

# Programação 1

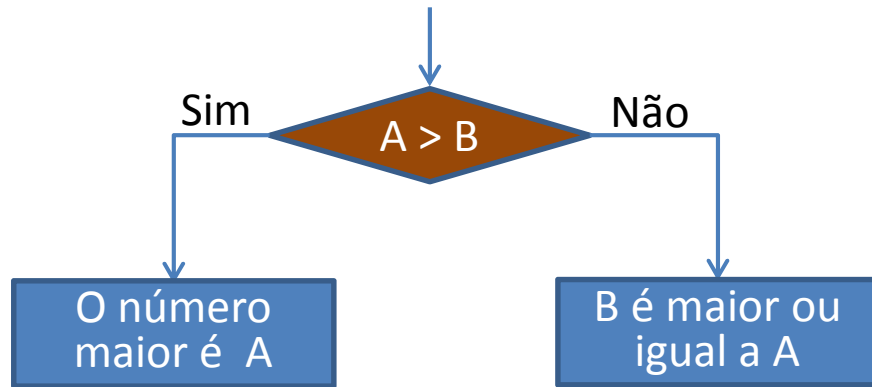
## Aula 4

Departamento de Eletrónica, Telecomunicações e Informática  
Universidade de Aveiro

<http://elearning.ua.pt/>

# Revisão da aula anterior

# Instrução if



```
if (A > B)
    System.out.println("O número maior é A");
else
    System.out.println("B é maior ou igual a A");
```

## Operação ternária

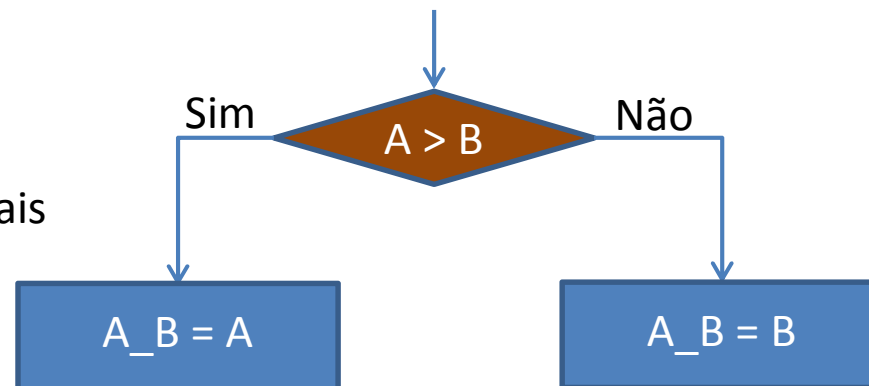
```
System.out.println(A > B ? "O número maior é A" : "B é maior ou igual a A");
```

