Modularização ou Subrotinas

Parte I

Objetivos da Aula

- Entender os benefícios no uso dos módulos ou **subrotinas**
- Entender conceitos de módulos
- Entender (relembrar) como utilizar módulos já existentes

Refletindo ...

Até o momento temos escrito nossos programas no PortugolStudio de maneira “sequencial” dentro da **função início**

```
1  programa
2  {
3      funcao inicio()
4      {
5
6          // DIGITE AQUI O SEU PROGRAMA
7
8      }
9  }
```

Refletindo ...

Mas aprendemos também que além dos comandos da linguagem (leia, escreva, se, enquanto) podemos **CHAMAR** (utilizar) rotinas (funções) prontas na linguagem

```
inclua biblioteca Matematica -->mat

funcao inicio() {
    real base, expoente, resultado
    escreva("Digite a base: ")
    leia(base)
    escreva("Digite o expoente: ")
    leia(expoente)

    // chamando a função que calcula a potencia
    // a função já existe na biblioteca Matematica
    resultado = mat.potencia(base, expoente)

    escreva(base, " elevado a ", expoente, " = ", resultado)
}
```

Refletindo.

- Note que, se não tivéssemos a função potencia pronta (na biblioteca Matematica) e precisássemos calcular a potência (x elevado a y) teríamos que construir esse código (programar).
- Mas como já existe (e nós conhecemos seu nome) nós apenas a chamamos.

```
resultado = mat.potencia(base, expoente)
```

```
resultado = Math.pow( base, expoente);
```

potencia é um exemplo de uma função (módulo ou subrotina) já existente no Portugol Studio

Módulos - conceituando

Existem outras funções “prontas”. Por exemplo:

- **double Math.abs(x):** Valor absoluto positivo
- **double Math.cos(x):** Valor do cosseno de X
- **double Math.round(x):** x arredondado
- **double Math.random ():** Valor aleatório (0-1)
- **double Math.toRadians(t):** t em radianos
- **double Math.min(a,b):** Valor mínimo entre a e b
- Etc.

Módulos - conceituando

Veremos mais adiante que nós também podemos construir nossas próprias funções (módulos ou subrotinas). Isso chama-se **MODULARIZAÇÃO!**

- A modularização de um programa é a divisão desse programa em subprogramas

Módulos – Parte 1

```
programa
{
    funcao inicio()
    {
        //codigo da funcao principal
    }

    funcao moduloA() {
        //código do módulo A
    }

    funcao moduloB() {
        //código do módulo B
    }
}
```

Modularização

o que é?

decomposição da solução de um problema computacional em módulos (ou subalgoritmos) para dominar a complexidade e organizar o processo de resolução;

Módulo

é um trecho de algoritmo com uma função bem definida e independente do resto do algoritmo.

o que se ganha?

- confiabilidade;
- legibilidade;
- manutenibilidade;
- flexibilidade;

Módulos / Funões / Métodos ?

À medida que os problemas vão se tornando mais complexos, **os programas (solução) tendem a ficar mais extensos.**

Modularizar permite “quebrar” o problema em subproblemas.

Módulos – Vantagens

- Neste ponto podemos entender **uma das vantagens** da utilização do módulo: **reuso!**
  - Imagine a economia de código que se faz ao chamar o a função RAIZ sem necessitar implementá-la?
  - O mesmo vale para a função **SORTEIA.**
  - E não é nem necessário CONHECER o código. Basta saber o **NOME da função, o que ela faz e sua sintaxe (parâmetros, etc)**

Bibliotecas

- A maioria das **linguagens de programação** já vem com um grupo de **funções** que facilitam a vida do programador.
- Estas funções realizam diversas tarefas, tais como:
  - cálculos aritméticos e trigonométricos
  - manipulação e conversão de dados

- Geralmente, essas funções são muito usadas em diversos programas e por isso mesmo já são disponibilizadas pelas linguagens de programação
  - Assim, o programador não tem que reinventar a roda a cada programa que faz.

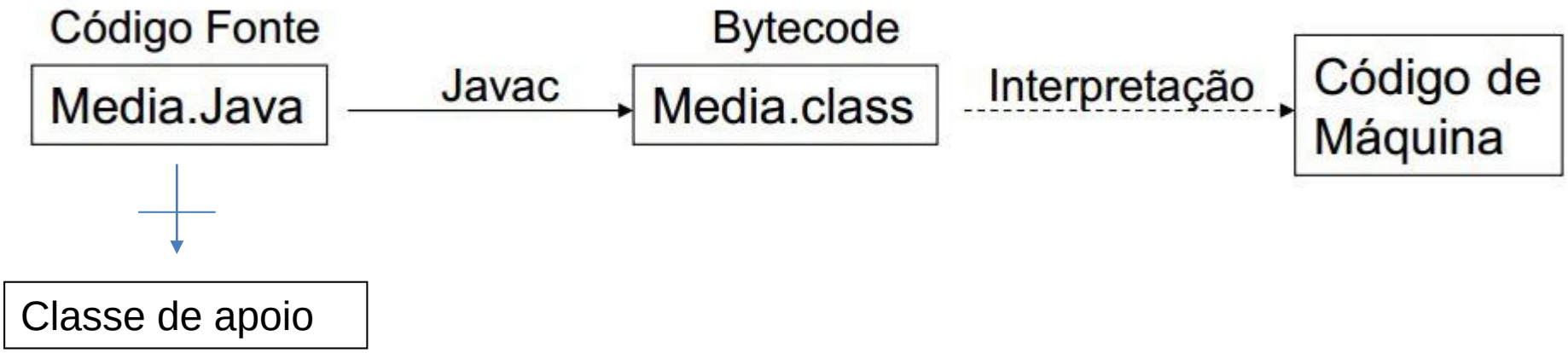
- A este grupo de funções dá-se às vezes o nome de **biblioteca.**

Módulos – Mais Vantagens

- Reaproveitamento de código: diminuição do código, diminuição do retrabalho (se tivéssemos que implementar de novo).
- Organização
- Legibilidade
- Flexibilidade

Essas vantagens ficarão mais claras quando começarmos a construir nossas próprias funções

Bytecode



Ex:
Scanner class
System class
Math class
JOptionPane class
Etc.

```
    // imprime matriz no formato
    public void imprime() {
        for (int i = 0; i < M; i
            System.out.print("|
            for (int j = 0; j <
                System.out.print
            System.out.println(
        }
    }
    // imprime matriz no formato
    public static Matriz leia(i
        Matriz A = new Matriz(M
        java.util.Scanner scann
                          scanner.
        for(int i= 0; i < M; i++
            for (int j = 0; j <
                System.out.print
                        i+1, j+
                A.data[i][j] =
            }
        }

        return A;
    }

    // cria e retorna uma matri
    public static Matriz aleato
```

Scans the next token of the input as an int.

An invocation of this method of the form nextInt() behaves in exactly the same way as the invocation nextInt(radix), where radix is the default radix of this scanner.

Returns:

the int scanned from the input

Throws:

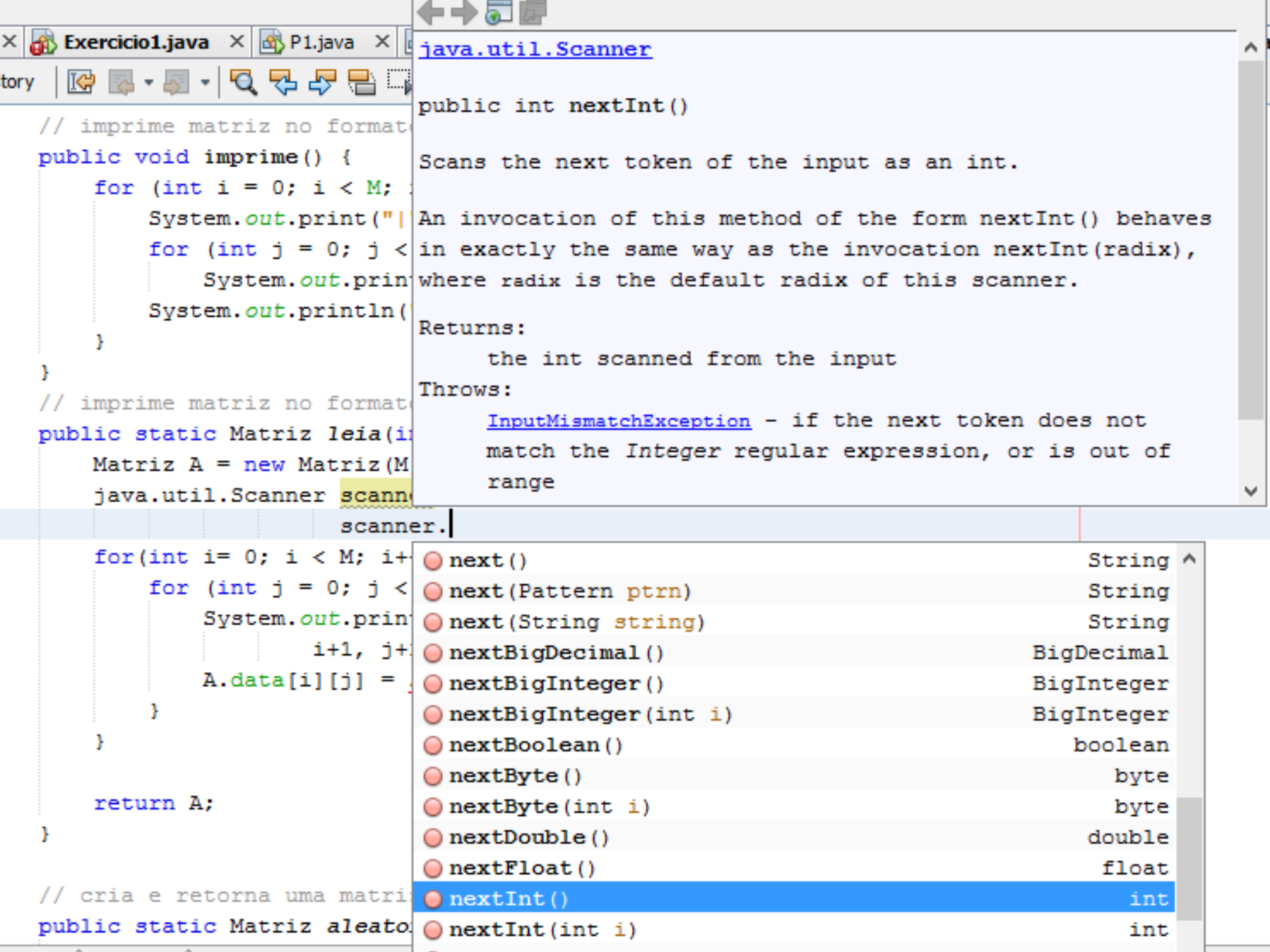
InputMismatchException - if the next token does not match the *Integer* regular expression, or is out of range

