```



10.6 método max e min

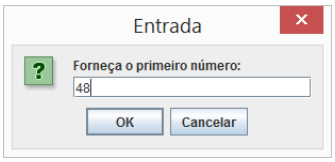
Utilizamos o método `max` para verificar o maior valor entre dois números, que podem ser do tipo `double`, `float`, `int` ou `long`. A sua forma de aplicação é a seguinte:

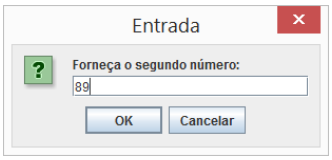
```
Math.max(<valor1>, <valor2>);
```

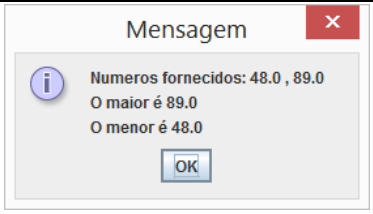
O método min fornece o resultado oposto ao método `max`, sendo então utilizado para obter o valor mínimo entre dois números. Do mesmo modo que o método `max`, esses números também podem ser `double`, `float`, `int` ou `long`. A sua forma de aplicação é a mesma do método `max`:

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

```java
import javax.swing.JOptionPane;

public class MathMinMax {

    public static void main(String[] args) {
        float num1 = Float.parseFloat(JOptionPane.showInputDialog("Forneça o primeiro número: "));
        float num2 = Float.parseFloat(JOptionPane.showInputDialog("Forneça o segundo número: "));

        JOptionPane.showMessageDialog(null, "Numeros fornecidos: "+num1 + " , " + num2 +
                "\nO maior é "+Math.max(num1, num2)+
                "\nO menor é " +Math.min(num1, num2));
        }
}
```







O cálculo do maior número pode ocorrer entre dois números do mesmo tipo de dado ou não, ou seja, podemos obter o maior ou menor valor entre um número do tipo `double` e outro do tipo `int`.

10.7 Método `sqrt` (potencias e raízes)

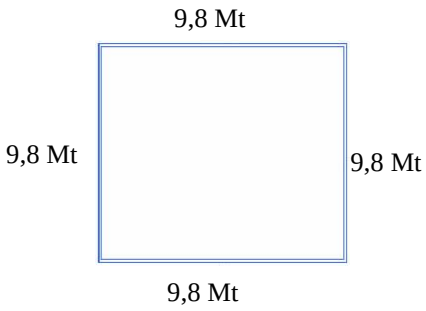
sqrt (número) - calcula a raíz quadrada de um número
pow (base, expoente) - calcula a potência da base elevada ao expoente.
cbrt (número) - calcula a raiz cúbica de um número.

O método sqrt da classe `Math` realiza o cálculo da raiz quadrada de um número. O resultado obtido é um número do tipo `double`. Veja a forma de aplicá-lo:

```
Math.sqrt(<valor_double>);
```

Lembra do Sr José? Pois é veja o caso abaixo:

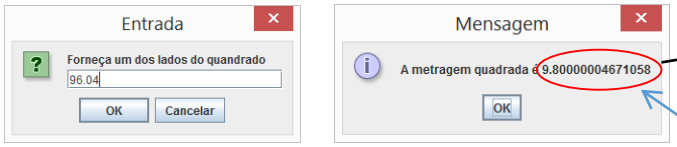
O Sr. José sempre gostou de reformar sua casa. Desta vez ele vai trocar o piso da sala, mas precisa saber qual a metragem de cada um dos lados da sala para saber a área e escolher o rodapé. Ele recebeu o piso em casa com a medida total de 96,04 Mts. Se a sala do Sr José tem todos os lados iguais, como calcular essa medida?. Dessa forma devemos aplicar a fórmula √96,04.



96,04Mt

Agora utilizaremos uma classe Java que através do método `srqt` retorne o valor de cada um dos 4 lados da sala do Sr José.