Estas operações  
são iguais

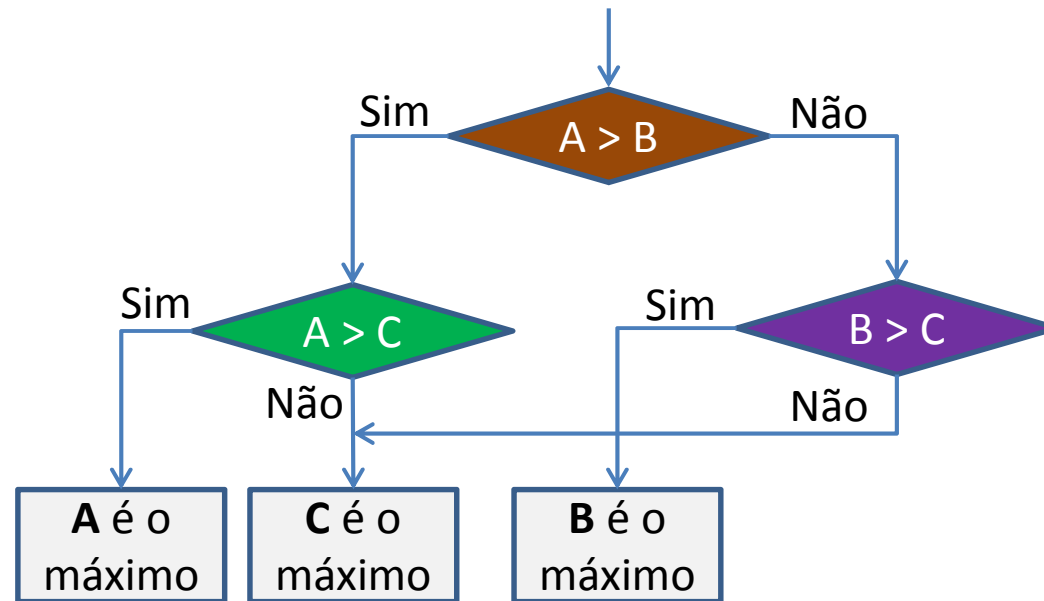
```
A_B = A > B ? A : B;
```

Estas operações são iguais

```
if (A > B) A_B = A;
else     A_B = B;
```



# Instrução if



```
System.out.println("O número maior é ");
```

```
if (A > B)
```

```
    if (A > C) System.out.println(A);
```

```
    else      System.out.println(C);
```

```
else
```

```
    if (B > C) System.out.println(B);
```

```
    else      System.out.println(C);
```

```
System.out.println("O número maior é ");
```

```
System.out.println(A > B ?
```

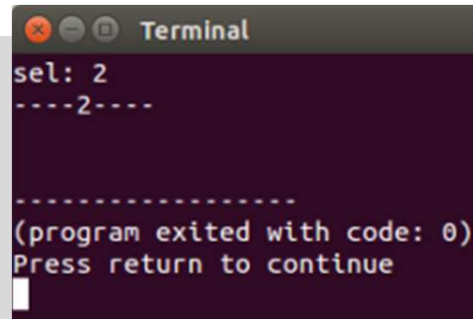
```
    (A > C ? A : C) :
```

```
    (B > C ? B : C));
```

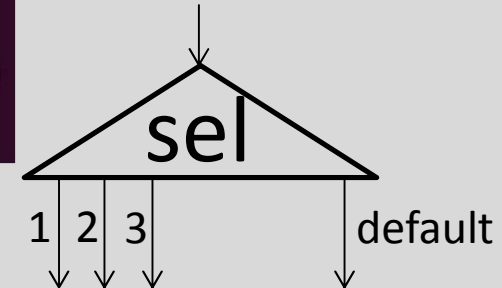
Estas operações  
são iguais

# Instrução **switch ... case**

```
int sel;  
System.out.print("sel: ");  
sel = sc.nextInt();  
switch(sel)  
{  
    case 1: System.out.println("----1----"); break;  
    case 2: System.out.println("----2----"); break;  
    case 3: System.out.println("----3----"); break;  
    default: System.out.println("diferente de 1 e 2 e 3");  
}
```



```
Terminal  
sel: 2  
----2----  
  
-----  
(program exited with code: 0)  
Press return to continue
```



```
double sel;  
System.out.print("sel: ");  
sel = sc.nextInt();  
switch(sel)
```

ERRO

```
double sel;  
System.out.print("sel: ");  
sel = sc.nextInt();  
switch((int)sel)
```

Ok

Só são permitidos valores convertíveis a inteiro

# Operadores aritméticos unários

```
int A = 1, B= 2, C = 3, Y; Y = A++ + B++ + C--; // Y = 6; A = 2; B = 3; C = 2
int A = 1, B= 2, C = 3 , Y; Y = ++A + B++ + C--; // Y = 7; A = 2; B = 3; C = 2
int A = 1, B= 2, C = 3 , Y; Y = ++A + ++B + --C; // Y = 7; A = 2; B = 3; C = 2
int A = 1, B= 2, C = 3 , Y; Y = ++A + ++B + ++C; // Y = 9; A = 2; B = 3; C = 4
int A = 1, B= 2, C = 3 , Y; Y = A++ + B++ + C++; // Y = 6; A = 2; B = 3; C = 4
int A = 1, B= 2, C = 3 , Y; Y = --A + --B + --C; // Y = 3; A = 0; B = 1; C = 2
```

