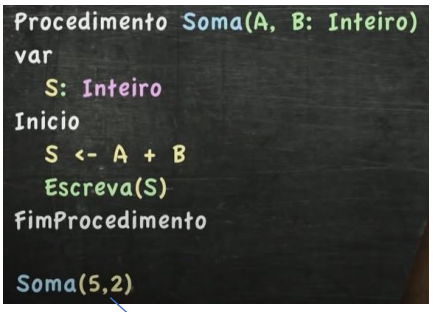
AULA 15 – MÉTODOS

- O Java não tem nenhuma palavra específica que indique um método!

- A identificação é feita através do tipo de retorno

- Se não tiver retorno, utiliza-se uma palavra específica

Procedimento em algoritmos:



Soma é o meu procedimento

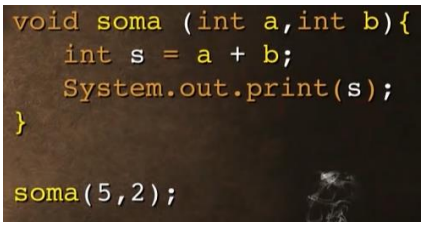
5 vai valer A e 2 vai valer B

(5,2) são os meus parâmetros

OBS: procedimentos **não retornam valor**!

Procedimento em Java:

- Para retornar valor em Java, eu devo utilizar uma palavra em específica: **void**



Void indica que não vai ter retorno!

No NetBeans:

```
public class TesteFuncao01 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

Como eu leio isso?

- **void** é um procedimento

- **main** é um método que não retorna valor que recebe um vetor como parâmetro (String [ ] args) que é um método estático e um método público

OBS: sempre que eu aperto play, o método executado por padrão é o **método main**. Então, se eu quero fazer uma chamada para soma, eu tenho que fazer dentro do método main.

```java
public class TesteFuncao01 {

    /**
     * @param args the command line arguments
     */

    void soma (int a, int b) {
        int s = a + b;
        System.out.println("A soma é " + s);
    }

    public static void main(String[] args) {
        // TODO code application logic here
        soma (5,2);
    }

}
```

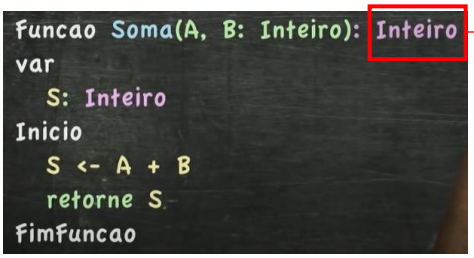
Nota-se que deu erro, isso porque o método main é estático, ou seja, é um método que serve para classe, não para uma instância. **Eu não posso chamar um procedimento dentro de um método estático se este procedimento não for estático**!

Para isso, é só **escrever static antes de void**. Quando eu coloco static na frente de um método eu o torno estático, fazendo com que este método seja apenas funcional dentro da classe, isto é, ele não faz parte de um instanciamento de um objeto!

```java
public class TesteFuncao01 {

    /**
     * @param args the command line arguments
     */

    static void soma (int a, int b) {
        int s = a + b;
        System.out.println("A soma é " + s);
    }

    public static void main(String[] args) {
        // TODO code application logic here
        soma (5,2);
    }

}
```

```
run:
A soma é 7
```

Funções em algoritmos:



Tipo de retorno

- Parece muito com o procedimento, mas ele tem o **tipo de retorno**

Como eu leio isso?

- Eu tenho uma função Soma, que vai receber dois parâmetros (A e B, do tipo inteiro) e que vai retornar um valor inteiro.

- Em vez de escrever o número na tela, eu tenho o comando **retorne**, que vai retornar um valor inteiro (que foi declarado previamente)

Para chamar uma função, existem várias maneiras

- Atribuindo esse valor de retorno a uma variável

```
sm <- Soma(5,2)
```

- Criei uma variável sm que vai receber o resultado da soma entre 5 e 2

Funções em Java:

```
int soma (int a,int b){
    int s = a + b;
    return s;
}

int sm = soma(5,2);
```

No NetBeans:

```
12  public class TesteFuncao01 {
13
14      /**
15       * @param args the command line arguments
16       */
17
18      static int soma (int a, int b) {
19          int s = a + b;
20          return s;
21      }
22
23      public static void main(String[] args) {
24          // TODO code application logic here
25          int sm = soma (5,2);
26          System.out.println("A soma vale " + sm);
27      }
28
29  }
```

```
run:
A soma vale 7
```

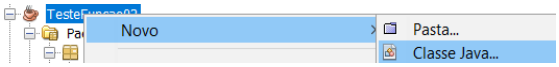
OBS: se eu tirar o **static** o programa vai dar erro!

- Ele não é obrigatório, porém estamos chamando a função soma dentro dentro do main que é estático

- Não posso chamar um método não estático dentro de um método estático!

Exemplos com múltiplas classes:

1º - após criar o projeto, criar uma nova classe



Para criar o public static void main: **psvm + tab**

TesteFuncao02.java × Operacoes.java ×

- Agora eu tenho uma aba para o meu projeto e outra para a minha classe

- O ideal é que eu torne o meu método criado público, ou seja, ele pode ser acessado por qualquer pessoa!

- Eu também tenho a opção de tornar privado (private) ou protege-la (protected)

```
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("Vai começar a contagem");
        System.out.println(Operacoes.contador(1,5));
    }

}
```

```
public class Operacoes {

    public static String contador (int i, int f) {
        String s = "";
        for (int c = i; c <= f; c++){
            s += c + "";
        }
        return s;
    }

}
```