next()	String
next(Pattern ptrn)	String
next(String string)	String
nextBigDecimal()	BigDecimal
nextBigInteger()	BigInteger
nextBigInteger(int i)	BigInteger
nextBoolean()	boolean
nextByte()	byte
nextByte(int i)	byte
nextDouble()	double
nextFloat()	float
nextInt()	int
nextInt(int i)	int
nextLine()	String
nextLong()	long
nextLong(int i)	long
nextShort()	short

ercicio1.Matriz > leia >

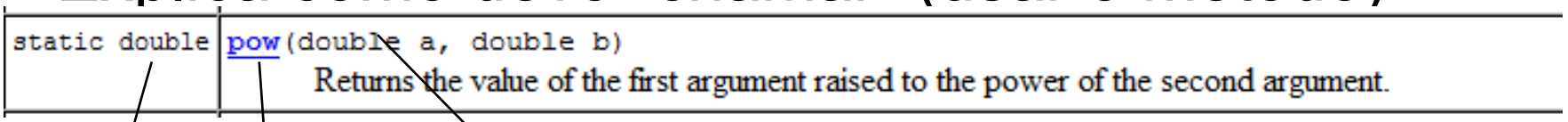
s | Output - Exercicio1 (run) ×

run:


Exercicio1.java
P1.java
// imprime matriz no formato
public void imprime() {
for (int i = 0; i < M;
System.out.print("|
for (int j = 0; j <
System.out.print
System.out.println(
}
}
// imprime matriz no formato
public static Matriz leia(i
Matriz A = new Matriz(M
java.util.Scanner scann
scanner.
for(int i= 0; i < M; i+
for (int j = 0; j <
System.out.print
i+1, j+
A.data[i][j] =
}
}
return A;
}
// cria e retorna uma matri
public static Matriz aleato
java.util.Scanner
public int nextInt()
Scans the next token of the input as an int.
An invocation of this method of the form nextInt() behaves in exactly the same way as the invocation nextInt(radix), where radix is the default radix of this scanner.
Returns:
the int scanned from the input
Throws:
InputMismatchException - if the next token does not match the Integer regular expression, or is out of range
next() String
next(Pattern ptrn) String
next(String string) String
nextBigDecimal() BigDecimal
nextBigInteger() BigInteger
nextBigInteger(int i) BigInteger
nextBoolean() boolean
nextByte() byte
nextByte(int i) byte
nextDouble() double
nextFloat() float
nextInt() int
nextInt(int i) int

Assinatura do método

- Explica como devo “chamar” (usar o método)



Valor de retorno

Nome

- Número de parâmetros
- Tipo de cada parâmetro

```
System.out.println("O resultado é: "+Math.pow(base, expoente));
```

Assinatura do método

- Assinatura indica o que posso fazer e o que não posso (veja exemplos abaixo):

```
Math.pow(2, 0.5);
```
Correto?

```
double pot = Math.pow(2,0.5);
```
correto

```
Math.pow(0.5);
```
incorreto

```
Math.pow("2","2.5");
```
incorreto

Explique porque está correto ou incorreto.

Módulos – Como utilizar?

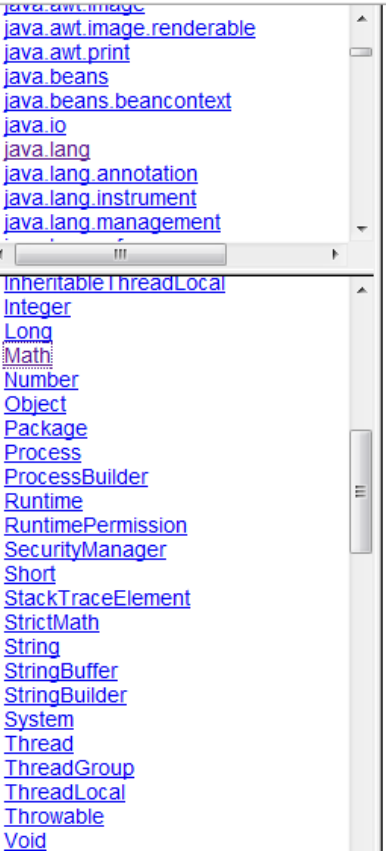
- Alguns métodos em JAVA podem ser usados de várias maneiras. Quem "*manda*" é a *assinatura* dele! Veja o método abs:

`static double`	`abs(double a)` Returns the absolute value of a `double` value.
`static float`	`abs(float a)` Returns the absolute value of a `float` value.
`static int`	`abs(int a)` Returns the absolute value of an `int` value.
`static long`	`abs(long a)` Returns the absolute value of a `long` value.

Esse é um dos exemplos de polimorfismo em Programação Orientada a Objetos

Módulos – Como utilizar?

No Java consulte a API: http://download.oracle.com/javase/6/docs/api/



Method Summary

`static double`	**`abs`**`(double a)` Returns the absolute value of a `double` value.
`static float`	**`abs`**`(float a)` Returns the absolute value of a `float` value.
`static int`	**`abs`**`(int a)` Returns the absolute value of an `int` value.
`static long`	**`abs`**`(long a)` Returns the absolute value of a `long` value.
`static double`	**`acos`**`(double a)` Returns the arc cosine of a value; the returned angle is in the range 0.0 through *pi*.
`static double`	**`asin`**`(double a)` Returns the arc sine of a value; the returned angle is in the range -*pi*/2 through *pi*/2.
`static double`	**`atan`**`(double a)` Returns the arc tangent of a value; the returned angle is in the range -*pi*/2 through *pi*/2.
`static double`	**`atan2`**`(double y, double x)` Returns the angle *theta* from the conversion of rectangular coordinates (`x`, `y`) to polar coordinates (`r`, *theta*).
`static double`	**`cbrt`**`(double a)` Returns the cube root of a `double` value.
`static double`	**`ceil`**`(double a)` Returns the smallest (closest to negative infinity) `double` value that is greater than or equal to the argument and is equal to a mathematical integer.
`static double`	**`copySign`**`(double magnitude, double sign)` Returns the first floating-point argument with the sign of the second floating-point argument.

COMPONENTES DE UM MÓDULO

Módulos - Componentes

- **(1) Identificador** (nome da função)
- **(2) Parâmetros** (o que a função recebe para para processar)
- **(3) retorno** (o que a função retorna)

(2) PARÂMETROS

```
resultado = mat.potencia(base, expoente)
```

(3) RETORNO

(1) IDENTIFICADOR

PARÂMETROS

- São os valores ou variáveis **enviadas ao módulo**, para que ele possa realizar o seu processamento
- Podemos passar nenhum ou vários parâmetros para o módulo
  - Tudo depende da finalidade do módulo (e de como ele foi construído)

(2) PARÂMETROS

```
resultado = mat.potencia(base, expoente)
```

PortugolStudio

Módulos – Componentes - Retorno

Uma variável recebe o valor de retorno do módulo

```
1  programa
2  {
3      inclua biblioteca Matematica -->mat
4
5      funcao inicio()
6      {
           real base, expoente, resultado
           escreva("Digite a base: ")
           leia(base)
           escreva("Digite o expoente: ")
           leia(expoente)

           resultado = mat.potencia(base, expoente)

14
15         escreva(base, " elevado a ", expoente, " = ", resultado)
16     }
17 }
```

INFORMAÇÕES ADICINAIS

- O Retorno de um módulo é o que define a classificação módulo: **PROCEDIMENTO ou FUNÇÃO**
- Módulos que não retornam valores são denominados PROCEDIMENTOS
- Módulos que retornam valores são denominados FUNÇÕES

NESSE CURSO VAMOS CHAMAR TUDO DE FUNÇÃO!

Módulos - Nomenclatura

- Em Portugol Studio
  - Os módulos são chamados de funções
- Em Java
  - Os módulos são chamados de MÉTODOS
- **Usaremos como sinônimos: chamaremos tudo de FUNÇÕES!**

Módulos – Formas de Utilização

```
class ExemploModulosJava {

   public static void main(String args[]) {

      System.out.println("Metodo abs(-30): "     + Math.abs(-30) );
      System.out.println("Metodo pow(2,3): "     + Math.pow(2,3) );
      System.out.println("Metodo sqrt(16): "     + Math.sqrt(16) );

   }
}
```

No java temos a biblioteca Math

abs – retorna o valor absoluto do número
pow – retorna a potência do número
sqrt – retonra a raiz quadrada no número

No JAVA indica-se também o nome da biblioteca que contém o módulo (função)

Módulos – Componentes - Retorno

```
class ExemploModulosJava {

   public static void main(String args[]) {

      System.out.println("Metodo abs(-30): "      + Math.abs(-30) );
      System.out.println("Metodo pow(2,3): "      + Math.pow(2,3) );
      System.out.println("Metodo sqrt(16): "      + Math.sqrt(16) );

   }
}
```

Só é possível imprimir o valor do módulo, potência e quadrado, porque os módulos *abs, pow e srqrt respectivamente* retornam valores

Todos os valores retornados por *abs, pow e srqrt poderiam ter sido atribuídos a uma variável*

Módulos – Componentes – Resumindo...

INDEPENDENTE DA LINGUAGEM: PortugolStudio OU Java

Math.sqrt(9)

mat.raiz(9, 2)

Resolvem o mesmo problema
O identificador é mnemônico
Retornam os valores 3
Possuem um parâmetro

limpa()

Não retorna valor
Não possui parâmetro

TODOS SÃO MÓDULOS!