## Instrução de atribuição com operação

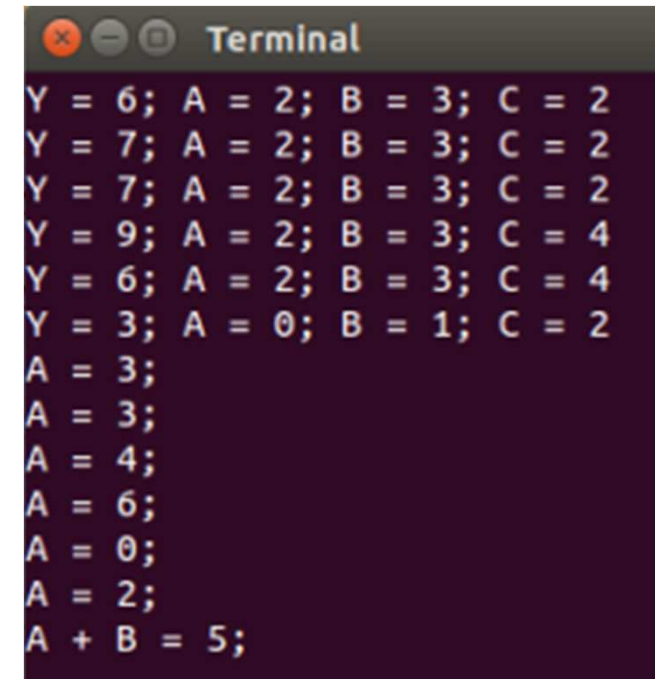
```
int A = 1, B= 2; A += B; // A = A + B = 3
int A = 1, B= 2; A += B++; // A = A + B++ = 3
int A = 1, B= 2; A += ++B; // A = A + ++B = 4
int A = 2, B= 3; A *= B; // A = A * B = 6
int A = 2, B= 3; A /= B; // A = A / B = 0
int A = 2, B= 3; A %= B; // A = A % B = 2
```

```

public class Operations
{
    public static void main (String args[])
    {
        int A = 1, B= 2, C = 3, Y; Y = A++ + B++ + C--;      // Y = 6; A = 2; B = 3; C = 2
        System.out.printf("Y = %d; A = %d; B = %d; C = %d\n",Y,A,B,C);
        A = 1; B= 2; C = 3; Y = ++A + B++ + C--;           // Y = 7; A = 2; B = 3; C = 2
        System.out.printf("Y = %d; A = %d; B = %d; C = %d\n",Y,A,B,C);
        A = 1; B= 2; C = 3; Y = ++A + ++B + --C;           // Y = 7; A = 2; B = 3; C = 2
        System.out.printf("Y = %d; A = %d; B = %d; C = %d\n",Y,A,B,C);
        A = 1; B= 2; C = 3; Y = ++A + ++B + ++C;           // Y = 9; A = 2; B = 3; C = 4
        System.out.printf("Y = %d; A = %d; B = %d; C = %d\n",Y,A,B,C);
        A = 1; B= 2; C = 3; Y = A++ + B++ + C++;           // Y = 6; A = 2; B = 3; C = 4
        System.out.printf("Y = %d; A = %d; B = %d; C = %d\n",Y,A,B,C);
        A = 1; B= 2; C = 3; Y = --A + --B + --C;           // Y = 3; A = 0; B = 1; C = 2
        System.out.printf("Y = %d; A = %d; B = %d; C = %d\n",Y,A,B,C);

        A = 1; B= 2; A += B;                                // A = A + B = 3
        System.out.printf("A = %d;\n",A);
        A = 1; B= 2; A += B++;                                // A = A + B++ = 3
        System.out.printf("A = %d;\n",A);
        A = 1; B= 2; A += ++B;                                // A = A + ++B = 4
        System.out.printf("A = %d;\n",A);
        A = 2; B= 3; A *= B;                                  // A = A * B = 6
        System.out.printf("A = %d;\n",A);
        A = 2; B= 3; A /= B;                                  // A = A / B = 0
        System.out.printf("A = %d;\n",A);
        A = 2; B= 3; A %= B;                                  // A = A % B = 2
        System.out.printf("A = %d;\n",A);
    }
}