```
3 import javax.swing.JOptionPane;
4
5 public class MathSrqt {
6
7     public static void main(String[] args) {
8         float medida  = Float.parseFloat(JOptionPane.showInputDialog("Forneça um dos lados do quandrado"));
9
10        JOptionPane.showMessageDialog(null, "A metragem quadrada é " + Math.sqrt(medida));
11    }
12
13 }
```



Então a medida de cada lado é: 9,8 Mt

Vamos melhorar esta apresentação com o método a classe `DecimalFormat` do Java nos próximos tópicos.

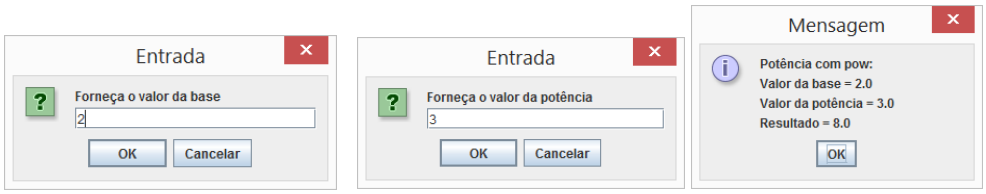
10.8 Método pow(potência)

O método `pow` da classe `Math` realiza a potenciação (elevar uma base a uma potência). Sua forma de utilização é a seguinte:

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

Ex: `Math.pow(2,3)` isto é igual a expressão 2^3 (dois ao cubo).

```
3 import javax.swing.JOptionPane;
4
5 public class MathPow {
6
7     public static void main(String[] args) {
8         float base = Float.parseFloat(JOptionPane.showInputDialog("Forneça o valor da base"));
9         float potencia = Float.parseFloat(JOptionPane.showInputDialog("Forneça o valor da potência"));
10
11        JOptionPane.showMessageDialog(null, "Potência com pow: " +
12        "\nValor da base = " + base +
13        "\nValor da potência = " + potencia +
14        "\nResultado = "+ Math.pow(base, potencia));
15    }
16
17 }
```



10.9 Método round(arredondando para inteiro)

Forma de uso:

```
Math.round(valor)
```

Arredonda um valor numérico de ponto-flutuante para o inteiro mais próximo.

Onde valor é um valor numérico de ponto-flutuante. Por exemplo 9,56

Se a parte decimal do valor a ser arredondado for 0,5 ou maior, então o resultado será igual ao menor inteiro maior que o argumento para o método. Do contrário, o método `round` retornará o maior inteiro menor ou igual ao valor fornecido como argumento.

Como arredondar um valor numérico de ponto-flutuante para o inteiro mais próximo:

```
3 public class MathRound {
4
5     public static void main(String[] args) {
6         float valor = 1.5f;
7           System.out.println("O resultado do arredondamento de " +
8                           valor + " é " + Math.round(valor));
9
10
11    }
12
13 }
```

```
Problems  Javadoc  Declaration  Console
<terminated> MathRound [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (27/03/2015 11:10:0
O resultado do arredondamento de 1.5 é 2
```

Altere a sua classe para o valor 1,4 assim o `round` exibirá este numérico de ponto-flutuante para o inteiro anterior:

```
3 public class MathRound {
4
5     public static void main(String[] args) {
6         float valor = 1.4f;
7           System.out.println("O resultado do arredondamento de " +
8                         valor + " é " + Math.round(valor));
9
10
11    }
12
13 }
```

Problems | Javadoc | Declaration | Console

```
<terminated> MathRound [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (27/03/2
O resultado do arredondamento de 1.4 é 1
```

10.10 Método rint (arredondando para decimal)

O método `rint` retorna um valor `double` mais próximo do valor passado a ele. No exemplo abaixo como o valor decimal é abaixo de 0,5 então o `rint` arredonda para baixo.

```
3 public class MathRint {
4         public static void main(String[] args) {
5             float valor = 1.4684f;
6               System.out.println("O resultado do arredondamento de " +
7                             valor + " é " + Math.rint(valor));
8
9         }
10 }
```

Problems | Javadoc | Declaration | Console

```
<terminated> MathRint [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (27/03/20
O resultado do arredondamento de 1.4684 é 1.0
```

No exemplo abaixo como o valor decimal maior de 0,5 então o `rint` arredonda para cima.