Módulos – Componentes – Resumindo...

INDEPENDENTE DA LINGUAGEM: PortugolStudio OU Java

Math.sqrt(9)

mat.raiz(9,2)

São FUNÇÕES

Limpa()

É PROCEDIMENTO

TODOS SÃO MÓDULOS!

Assinatura

A assinatura da função define

- O tipo de acesso
- O que ela recebe (entrada) = parâmetros
- O que ela retorna (saída) = int, double, tipo

```
static int caixaPreta(int a, int b) {
    return a*a + b;
}
```

Chamada de função

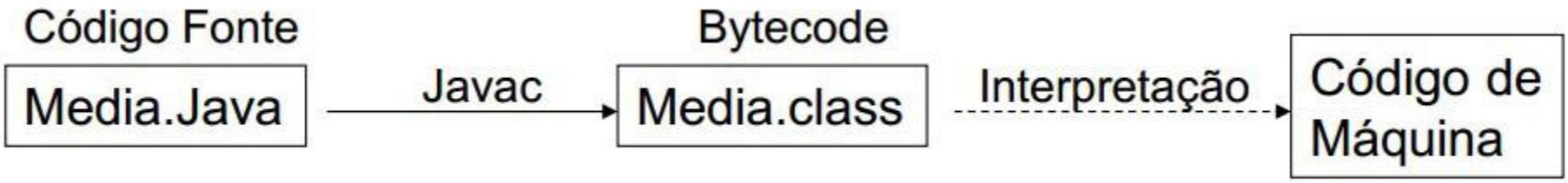
```
1  public class Funcao01
2  {
3      public static void main(String []args) {
4
5          int resposta;
6
7          resposta = caixaPreta(3, 5);
8
9          System.out.println(resposta);
10     }
11 }
```

A ordem dos módulos na classe não importa

```
1  public class Funcao01
2  {
3      static int caixaPreta(int a, int b) {
4          return a*a + b;
5      }
6
7      public static void main(String []args) {
8
9          int resposta;
10
11         resposta = caixaPreta(3, 5);
12
13         System.out.println(resposta);
14     }
15 }
```

Compilação

```
sh-4.2# javac Funcao01.java
sh-4.2# java Funcao01
14
```



```
1  public class Funcao02
2  {
3      static double calcular(double a, double b) {
4          double x;
5
6          x = a*a + 2*a*b + b*b;
7
8          return x;
9      }
10
11     public static void main(String []args) {
12
13         double resposta;
14
15         resposta = calcular(2, 3);
16
17         System.out.println(resposta);
18     }
19 }
```

```java
import java.math.*;

public class Funcao03
{
    static double calcular(double a, double b) {
        double x;

        x = Math.pow(a,2) + 2*a*b + Math.pow(b,2);

        return x;
    }

    public static void main(String []args) {

        double resposta;

        resposta = calcular(2, 3);

        System.out.println(resposta);
    }
}
```

```java
import java.math.*;

public class Cilindro
{
    static double calcularVolume(double r, double h) {
        return Math.PI * r*r * h;
    }

    public static void main(String []args) {
        double r, h;

        r = 2;
        h = 10;

        System.out.println( calcularVolume(r, h) );
    }
}
```

COISAS PARA NÃO ESQUECER

Coisas para não esquecer

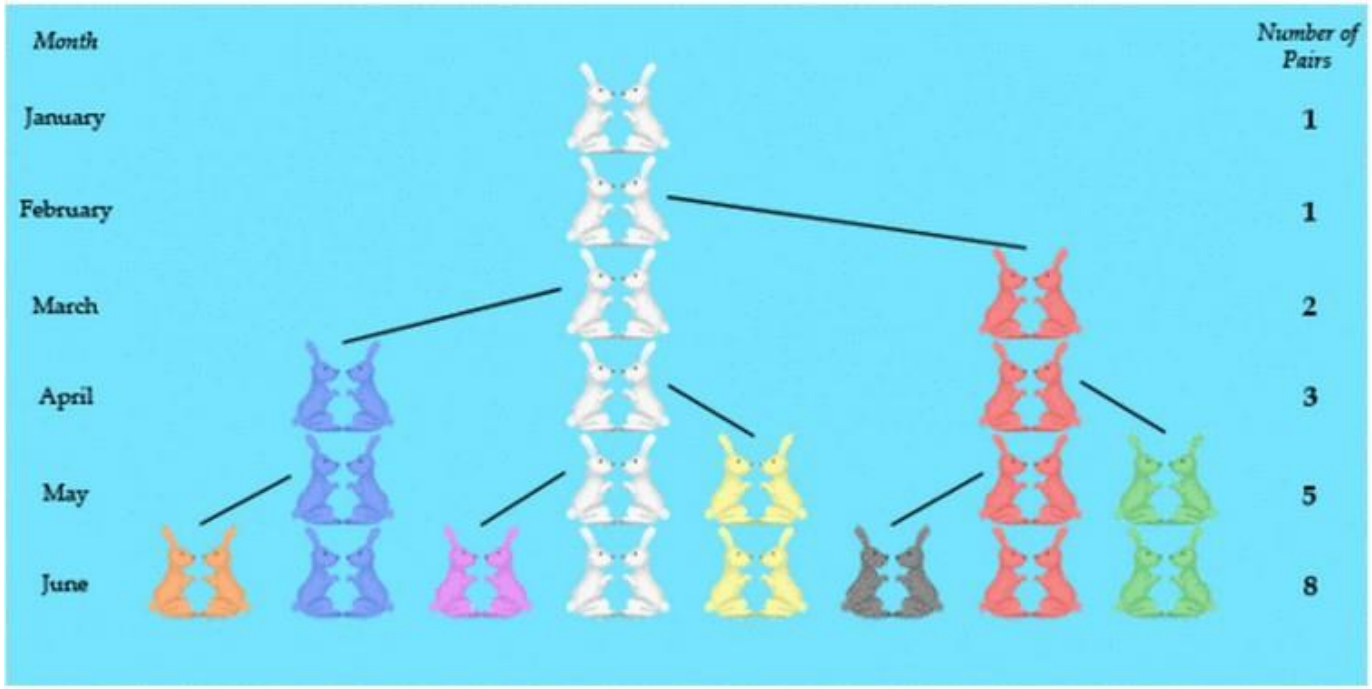
- Módulos são *subprogramas* que podemos chamar dentro do programa principal ou dentro de outros módulos
- Um módulo DEVE possuir: um identificador ou nome de chamada
- Um módulo PODE possuir: parâmetros e/ou valor de retorno
- Um módulo PODE ou não retornar valores

Coisas para não esquecer

- Um módulo que Não possui valor de retorno é chamado de PROCEDIMENTO
- Um módulo que possui valor de retorno é chamado de FUNÇÃO

O PRÓXIMO PASSO É APRENDER A CRIAR OS NOSSO PRÓPRIOS MÓDULOS (FUNÇÕES)

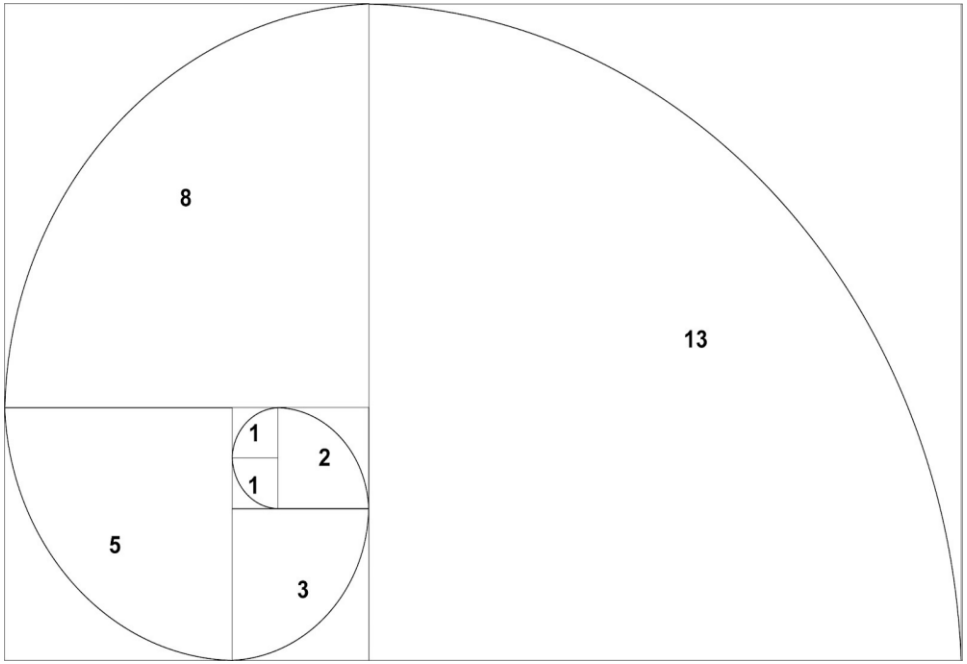
Os números de Fibonacci foram propostos por Leonardo di Pisa (Fibonacci), em 1202, como uma solução para o problema de determinaro tamanho da população de coelhos