```

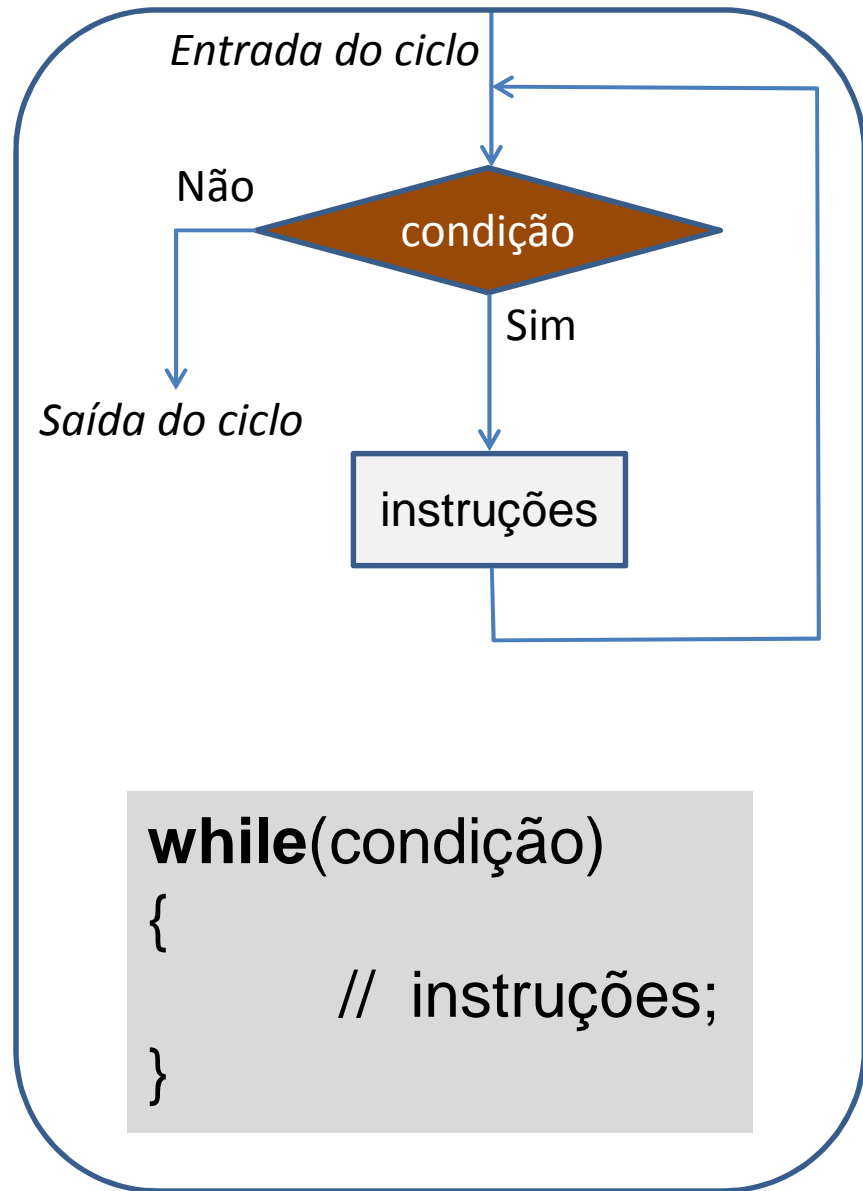
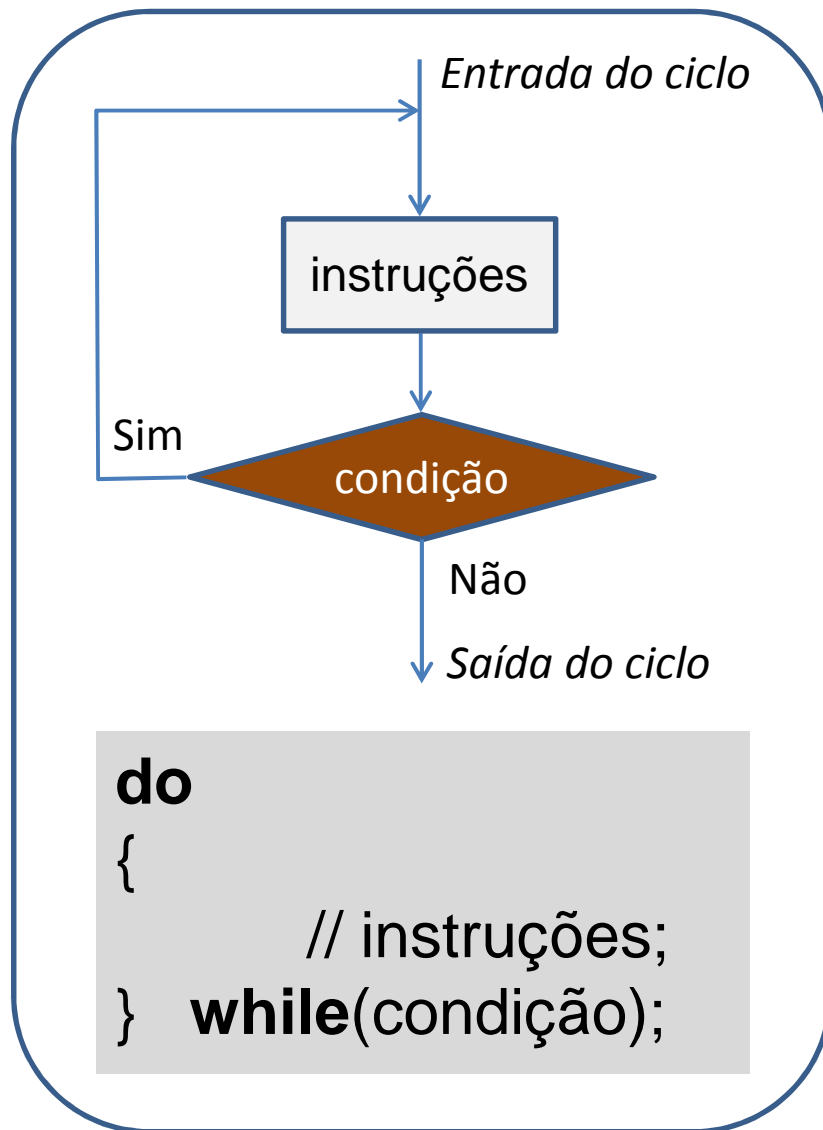


```

Terminal
Y = 6; A = 2; B = 3; C = 2
Y = 7; A = 2; B = 3; C = 2
Y = 7; A = 2; B = 3; C = 2
Y = 9; A = 2; B = 3; C = 4
Y = 6; A = 2; B = 3; C = 4
Y = 3; A = 0; B = 1; C = 2
A = 3;
A = 3;
A = 4;
A = 6;
A = 0;
A = 2;
A + B = 5;