```
3 public class MathRint {
4         public static void main(String[] args) {
5             float valor = 1.5684f;
6               System.out.println("O resultado do arredondamento de " +
7                             valor + " é " + Math.rint(valor));
8
9         }
10 }
```

Problems | Javadoc | Declaration | Console

```
<terminated> MathRint [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (27/03/2015 11:56:26)
O resultado do arredondamento de 1.5684 é 2.0
```

10.11 método random (números aleatórios)

Números aleatórios ou randômicos são obtidos usando o método `random()`que retorna um valor do tipo `double` em 0.0 e 1.0.

Para conseguirmos um valor limite ou um alcance (comumente chamado de range) delimitado, devemos fazer pequenas operações matemáticas.
Essas operações são simples e podem ser resumidas da seguinte maneira.

- O **limite inferior**, ou valor inicial (start value) será sempre *somado* ao número randômico.
- O **limite superior**, ou alcance (range) será sempre o **limite superior** *subtraído* o **limite inferior** e depois *multiplicado* ao número randômico.

Por exemplo, se quisermos que um número randômico sempre fique entre 5 e 10, procederíamos da seguinte maneira:

O menor número possível é **5**, portanto ele será nosso **start value** ou limite inferior.

O maior número possível é **10**, então 10 será nosso **limite superior**.

Como temos ambos os limites (inferior e superior) devemos criar um alcance (range). Para obtermos isso, subtrairemos o limite superior com o limite inferior (10-5=5).

```
3  public class MathRandom {
4
5      public static void main(String[] args) {
6          int limiteInferior = 5;
7          int limiteSuperior = 10;
8
9          int alcance = limiteSuperior - limiteInferior;
10
11         double nrRandomico = Math.random();
12
13         System.out.println("O número randômico escolhido entre 5 e 10 foi "
14         + Math.round(limiteInferior + nrRandomico * alcance));
15
16     }
17
18 }
```

Utilizado para transformar os valores do método `random` em inteiro

```
Problems  Javadoc  Declaration  Console
<terminated> MathRandom [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (27/03/2015 11:1
O número randômico escolhido entre 5 e 10 foi 9
```

Outro exemplo do uso do método random:

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

Com isso seriam gerados números inteiros entre 0 e 99. Neste caso o conversor (int) na frente foi usado para truncar ou força a conversão do valor do random e traz um número sem o ponto flutuante para que o número seja gerado da seguinte forma: Entre 0 e 99.

```java
public class MathrandomCartaoLoteria {

    public static void main(String[] args) {
        for (int i = 0; i <=5 ; i++){

            System.out.println("Número sorteado: "+Math.round(Math.random() * 100));

        }
    }
}
```

```
Problems  Javadoc  Declaration  Console
<terminated> MathrandomCartaoLoteria [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (27/03/2015 11
Número sorteado: 9
Número sorteado: 29
Número sorteado: 66
Número sorteado: 24
Número sorteado: 39
Número sorteado: 93
```

Vamos aprimorar o exemplo com a classe `JOptionPane` para simular um jogo de loteria. Utilizaremos o laço `for` para sortear uma sequência de 4 valores.

```java
import javax.swing.JOptionPane;

public class MathrandomCartaoLoteria {

    public static void main(String[] args) {
        for (int cartao = 1; cartao <=4 ; cartao++){
            String numerosCartao = "";
            for (int numcartao = 1; numcartao <=6 ; numcartao++){
                int num = (int) (Math.random() * 100);
                numerosCartao += num + " ";
            }

            JOptionPane.showMessageDialog(null, "Numero do cartão: " + cartao +
                                                "\n" + numerosCartao);
        }
    }
}
```

10.12 Formatação com a classe DecimalFormat

Os cálculos matemáticos, em especial os que envolvem multiplicação e divisão, podem gerar resultados com muitas casas decimais como no caso do Sr. José. Isso nem sempre é necessário e esteticamente correto, pois apresentar um resultado com muitas casas decimais não é muito agradável nem legível a 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 0,666666666666666, que não é o mais adequado para se apresentar. Seria mais conveniente mostrar o resultado formatado com duas ou três casas decimais. Para realizar esta formatação, é necessário definir um modelo de formatação. Pense por exemplo a exibição de valores monetários onde temos os símbolos R$ junto com o valor decimal e no máximo duas casas após a vírgula. Esse modelo de formatação da classe `DecimalFormat` é conhecido como **`pattern`** como o estilo de formatação que será apresentado sobre um valor numérico.

Para definir um `pattern` são usados caracteres especiais: Veja a tabela.

Caractere	Significado	O que faz
0	Imprime o dígito normalmente, ou caso ele não exista, coloca 0 em seu lugar.	Sejam as variáveis int x =4, y =32 e z =154, ao usar o pattern “000”, o resultado impresso na
#	Imprime o dígito normalmente, desprezando os zeros à esquerda do número.	Sejam as variáveis double x = 0.4 e y = 01.34, ao usar o pattern “##.##”, o resultado na tela será x .4, y 1,34
.	Separador decimal ou separador decimal monetário.	
-	Sinal de número negativo	

A classe `DecimalFormat` precisa ser importada pois está no pacote java.text.