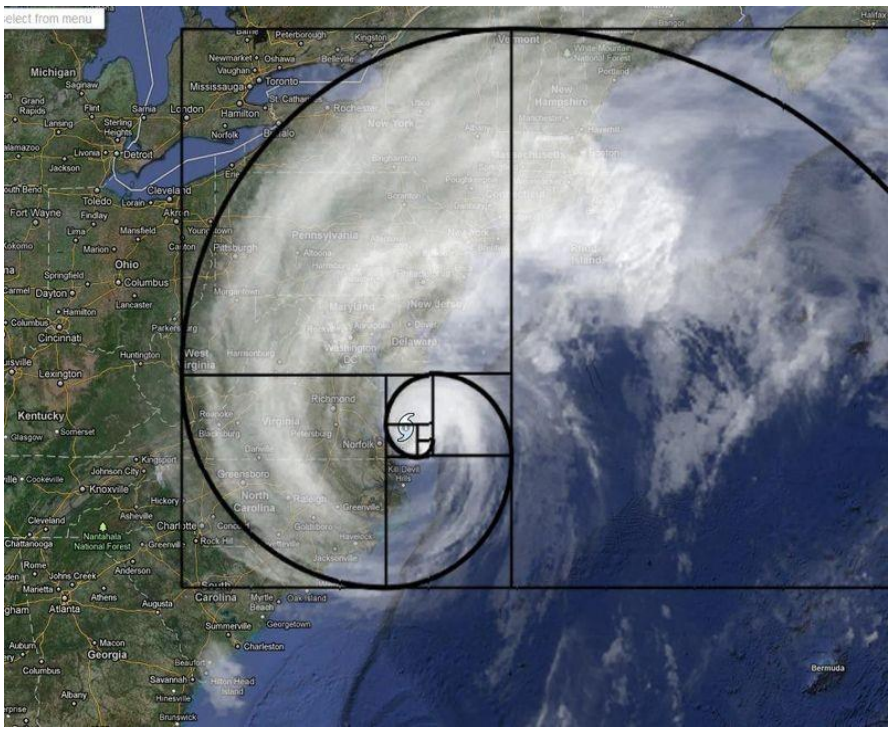
(*) fonte http://www.oxfordmathcenter.com/drupal7/node/487

Série de Fibonacci

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181	6765

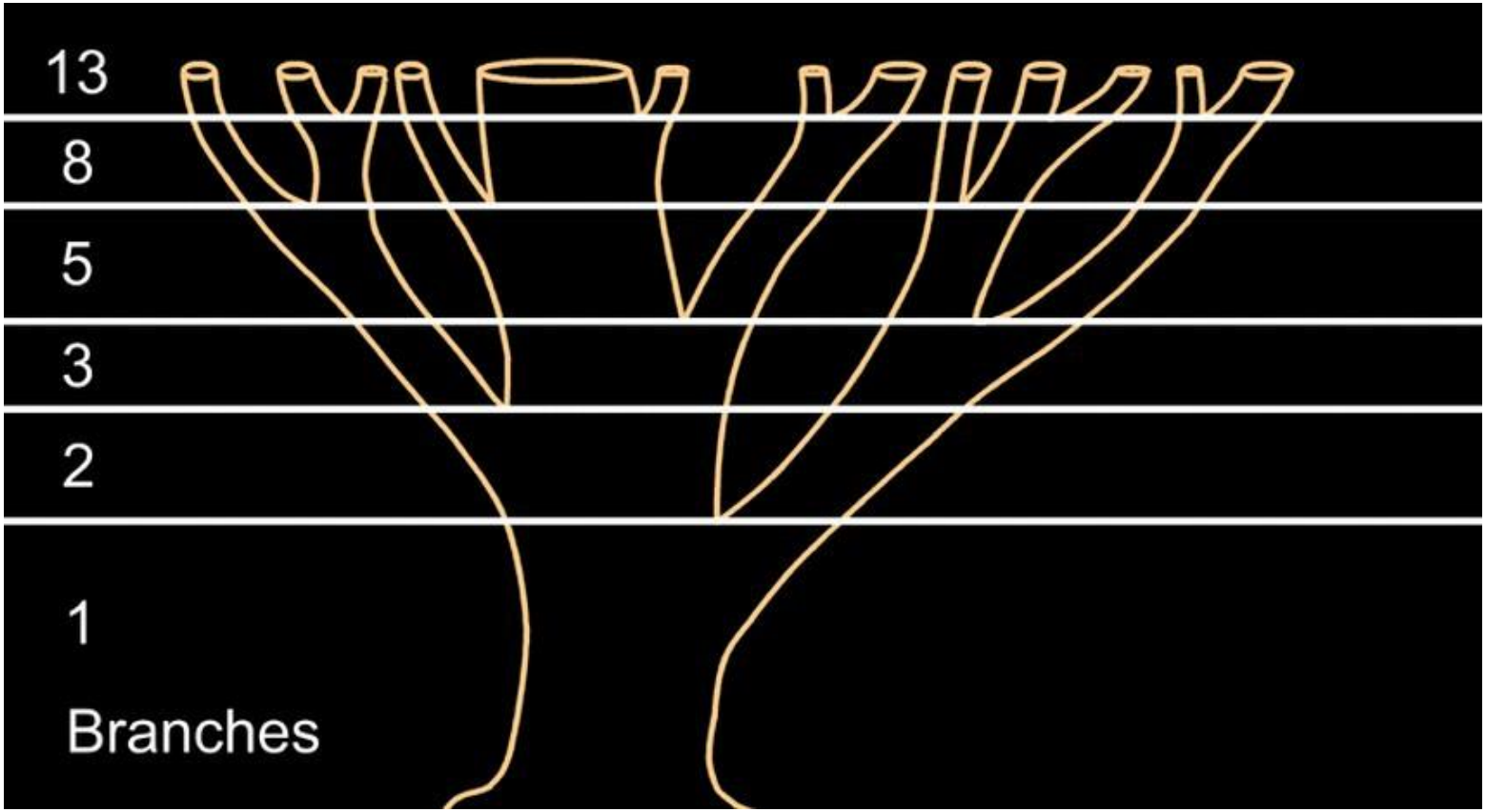


Série de Fibonacci



https://www.youtube.com/watch?v=SjSHVDfXHQ4

Série de Fibonacci



(*) fonte http://britton.disted.camosun.bc.ca/fibslide/jbfibslide.htm

Série de Fibonacci

Os números de Fibonacci estão relacionados com a razão aurea e o i-ésimo número pode ser aproximado pela seguinte equação (formula explícita):

$$F_i = \left\lfloor \frac{\left(\frac{1+\sqrt{5}}{2}\right)^i - \left(\frac{1-\sqrt{5}}{2}\right)^i}{\sqrt{5}} \right\rfloor$$

Crie uma função/método em Java que receba um número inteiro i, e devolva Fi.

Assinatura:

```
static double iessimoTermo(int i)
```

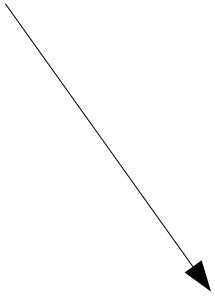
Série de Fibonacci

```
static double iessimoTermo(int i) {

    double var1, var2, var3;

    var1 = Math.pow( (1+Math.sqrt(5))/2, i);
    var2 = Math.pow( (1-Math.sqrt(5))/2, i);
    var3 = Math.sqrt(5);

    return Math.floor((var1-var2)/var3);
}
```

```java
import java.math.*;

public class Fibonacci
{
    static double iessimoTermo(int i) {

        double var1, var2, var3;

        var1 = Math.pow( (1+Math.sqrt(5))/2, i);
        var2 = Math.pow( (1-Math.sqrt(5))/2, i);
        var3 = Math.sqrt(5);

        return Math.floor((var1-var2)/var3);
    }

    public static void main(String []args) {

        System.out.println( iessimoTermo(10) );
        System.out.println( iessimoTermo(11) );
        System.out.println( iessimoTermo(12) );
     }
}
```

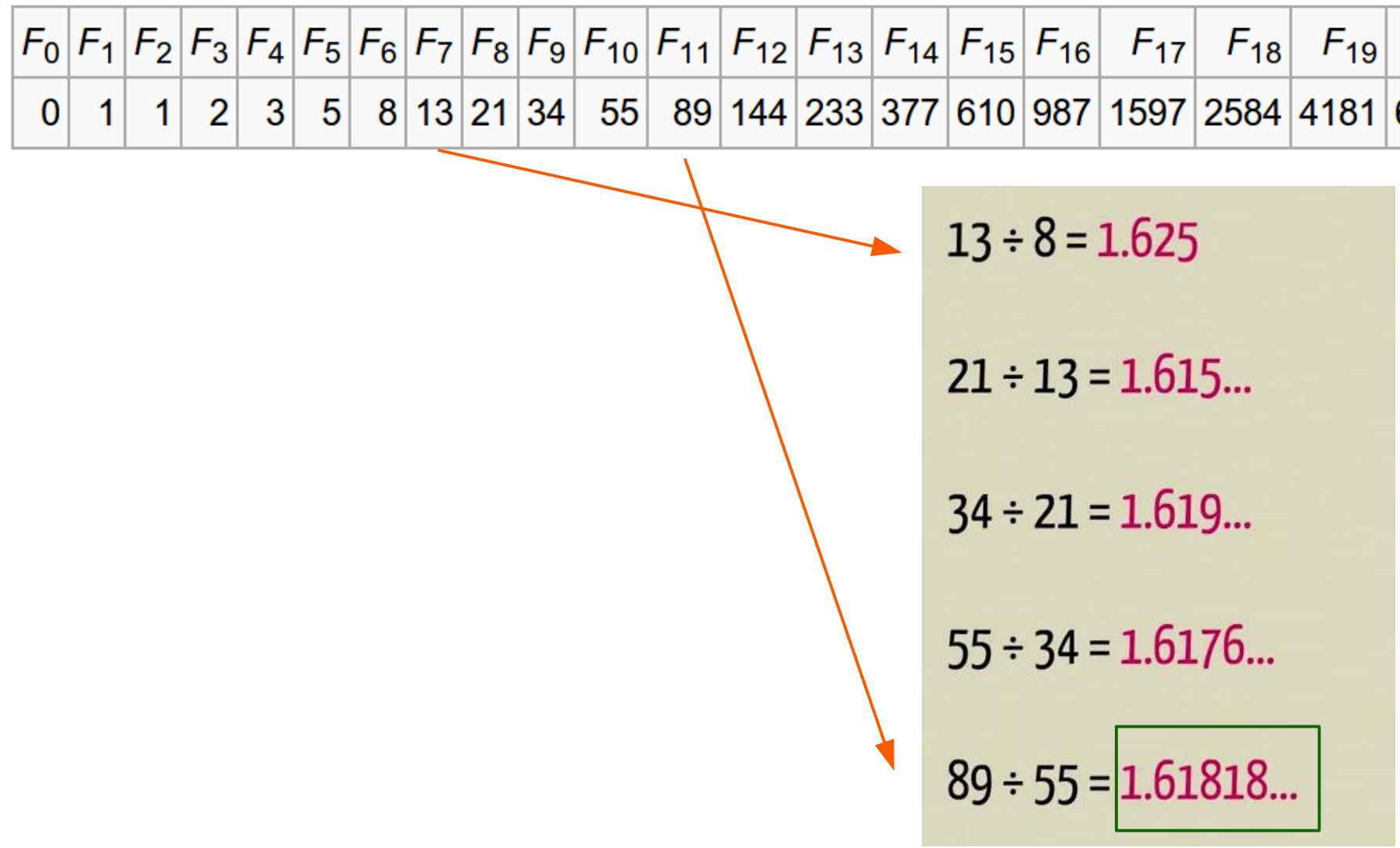
Série de Fibonacci

```
sh-4.2# javac Fibonacci.java
sh-4.2# java Fibonacci
55.0
89.0
144.0
```