```

# Ciclos





# Instrução repetitiva for

**for**(inicialização ; condição ; atualização)



Podemos apagar **inicialização** e/ou **condição** e/ou **atualização**  
mas não podemos apagar os pontos e vírgula (;)

Exemplos:

- 1) **for**(;;) – ciclo infinito (pode ser útil)
- 2) **for**(int a = 10;;) – ciclo que só tem **inicialização** (pode ser útil)
- 3) **for**(;a>b;) – ciclo que só tem **condição** (pode ser útil)
- 4) **for**(;;a++) – ciclo que só tem **atualização** (pode ser útil)
- 5) **for**(int a = 10; a>b;)
- 6) **for**(int a = 10; a>b; a++)
- 7) **for**(int a = 10; a>b; a++, b--)

## Instruções de salto **break** e **continue**

# Aula 4

- Introdução à programação modular
- Funções
- Tipos primitivos como argumentos
- Visibilidade das variáveis
- Exemplos

# Introdução à programação modular

- Na especificação de um problema obtemos um conjunto de tarefas básicas (ex: `ler`, `calcular`, `imprimir`).
- Com o aumento da complexidade dos problemas que queremos resolver, torna-se vantajoso a implementação e teste de cada uma dessas tarefas em separado.
- A linguagem JAVA permite-nos criar **funções** para implementar as várias tarefas básicas de um programa.
- Uma **função** permite realizar um determinado conjunto de operações e, se necessário, devolver um valor.
- As funções desenvolvidas pelo programador são chamadas no programa da mesma forma que as funções criadas por terceiros (por exemplo as funções de leitura ou escrita de dados ou as funções da classe `Math`).