```
2 import java.text.DecimalFormat;
3
4 public class DecimalFormatValor {
5
6     public static void main(String[] args) {
7         DecimalFormat df = new DecimalFormat();
8         short idade =38;
9         df.applyPattern("000");
10        System.out.println(df.format(idade));
11
12        int quantidade = 9750;
13        df.applyPattern("#0,000");
14        System.out.println(df.format(quantidade));
15
16                long estoque = 198564;
17        df.applyPattern("#,##0,000");
18        System.out.println(df.format(estoque));
19
20        float altura = 1.74f;
21        df.applyPattern("#0.00");
22        System.out.println(df.format(altura));
23
24        double peso = 70.25;
25        df.applyPattern("#0.00");
26        System.out.println(df.format(peso));
27
28        String valorReais = "2583.75";
29        df.applyPattern("R$ #,##0.00");
30        System.out.println(df.format(Double.parseDouble(valorReais)));
31
32    }
33 }
```

Problems | Javadoc | Declaration | Console

```
<terminated> DecimalFormatValor [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (27/03/2015 12:30:
038
9.750
198.564
1,74
70,25
R$ 2.583,75
```

10.13 FormatCurrency

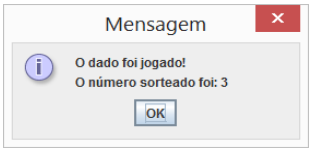
```
2 import java.util.Scanner;
3 import java.text.NumberFormat;
4 import java.util.Locale;
5
6 public class FormatCurrencyMoeda{
7     public static void main(String[] args){
8         Scanner s = new Scanner(System.in);
9         System.out.print("INFORME A LARGURA DO PISO EM MT :");
10        String larg = s.nextLine();
11        double Largura = Double.parseDouble(larg);
12
13        System.out.print("INFORME O COMPRIMENTO PISO EM MT :");
14        String compr = s.nextLine();
15        double Comprimento = Double.parseDouble(compr);
16
17        System.out.println("\nO VALOR COBRADO E DE R$ 36,00 O  MT2!");
18
19        double valorcobrado;
20
21            valorcobrado = (Largura*Comprimento)*36;
22            NumberFormat formato1 = NumberFormat.getCurrencyInstance();
23
24            System.out.println("\n\nO VALOR TOTAL DO SERVICO SERA "+formato1.format(valorcobrado));
25    }
26
27 }
```

Problems | Javadoc | Declaration | Console

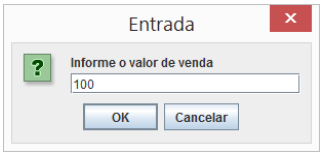
```
<terminated> FormatCurrencyMoeda [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (29/03/2015 17:11:50)
INFORME A LARGURA DO PISO EM MT :23
INFORME O COMPRIMENTO PISO EM MT :12

O VALOR COBRADO E DE R$ 36,00 O  MT2!