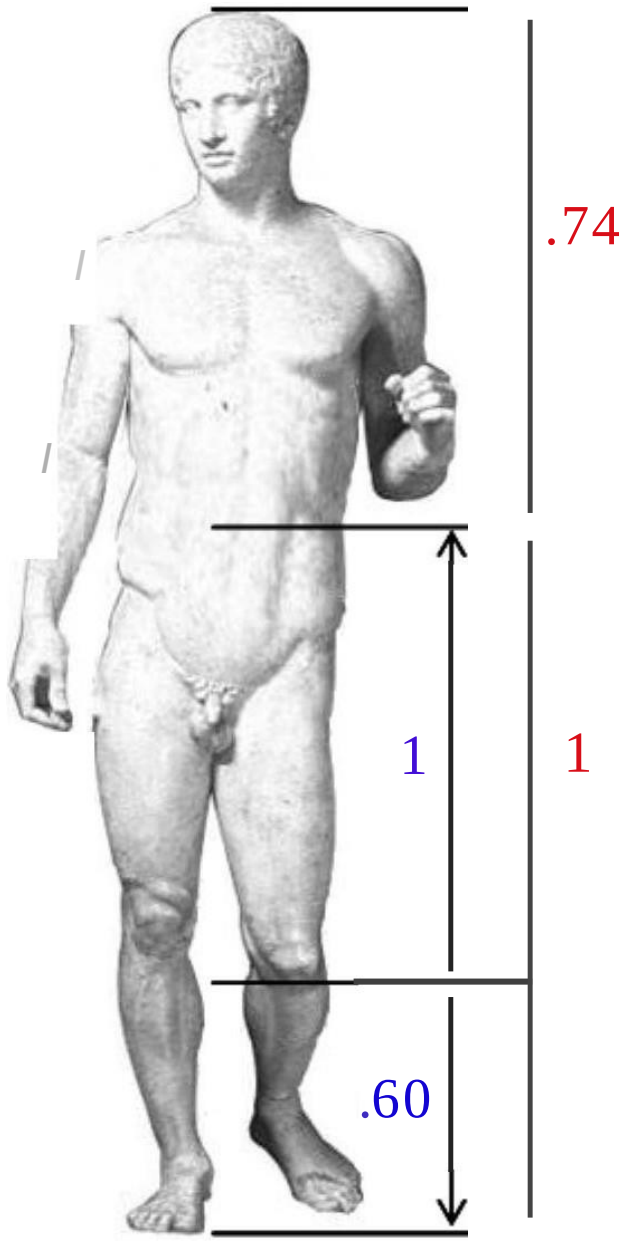


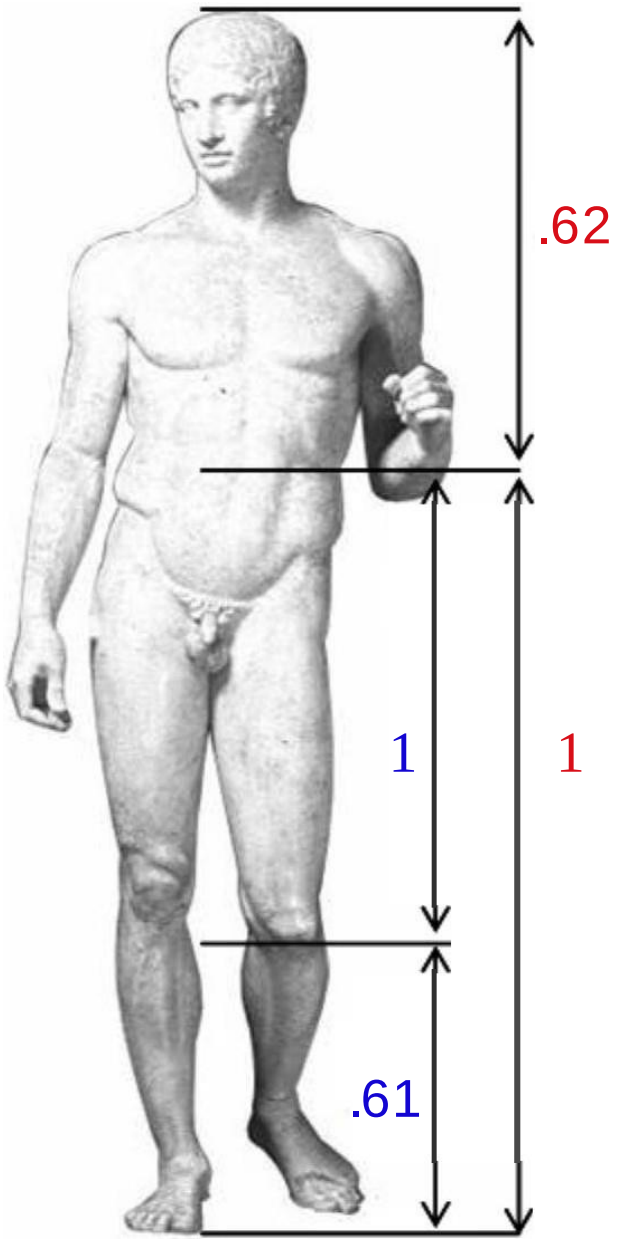
F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181	6765

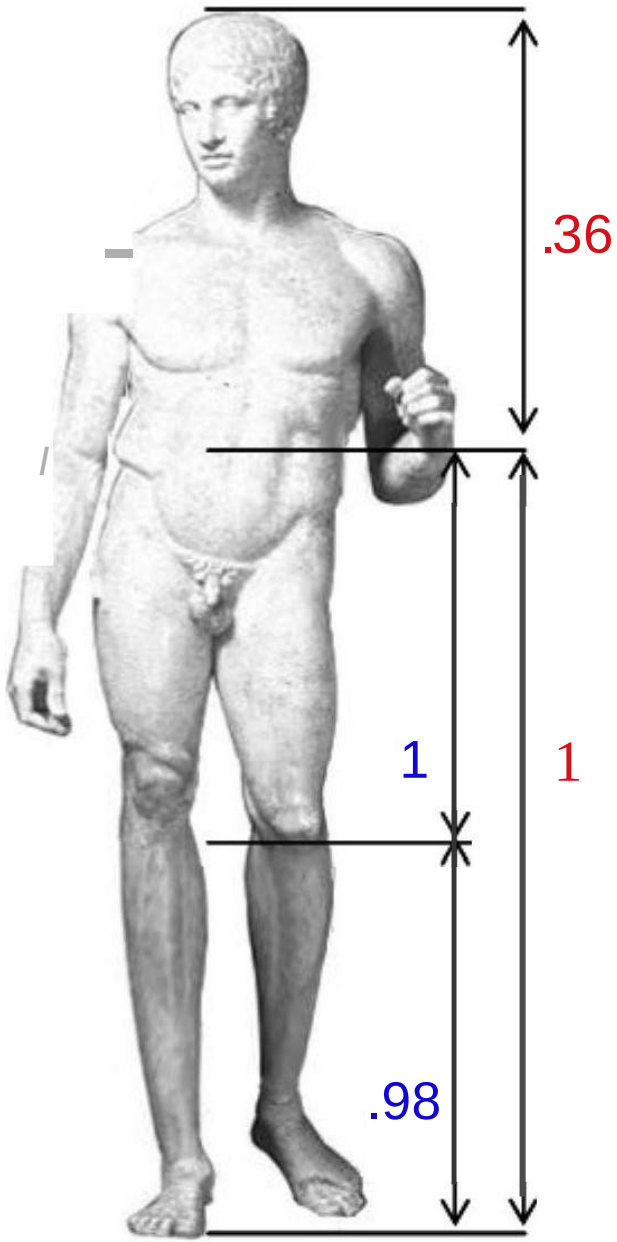
F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181	6765