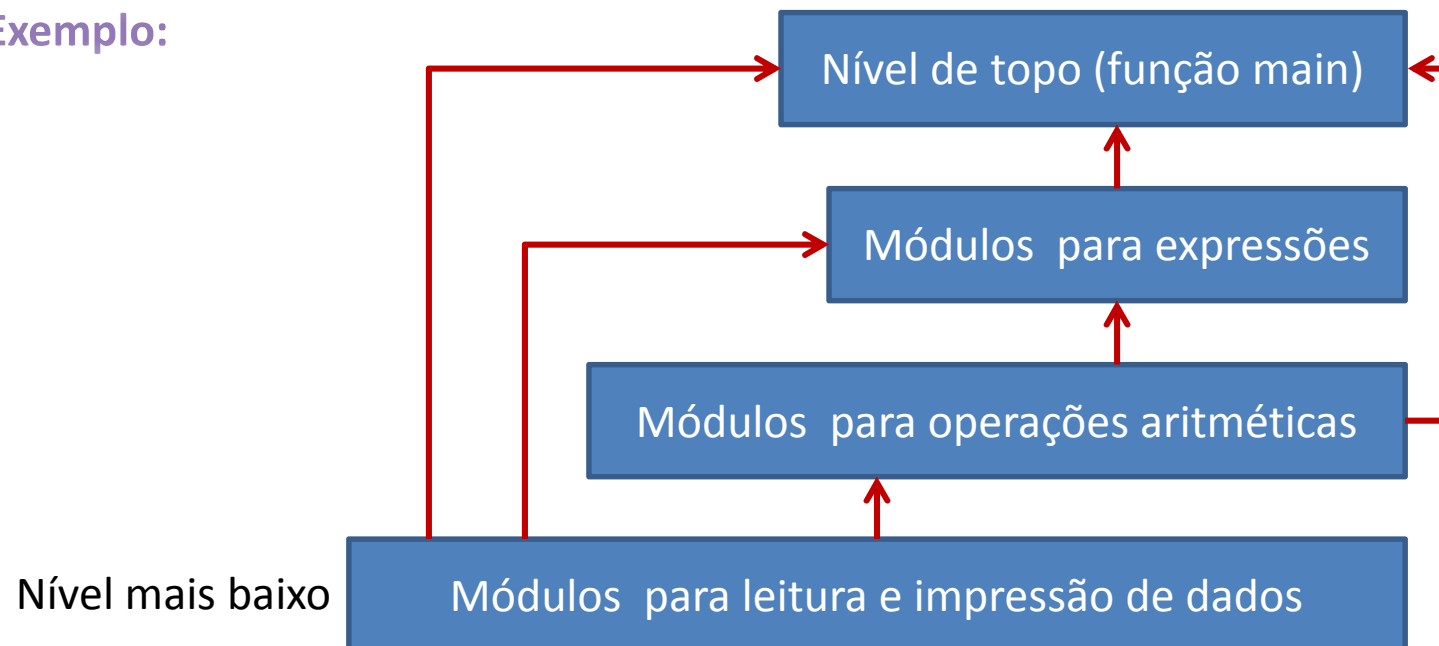
## Programação modular permite descrever o código de programa hierarquicamente

1. Descrever módulos do nível mais baixo (ponto 1);
2. Descrever módulos do nível 2 (utilizando módulos do ponto 1);
3. Descrever módulos do nível 3 (utilizando módulos dos pontos 1 e 2);

.....

Descrever módulos do nível de topo (utilizando módulos dos pontos anteriores);

Exemplo:



- Estrutura de um programa (relembrar):

```
//inclusão de classes externas
```

```
public class Programa
```

```
{// declaração de constantes e variáveis visíveis em
```

```
// classe Programa
```

```
public static void main (String[] args)
```

```
{
```

```
    // declaração de constantes e variáveis locais
```

```
    // sequências de instruções
```

```
}
```

```
funções desenvolvidas pelo programador
```

```
}
```

```
// definição de tipos de dados (registos)
```

- As funções são criadas depois **ou antes** da definição da função main.

# Definição de uma função

```
// cabeçalho da função  
{  
    // corpo da função  
}
```

No cabeçalho da função indicam-se os qualificadores de acesso (neste momento sempre **public static**), o tipo de resultado, o nome da função e dentro de parênteses curvos a lista de argumentos.

```
public static tipo nome (argumentos)  
{  
    // declaração de variáveis  
    // sequências de instruções  
}
```

# Definição de uma função

- Uma função é uma unidade auto-contida que recebe dados do exterior através dos argumentos, se necessário, realiza tarefas e devolve um resultado, se necessário.
- O resultado de saída de uma função pode ser de qualquer tipo primitivo (`int`, `double`, `char`, ...), de qualquer tipo referência (iremos ver mais à frente) ou `void` ( no caso de uma função não devolver um valor).
- A lista de argumentos (ou parâmetros) é uma lista de pares de indentificadores separados por vírgula, onde para cada argumento se indica o seu tipo de dados e o seu nome.
- O corpo da função assemelha-se à estrutura de um **módulo**.
- Se a função devolver um valor utiliza-se a palavra reservada **return** para o devolver.
- O valor devolvido na instrução de **return** deve ser compatível com o tipo de saída da função.

