

AULA 7

```
public static void main(String[] args) {  
    // TODO code application logic here  
    int n1 = 3;  
    int n2 = 5;  
    float m = (n1 + n2 / 2);  
    System.out.println("A média é: " + m);  
}
```

Esse + é operador de adição

Esse + é operador de concatenação
(trata-se de uma string)

No swing:



Objetivo: Digitar um número para o numerador, outro para o denominador e quando eu mandar dividir ele vai mostrar quanto vale a divisão e o resto desta divisão

Nome das variáveis:

jTextField1 – txtNum

jTextField2 – txtDen

jButton1 – btnDividir

jLabel5 – lblDiv

jLabel6 – lblResto

No código-fonte:

- Criar uma variável para o numerador (n) e outra para o denominador (d)
- A variável para o numerador vai receber o que está na caixa de texto

LEMBRANDO: txtNum.getText retorna um valor **string** e estou tentando jogar dentro de uma variável **inteira**

ENTÃO: colocar Integer.parseInt

- Após, criar uma variável **float** chamada de *div*, no qual vai receber o numerador dividido pelo denominador
- Criar uma variável **float** chamada *res*, no qual vai receber o resto da divisão entre o numerador e o denominador (n%d)

- Após, mostrar no meu lblDiv.setText o div (que é minha variável de divisão)

LEMBRANDO: **float** não pode ser convertido para **string**!

ENTÃO: converter Float.toString(div)

`lblDiv.setText(div);` → `lblDiv.setText(Float.toString(div));`

- Fazer a mesma coisa para lblResto

CÓDIGO-FONTE COMPLETO:

```
private void btnDividirActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    int n = Integer.parseInt(txtNum.getText());  
    int d = Integer.parseInt(txtDen.getText());  
    float div = n / d;  
    float res = n % d;  
    lblDiv.setText(Float.toString(div));  
    lblResto.setText(Float.toString(res));  
}
```

OPERADORES UNÁRIOS

`++` → faz um incremento → ex: `a++` → é a mesma coisa que: `a = a + 1`

- Este operador soma uma unidade a uma determinada variável

`--` → faz um decremento → ex: `a--` → é a mesma coisa que: `a = a - 1`

- Este operador diminui uma unidade a uma determinada variável

OBS: a posição deste incremento ou decremento vai influenciar diretamente no resultado.

Pós incremento: 5 + numero ++: quando o ++ vem depois do número, significa: utilize o número e depois de fazer essa soma, ele soma uma unidade

Pré incremento: 5 + ++numero: quero que pegue o 5, e some com o numero já pré incrementado de uma unidade

Exemplos no código-fonte:

```
int numero = 5;  
numero++;  
System.out.println(numero);
```

A resposta é 6

```
int numero = 5;  
int valor = 5 + numero++;  
System.out.println(valor);
```

A resposta é 10

```
int numero = 5;  
int valor = 5 + ++numero;  
System.out.println(valor);
```

A resposta é 11

```
int numero = 10;
int valor = 4 + numero--;
System.out.println(valor);
System.out.println(numero);
```

Valor = 14
Numero = 9

- E se eu quiser somar 2 unidades?

Temos outros operadores para isso:

OPERADORES DE ATRIBUIÇÃO

`+=` → faz uma soma e atribui → `a += b` → é a mesma coisa que `a = a + b`

`-=` → ele subtrai e então atribui → `a -= b` → é a mesma coisa que `a = a - b`

EU VOU TER ISSO PARA TODOS OS OUTROS OPERADORES ARITMÉTICOS!

`*=`

`/=`

`%=`

OBS: o Java NÃO tem um operador de EXPONENCIAÇÃO! Para isso:

CLASSE MATH

- Ela possui constantes e métodos internos para a realização de cálculos matemáticos

Exemplos:

PI	Constante π	Math.PI	3.1415...
pow	Exponenciação	Math.pow(5,2)	25
sqrt	Raiz Quadrada	Math.sqrt(25)	5
cbrt	Raiz Cúbica	Math.cbrt(27)	3

Arredondamentos

- Abs: elimina o valor negativo. É o valor absoluto.

3 maneiras de arredondar:

- 1- Floor – arredonda para baixo
- 2- Ceil – arredonda para cima
- 3- Round – arredondamento aritmético

abs	Valor Absoluto	Math.abs(-10)	10
floor	Arredonda para Baixo	Math.floor(3.9)	3
ceil	Arredonda para Cima	Math.ceil(4.2)	5
round	Arredonda Aritmeticamente	Math.round(5.6)	6

Na prática:

Lembrar sempre das conversões!

```
float v = 8.3f;  
int ar = Math.floor(v);
```



```
float v = 8.3f;  
int ar = (int) Math.floor(v);
```

Math.floor(v) é **double**

ar é **int**

coloco um (int) antes do Math.floor(v)

Exemplos:

```
float v = 8.3f;  
int ar = (int) Math.floor(v);  
System.out.println(ar);
```

Resultado é 8

```
float v = 8.3f;  
int ar = (int) Math.ceil(v);  
System.out.println(ar);
```

Resultado é 9

O Java tem suporte para gerar NÚMEROS ALEATÓRIOS!

Math.**random**()

- gera números entre 0.0 e 1.0

E se eu quiser números aleatórios entre 5 e 10?

```
5 + Math.random() * (10-5)
```

No código-fonte:

```
double aleatorio = Math.random();  
int n = (int) (5 + aleatorio * (10-5));  
System.out.println(n);
```