Golden ratio


.74
1
1
.60

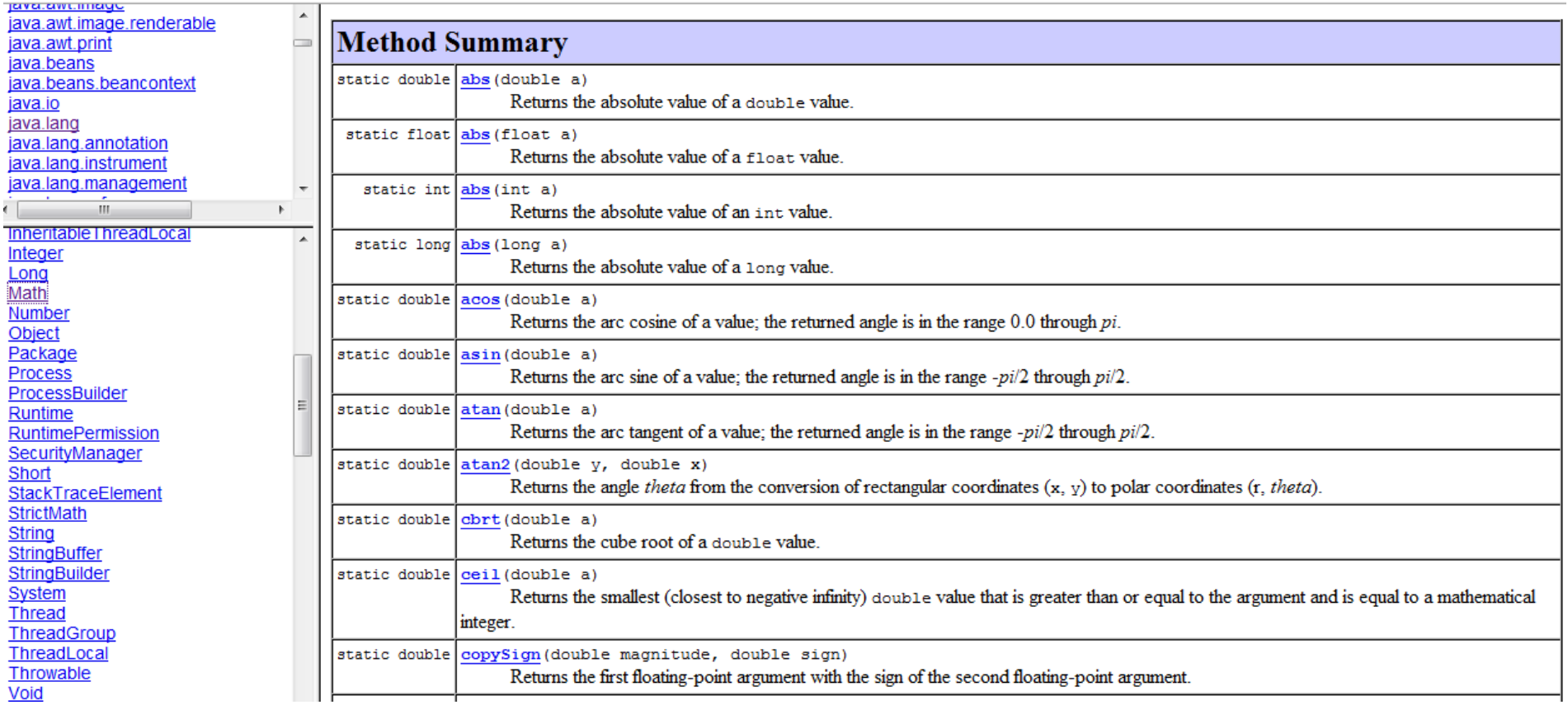

.62
1
1
.61


.36
1
1
.98

Modularização ou Subrotinas

Parte II

JAVADOCS



Métodos da classe Math

Acesso

- No Java, as funções são distribuídas em classes que lidam com classes distintas de problemas
- Métodos de classes do JAVA seguem o paradigma de Linguagens Orientadas a Objetos
- Em tal paradigma, Objetos de uma determinada classe podem acessar outros métodos e variáveis em outras classes, se os mesmos estiverem declarados como "**Public**"
- As declarações possíveis são "Public", "Private" e "Protected"

Declaração de classes e métodos

Publico/Privado → Controle de acesso por outras classes

static → variável/método não-instanciado (global para todos os objetos da classe)

```
public class Exemplo1 {
        public static double Pi= 3.14159;
        public static void main() {}
        public int calculaRaiz(x, y) {}
}
```

Passagem de parâmetros

Passagem de parâmetros é um mecanismo pelo qual se estabelece uma comunicação bidirecional entre um módulos;

dados locais a um módulo podem ser enviados para um outro módulo, que por sua vez poderá utilizar estes dados e alterá-los ou não;

estabelece associação entre parâmetros atuais e formais e pode ser de dois tipos:

ALGORITMO – PASSOS PARA CONSTRUÇÃO DE MÉTODOS

Saída:
Entrada:
Processamento:

ALGORITMO – PASSOS PARA CONSTRUÇÃO DE MÉTODOS

Saída: Valor retornado ao ponto de chamada

Entrada: Parâmetros passados + variáveis lidas no método

Processamento: Como chegar das entradas nas saídas?

ALGORITMO – PASSOS PARA CONSTRUÇÃO DE MÉTODOS

Saída: Valor retornado ao ponto de chamada

Entrada: Parâmetros passados + variáveis lidas no método

Processamento: Como chegar das entradas nas saídas?

```
public static int calcula_idade (int ano_de_nascimento, int ano_atual) {
        int idade = ano_atual - ano_de_nascimento;
        return idade;
}
```

ALGORITMO – PASSOS PARA CONSTRUÇÃO DE MÉTODOS

Saída: Variável retornada ao ponto de chamada

Entrada: Parâmetros passados + variáveis lidas no método

Processamento: Como chegar das entradas nas saídas?

Saída

```
public static int calcula_idade (int ano_de_nascimento, int ano_atual) {
        int idade = ano_atual - ano_de_nascimento;
        return idade;
}
```

ALGORITMO – PASSOS PARA CONSTRUÇÃO DE MÉTODOS

Saída: Variável retornada ao ponto de chamada

Entrada: Parâmetros passados + variáveis lidas no método

Processamento: Como chegar das entradas nas saídas?

Entrada

```
public static int calcula_idade (int ano_de_nascimento, int ano_atual) {
        int idade = ano_atual - ano_de_nascimento;
        return idade;
}
```

ALGORITMO – PASSOS PARA CONSTRUÇÃO DE MÉTODOS

Saída: Variável retornada ao ponto de chamada

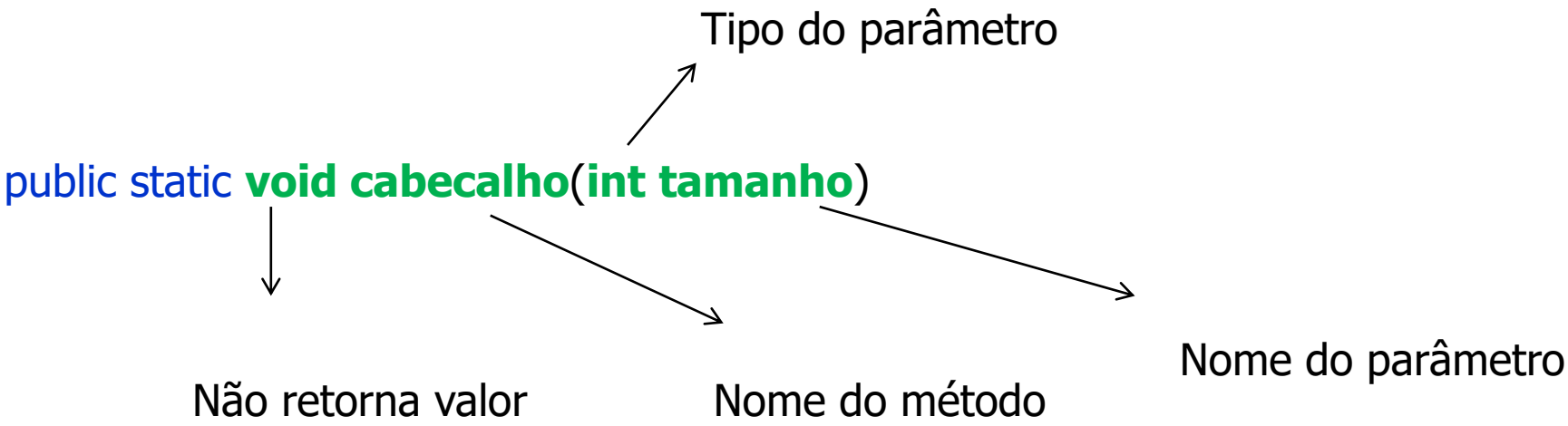
Entrada: Parâmetros passados + variáveis lidas no método

Processamento: Como chegar das entradas nas saídas?

```
public static int calcula_idade (int ano_de_nascimento, int ano_atual) {
        int idade = ano_atual - ano_de_nascimento;
        return idade;
}
```

Processamento

SINTAXE DO EXEMPLO



- Os valores que estão entre **parênteses**, representados pelas palavras como *expressão*, *base* e *expoente*, são os **parâmetros**, ou como dizem alguns autores, os **argumentos** que passamos para a função para que realize seus cálculos e retorne um valor, que usaremos no programa.
- Algumas funções, como **Rand**, não precisam de parâmetros, mas a maioria tem um ou mais. Constantes e variáveis, como **Pi** não tem "()"
- O valor dos parâmetros naturalmente altera o valor retornado pela função.

```
int x = Integer.parseInt("21");
```

pow é um método Pré-definido que você já usou!

```
package aula3;
import java.util.*;
import java.lang.Math;
public class Main {
    public static void main(String[] args) {
        // TODO code application logic here
        Scanner in = new Scanner(System.in);
        double pot;
        double base, expoente;
        System.out.println("Digite a base ");
        System.out.println("Digite o expoente ");
        base = in.nextDouble();
        expoente = in.nextInt();
        System.out.println("O resultado é: "+Math.pow(base, expoente));
    }
}
```