## Exemplo 1:

```
import java.util.*;
public class Ler
{
    static Scanner kb = new Scanner(System.in);
    public static int lerPositivo()
    {
        int x;
        do {
            System.out.print("Valor inteiro: ");
            x = kb.nextInt();
        } while(x < 0);
        return x;
    }

    public static void main(String[] args)
    {
        System.out.printf("soma = %d\n", soma(lerPositivo(),lerPositivo()));
    }

    public static int soma (int x, int y)
    {
        int soma;
        soma = x + y;
        return soma;
    }
}
```

Visibilidade



## Exemplo 1:

### Função soma

Tipo do valor de retorno

Nome da função

Tipo (int) e  
nome (x) do  
primeiro  
argumento

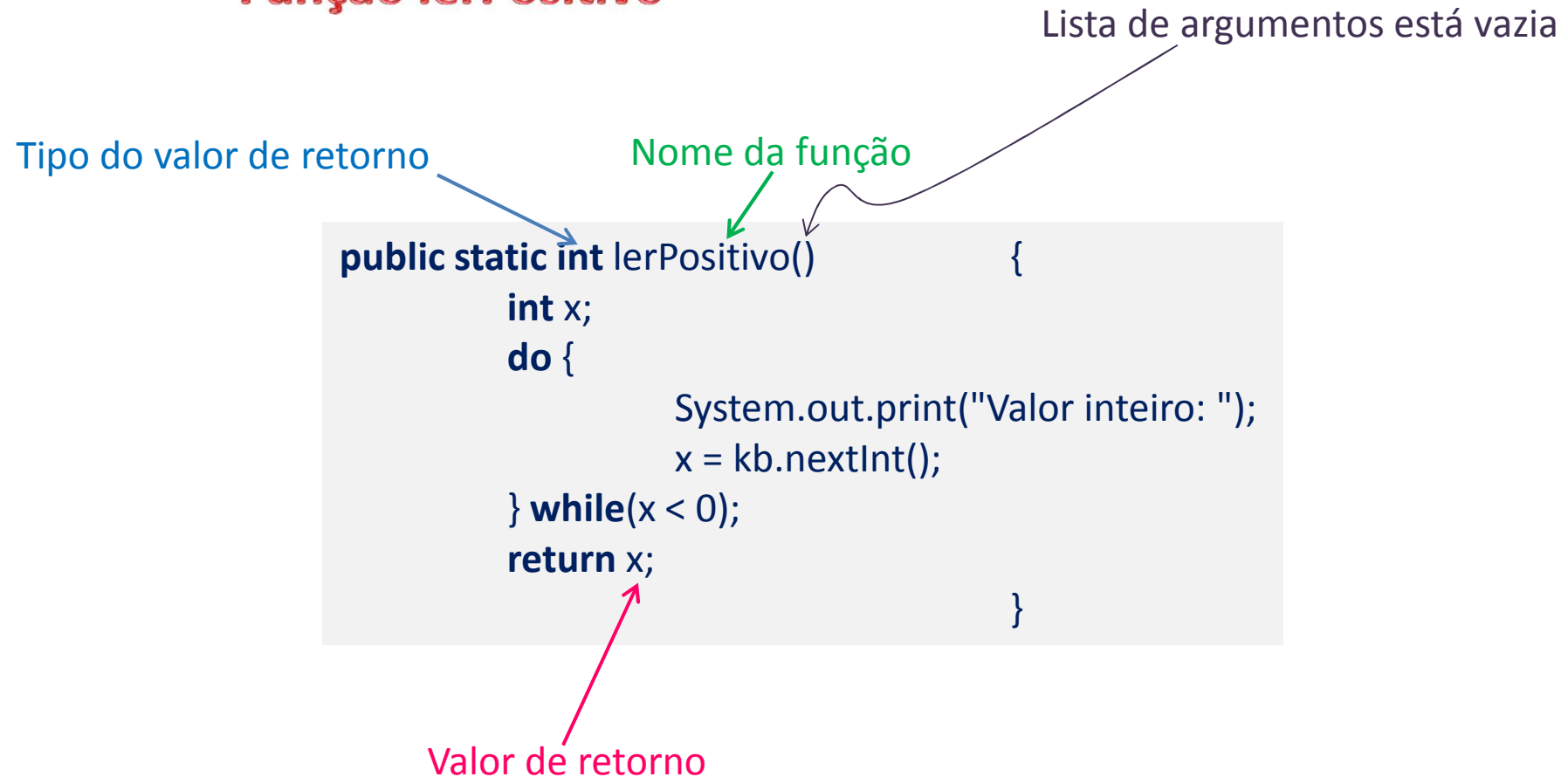
Tipo (int) e  
nome (y) do  
segundo  
argumento

```
public static int soma (int x, int y) {  
    int soma;  
    soma = x + y;  
    return soma;  
}
```