O VALOR TOTAL DO SERVICO SERA R$ 9.936,00
```

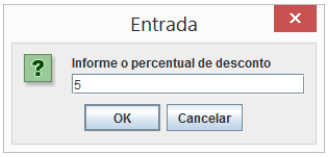
Atividades - 10

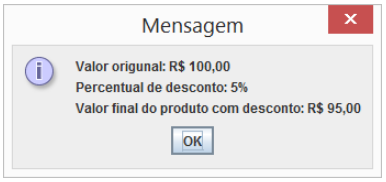
1) Crie uma classe que simule a jogada de um dado de seis lados (números de 1 a 6).



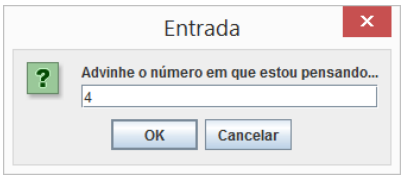
2) Um vendedor camarada oferece pequenos descontos na venda de seus produtos. Faça uma classe que receba o valor de venda e um percentual de desconto. Além de calcular o desconto, o valor final deve ser arredondado para baixo, isto é, para o valor inteiro, de acordo com a figura.

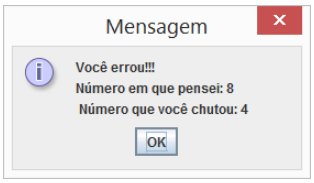






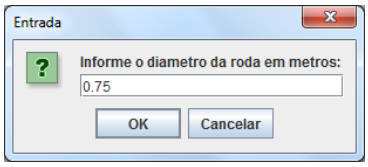
3) Crie uma classe que gere um número aleatoriamente (entre zero e 9) por `Math.randon`. Em seguida solicite ao usuário a digitação de um valor entre 0 e 9 e verifique se o número sorteado é igual ao número digitado pelo usuário.

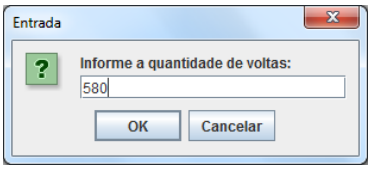


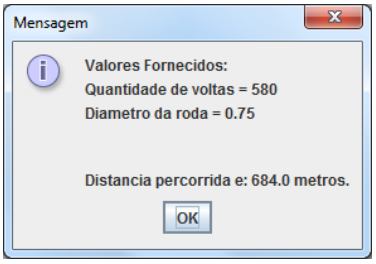


4) Um ciclista criou um aparelho que marca quantas voltas a roda de sua bicicleta dá em um determinado percurso. Com o uso desse aparelho é possível descobrir a distância percorrida desde que o raio do pneu da bicicleta seja fornecido. Faça uma classe que calcule a distância percorrida, em que devem ser fornecidos pelo usuário a quantidade de voltas que a roda da bicicleta deu e o diâmetro da roda em metros.

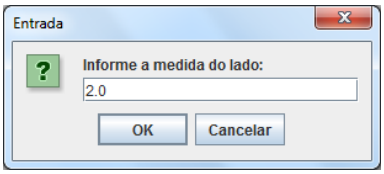
Para saber o comprimento da roda a partir do diâmetro fornecido pelo usuário, **use a fórmula c = π * d/2 (em que c = comprimento, π = 3.1416 e d = diâmetro da circunferência)**. Sabendo o comprimento da roda, basta multiplica-la pelo número de voltas para descobrir a distância do percurso**. Arredonde o resultado para seu próximo inteiro**. Veja na figura abaixo um exemplo de execução deste exercício. Use a classe JOptionPane.

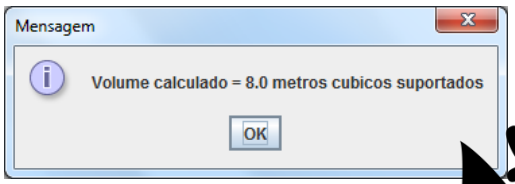






5) Crie uma classe que calcule quantos metros cúbicos de água suporta uma determinada caixa-d´água em forma de cubo (todos os lados são iguais). O usuário deve informar o valor de um dos lados e o volume de água será calculado pela fórmula Volume = Lado^3. Arredonde o resultado para seu inteiro anterior. A figura abaixo mostra a execução deste exercício.





Exercício Bônus.

6) Crie uma classe que desenhe uma moldura na tela. Essa moldura deve ter 80 caracteres de comprimento por 5 de altura. Para isso utilize a sequência de caracteres ASCII, como mostra a figura abaixo. Se você utiliza a plataforma Windows, para gerar esses caracteres você deve primeiramente segurar a tecla ALT (lado esquerdo) e digitar o código correspondente ao caractere no teclado numérico. Se você usa a plataforma LINUX ou outra qualquer, e encontrar dificuldades em gerar estes caracteres, utilize qualquer outra, pois o mais importante é resolver a lógica do exercício.

Códigos ASC e seus caracteres correspondentes:

201 ╔ = 205 = ═ 187 ╗
186 ║ ‖ 186
200 ╚ = 205 = ═ 188 ╝

A figura abaixo demonstra como deverá ser o resultado.