Indica a classe a que o método pertence: Math

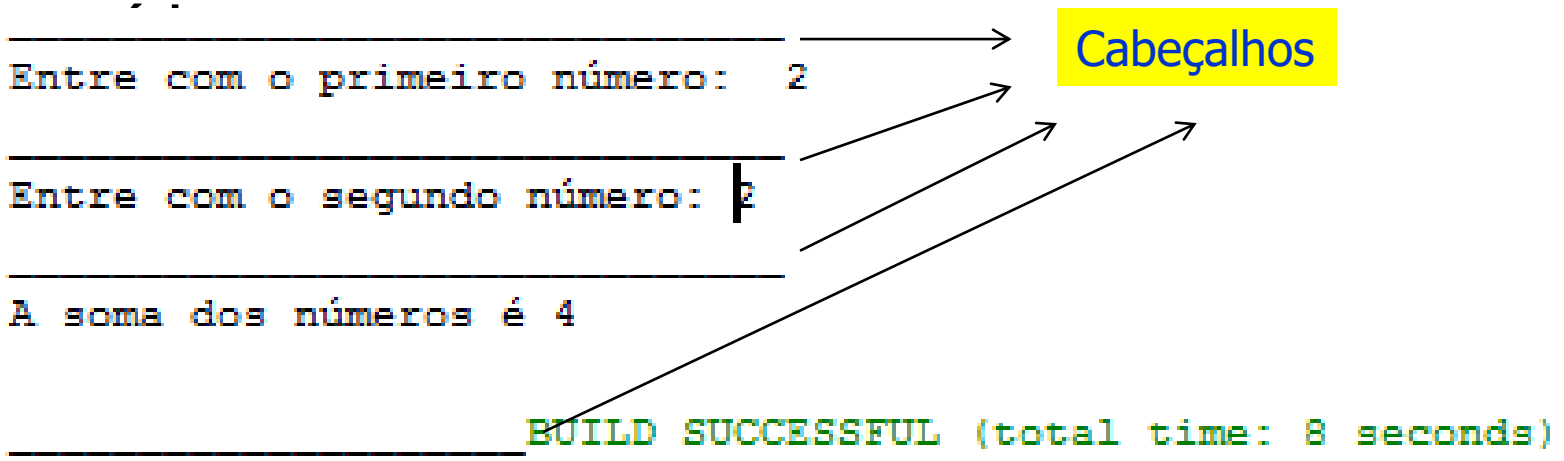
É a chamada do méto

Para chamar um método dentro do seu programa você precisa Saber a classe que ele pertence e a assinatura dele.
A classe e assinatura estão na API do JAVA:
http://download.oracle.com/javase/6/docs/api/

```
public static void main(String[] args) {
    Matriz A = Matriz.leia(3, 3);
    Matriz B = A.transposta();
    Matriz C = A.vezes(B);
    double determinante = determinante(C);
    imprime(A);
    imprime(B);
    imprime(C);
    imprime(determinante);
}

public static void imprime(String t[]) {...5 lines }

public static double determinante(Matriz X) {...5 lines }
```

Estudo de Caso

- Imagine um programa simples para calcular a soma de dois números e que exibe a seguinte



```java
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        for (int i=1; i<=30;i++){
            System.out.print("_");
        }
        System.out.println();
        System.out.print("Entre com o primeiro número:  ");
        int num = in.nextInt();
        for (int i=1; i<=30;i++){
            System.out.print("_");
        }
        System.out.println();
        System.out.print("Entre com o segundo número: ");
        int num2 = in.nextInt();
        for (int i=1; i<=30;i++){
            System.out.print("_");
        }
        System.out.println();
        System.out.println("A soma dos números é " +(num+num2));
        System.out.println();

        for (int i=1; i<=20;i++){
            System.out.print("_");
        }
         System.out.println();
    }
}
```

PROBLEMA!!!!
Muito código REPETIDO
Somente para imprimir os Cabeçalhos....

```
______________________________
Entre com o primeiro número:  2
______________________________
Entre com o segundo número: 2
______________________________
A soma dos números é 4

____________________BUILD SUCCESSFUL (total tim
```

Módulos – Parte 1

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        cabecalho(30);
        System.out.print("Entre com o primeiro número:  ");
        int num = in.nextInt();
        cabecalho(30);
        System.out.print("Entre com o segundo número: ");
        int num2 = in.nextInt();
        cabecalho(30);
        System.out.println("A soma dos números é " +(num+num2));
        System.out.println();
        cabecalho(30);
    }
    public static void cabecalho(int tamanho){
        for (int i=1; i<=tamanho;i++){
            System.out.print("_");
        }
        System.out.println();

    }
}
```

Definição do método.

OBS1: Dentro da classe princ

OBS2: Fora do método *main* *(antes ou depois)*

CHAMADA DO MÉTODO

```
public static void main(String[] args) {
    Scanner s= new Scanner(System.in);
    System.out.println("Entre com o ano de nascimento: ");
    int nasc = s.nextInt();
    System.out.println("Entre com o ano atual: ");
    int ano = s.nextInt();
    System.out.println("A idade é " + calcula_idade(nasc, ano));
}
public static int calcula_idade (int ano_de_nascimento, int ano_atual) {
    int i = ano_atual - ano_de_nascimento;
    return i;
}
```

CHAMADA DO MÉTODO

```
public static void main(String[] args) {
    Scanner s= new Scanner(System.in);
    System.out.println("Entre com o ano de nascimento: ");
    int nasc = s.nextInt();
    System.out.println("Entre com o ano atual: ");
    int ano = s.nextInt();
    System.out.println("A idade é " + calcula_idade(nasc, ano));
}
public static int calcula_idade (int ano_de_nascimento, int ano_atual) {
    int i = ano_atual - ano_de_nascimento;
    return i;
}
```

CHAMADA DO MÉTODO

```
public static void main(String[] args) {
    Scanner s= new Scanner(System.in);
    System.out.println("Entre com o ano de nascimento: ");
    int nasc = s.nextInt();
    System.out.println("Entre com o ano atual: ");
    int ano = s.nextInt();
    System.out.println("A idade é " + calcula_idade(nasc, ano));
}
public static int calcula_idade (int ano_de_nascimento, int ano_atual) {
    int i = ano_atual - ano_de_nascimento;
    return i;
}
```

Definição de Métodos

- “Algoritmo”
  1. Atribuir um nome para o método
  2. Definir se ele precisa ou não retornar algum valor
     2.1 Se sim: defina qual o tipo do valor (mesmo que o no return)
     2.2 Se não: o valor é “void”
  3. Definir se ele precisa de parâmetros
     3.1 Se sim: defina quais os nomes e o tipo de cada insumo
  4. Criar a assinatura do método em java
  5. Colocar o código dentro da assinatura
  6. Chamar o método

Definição de Métodos

- "Algoritmo"
  1. Dar um nome para o método
     - O nome deve indicar o que o método faz
     - Exemplo: cabecalho

```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        cabecalho(30);
        System.out.print("Entre com o primeiro número:  ");
        int num = in.nextInt();
        cabecalho(30);
        System.out.print("Entre com o segundo número: ");
        int num2 = in.nextInt();
        cabecalho(30);
        System.out.println("A soma dos números é " +(num+num2));
        System.out.println();
        cabecalho(30);
    }
    public static void cabecalho(int tamanho){
        for (int i=1; i<=tamanho;i++){
            System.out.print("_");
        }
        System.out.println();

    }
}
```

Assinatura indica onde começa a definição do método

Dentro da classe Principal

APÓS o método main

Definição de Métodos

- "Algoritmo"
  4. Criar uma assinatura para o método em JAVA
  - Outros exemplos…..

```
public static double pow (double base, double altura)
public static int soma (int num1, int num2)
public static double soma (double num1, double num2)
public static void compara (String nome1, String nome2)
public static int exemplo1 (String s, int a, double b)
public static void exemplo2 (double[] a, String[] b)
public static void main (String[] args)
```

Nosso "velho" amigo main

Chamada de Métodos

- Posso chamar quantas vezes eu quiser
- Posso chamar dentro de outros métodos
- Posso chamar dentro de outras classes (fora do escopo dessa disciplina)
- Posso chamar um método dentro dele mesmo, **RECURSIVIDADE** (fora do escopo dessa disciplina)

```
3  import java.util.*;
4
5  public class Modulo2 {
6
7      static final int k = 5;
8
9      public static void main(String[] args) {
10
11         Scanner entrada = new Scanner(System.in);
12         int num;
13         double y;
14
15         System.out.print("Digite o valor de x: ");
16         num = entrada.nextInt();
17
18         y = FCalculaY(num);
19
20         System.out.println("F("+ num + ") = " + y);
21     }
```

K é uma constante Global (declarada "fora" do main)

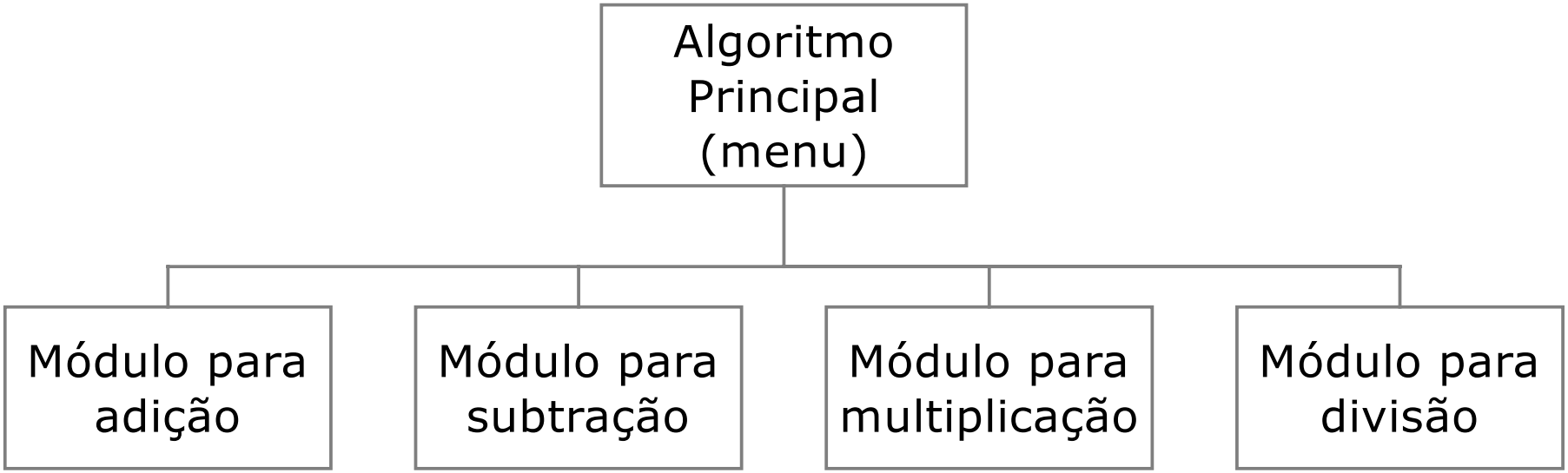
Chamada do Método

```
Digite o valor de x: 5
F(5) = 330.0
```

```
24     static double FCalculaY(int x) {
25         double aux1, aux2;
26         aux1 = 2 * Math.pow(x, 3);
27         aux2 = 3 * x * x + k;
28         return (aux1 + aux2);
29     }
30 }
```

Método que retorna valor

Poderíamos criar vários métodos...



Estudar o próximo exemplo!

Observe que ele usa o comando CASE.....

Exemplo: Java – Principal (main)

```
   import javax.swing.JOptionPane;
2. public class Menu {
3.   public static void main (String args[]){
4.     int opcao;
5.     opcao = Integer.parseInt(JOptionPane.showInputDialog(
6.         "Escolha a sua opção:\n" +
7.         "1 - Adição\n" +
8.         "2 - Subtração\n" +
9.         "3 - Multiplicação\n" +
10.            "4 - Divisão"));
11.        switch (opcao){
12.          case 1 : modAdicao(); break;
13.          case 2 : modSubtr(); break;
14.          case 3 : modMultipl(); break;
15.          case 4 : modDiv();break;
16.          default : JOptionPane.showMessageDialog(
17.              null, "Fim do Programa");
18.        }
19.     }
```

Classes Métodos

```
static void modAdicao( ){
      double v1;
      double v2;
      double res;
      v1 =
  Double.parseDouble(JOptionPane.showInputDialog(
          "Digite o  primeiro valor"));
      v2 =
  Double.parseDouble(JOptionPane.showInputDialog(
          "Digite o segundo valor"));
      res = v1 + v2;
      JOptionPane.showMessageDialog(
          null, "Soma = " + res);
  }
```

Classes Métodos

Exemplo: Java – modSubtr

```
static void modSubtr( ){
        double v1;
        double v2;
        double res;
        v1 = Double.parseDouble(JOptionPane.showInputDialog(
            "Digite o primeiro valor"));
        v2 = Double.parseDouble(JOptionPane.showInputDialog(
            "Digite o segundo valor"));
        res = v1 - v2;
        JOptionPane.showMessageDialog(null, "Subtração = " +
   res);
 }
```

Classes Métodos

Exemplo: Java – modMultipl

```
static void modMultipl( ){
   double v1;
   double v2;
   double res;
   v1 =
 Double.parseDouble(JOptionPane.showInputDialog(
       "Digite o primeiro valor"));
     v2 =
 Double.parseDouble(JOptionPane.showInputDialog(
        "Digite o segundo valor"));
     res = v1 * v2;
     JOptionPane.showMessageDialog( null,
 "Multiplicação = " + res);
}
```

Mais um exemplo.....

```
import javax.swing.JOptionPane;
public class Exemplo {
  public static void main(String args []){
    int numero =
          Integer.parseInt(JOptionPane.showInputDialog(
            "Entre o   número"));
    int fat = fatorial(numero);
    JOptionPane.showMessageDialog(null,
          "O fatorial de " + numero + " é " + fat);
  }
 static int fatorial (int numero){
    int f = 1;
    for (int i = 1; i <= numero; i++)
      f = f * i;
    return f;
  }
}
```

métodos de leitura e escrita em um vetor

```
public static void leVetor(int A[]) {
   Scanner s = new Scanner(System.in);
   for(int i = 0; i < A.length; i++)
      A[i] = s.nextInt();
}
public static void imprime(int V[]) {
   for(int i = 0; i < V.length; i++)
        System.out.print (V[i]+"\t");
   }
   System.out.println();
}
```

```
public static void main(String[] args) {
   Scanner sc = new Scanner(System.in);
   System.out.println("Entre com a
         dimensao do vetor e com seus
         elementos");
   int n = sc.nextInt();
   int x[] = new int[n];
   leVetor(x);
   imprime(x);

   }

}
```

Produto escalar de dois vetores

```
public class Aula7 {

    public static double InnerProduct(double A[], double B[]) {
        double soma = 0;
        for (int i = 0; i < A.length; i++) {
            soma += A[i] * B[i];
        }
        return soma;
    }

    public static void main(String[] args) {
        double u[] = {1, 2, 3};
        double v[] = {3, 2, 1};
        System.out.println(InnerProduct(u, v));
    }
}
```

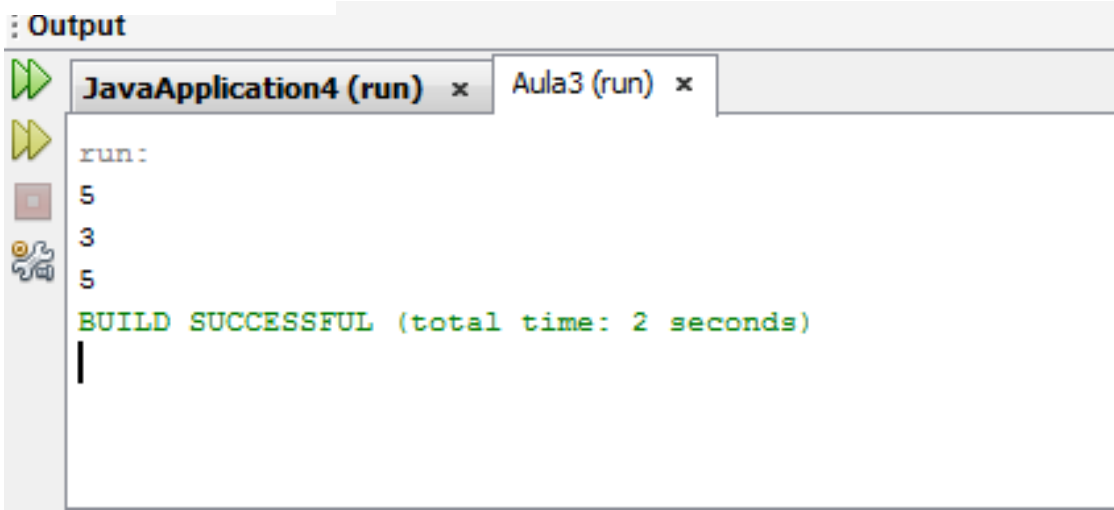
Observações sobre métodos

- Uma variável que é declarada dentro de um método é chamada de variável local.
- A variável local existe somente dentro do método.
- O método fica na memória durante a sua execução, depois disso ele é “destruído” da memória e suas variáveis também.
- Uma varíável criada fora de qualquer método e dentro da classe principal é chamada de variável global.
- A variável global existe dentro de qualquer método criada dentro do programa.
- A variável global existe durante toda execução do programa

```
package aula3;
public class Main {

    public static void main(String[] args) {
        int a=5;
        System.out.println(a);
        exemplo(a);

    }
    public static void exemplo(int b){
         int a=3;
         System.out.println(a);
         System.out.println(b);
    }

}
```

Qual a saída???

```java
package aula3;
public class Main {

    public static void main(String[] args) {
        int a=5;
        System.out.println(a);
        exemplo(a);

    }
    public static void exemplo(int b){
         int a=3;
         System.out.println(a);
         System.out.println(b);
    }

}
```



```java
package aula3;
public class Main {
    static int c=0;
    public static void main(String[] args) {
       int a=5;
       System.out.println(a);
       exemplo(a);
       System.out.println(c);

    }
    public static void exemplo(int b){
        int a=3;
        System.out.println(a);
        System.out.println(b);
        c = a*b;
        System.out.println(c);
    }

}
```

Exemplo de declaração De variável global

Qual a saída?

```java
package aula3;
public class Main {
    static int c=0;
    public static void main(String[] args) {
        int a=5;
        System.out.println(a);
        exemplo(a);
        System.out.println(c);

    }
    public static void exemplo(int b){
         int a=3;
         System.out.println(a);
         System.out.println(b);
         c = a*b;
         System.out.println(c);
    }

}
```

Exemplo de declaração De variável global

Estude esse exemplo!!!!

Output

JavaApplication4 (run) × | Debugger Console × | Aula3 (run) ×

```
run:
5
3
5
15
15
BUILD SUCCESSFUL (total time: 1 second)
```

Observações

- Mesmo nome dos métodos, porém com parâmetros distintos → isto funciona!
- Veja que conseguimos alterar o conteúdo de vetores
- Porém, não conseguimos alterar o conteúdo de variáveis simples

```java
public class Aula7 {
    public static int idade = 30;
    public static final String var = "Teste";
    public static int cont = 0;
    public static void main(String[] args) {
        Scanner s= new Scanner(System.in);
        int nasc = leInt("Entre com o ano de nascimento: "),
            ano = leInt("Entre com o ano atual: ");
        System.out.println("A idade é " + calcula_idade(nasc, ano));
    }
    public static int leInt(String s) {
        return Integer.parseInt(JOptionPane.showInputDialog(s));
    }
    public static double leDouble(String s) {
        return Double.parseDouble(JOptionPane.showInputDialog(s));
    }
    public static float leFloat(String s) {
        return Float.parseFloat(JOptionPane.showInputDialog(s));
    }
    public static int calcula_idade (int ano_de_nascimento, int ano_atual) {
        int i = ano_atual - ano_de_nascimento;
        return i;
    }

}
```

```java
public class Aula7 {
    public static void main(String[] args) {
        System.out.println("A idade é " +
                calcula_idade(leInt("Entre com o ano de nascimento: "),
                leInt("Entre com o ano atual: ")));
    }
    public static int leInt(String s) {
        return Integer.parseInt(JOptionPane.showInputDialog(s));
    }
    public static int calcula_idade (int ano_de_nascimento, int ano_atual) {
        int i = ano_atual - ano_de_nascimento;
        return i;
    }
}
```

Exercícios

1. Escrever um método simples para imprimir espaços em branco.
2. Elaborar um *método* para apresentar o somatório dos N primeiros números inteiros, definidos pelo usuário (1+2+3+4+5+6..+N)

Exercícios

3. Desenvolva um *método* para ler a temperatura em graus centígrados e apresente-a convertida em graus Fahrenheit. A fórmula de conversão é: F = (9 * C + 160)/5, na qual F é a temperatura em Fahrenheit e C é a temperatura em centígrados.
4. Escreva um método para calcular a série de Fibonacci de N termos, e retorne o valor do termo N. A série de Finonacci é formada pela seqüência 1, 1, 2, 3, 5, 8, 13, 21, 34, .. etc., em que se caracteriza pela soma de um termo posterior como seu anterior subseqüente.