Valor de retorno

## Exemplo 1:

### Função lerPositivo



## ***Código alternativo:***

```
public static int soma (int x, int y)  {  
    int soma;  
    soma = x + y;  
    return soma;  
}
```

=

```
public static int soma (int x, int y)  {  
    return x + y;  
}
```

## Código alternativo:

```
public static int lerPositivo() {  
    int x;  
    do {  
        System.out.print("Valor inteiro: ");  
        x = kb.nextInt();  
    } while(x < 0);  
    return x;  
}
```

=

```
public static int lerPositivo() {  
    int x = 0;  
    for(;x <= 0;  
        System.out.print("Valor inteiro: ");  
        x = kb.nextInt());  
    return x;  
}
```

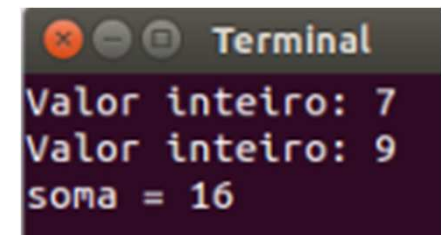
Operação vírgula

```

import java.util.*;
public class Ler
{
    static Scanner kb = new Scanner(System.in);
    public static int lerPositivo() {
        int x = 0;
        // do{
        //     System.out.print("Valor inteiro: ");
        //     x = kb.nextInt();
        // }while(x < 0);
        // return x;
        for(;x <= 0; System.out.print("Valor inteiro: "), x = kb.nextInt());
        return x;
    }

    public static void main(String[] args)
    {
        System.out.printf("soma = %d\n", soma(lerPositivo(),lerPositivo()) );
    }
    public static int soma (int x, int y){
        //int soma;
        //soma = x + y;
        //return soma;
        return x + y;
    } }

```



```

Terminal
Valor inteiro: 7
Valor inteiro: 9
soma = 16

```

```

... main (...){ // Soma de dois números positivos
    int a, b, r;
    a = lerPositivo(); // utilização das funções definidas pelo programador
    b = lerPositivo(); // da mesma forma que utilizamos todas as outras...
    r = soma(a, b); // o valor de a e b são passados á função soma
    printf("%d + %d = %d\n", a, b, r);
}
public static int lerPositivo(){
    ...
    do{
        ...
    }while(x < 0);
    return x; // devolução do valor lido através do teclado, após validação
}
public static int soma (int x, int y){ // neste exemplo x = a e y = b
    int soma;
    soma = x + y;
    return soma;
}

```

```

public static int soma (int x, int y) {
    return x + y;
}

```

**Exemplo 2:** Escreva um programa que permite calcular o fatorial de N ( $1 \leq N \leq 10$ ) utilizando uma função

Função fact



```
int N, fatorial = 1;
do {
    System.out.print("Introduza um numero: ");
    N = sc.nextInt();
    if (N > 10 || N < 1)
        System.out.println("o número errado");
    } while(N > 10 || N < 1);
    for (int i = 1; i <= N; i++)
        fatorial *= i;
    System.out.println("fatorial = "+fatorial);
```

```
public static int fact(int N)    {
    int fatorial = 1;
    do {
        System.out.print("Introduza um numero: ");
        N = sc.nextInt();
        if (N > 10 || N < 1)
            System.out.println("o número errado");
        } while(N > 10 || N < 1);
        for (int i = 1; i <= N; i++)
            fatorial *= i;
        return fatorial;
    }
```

**Exemplo 2:** Escreva um programa que permite calcular o fatorial de N ( $1 \leq N \leq 10$ ) utilizando uma função

Função fact

```
public static int fact(int N)    {
int fatorial = 1;
do {
    System.out.print("Introduza um numero: ");
    N = sc.nextInt();
    if (N > 10 || N < 1)
        System.out.println("o número errado");
    } while(N > 10 || N < 1);
for (int i = 1; i <= N; i++)
    fatorial *= i;
return fatorial;                }
```

O código pode ser melhorado:

```
public static int lerPositivo()  {
    int x;
    do {
        System.out.print("Valor positivo: ");
        x = sc.nextInt();
    } while(x < 0);    return x;
}
```

Ler dados

```
public static int fact(int N)  {
    int fatorial = 1;
    for (int i = 1; i <= N; i++)
        fatorial *= i;
    return fatorial;            }
```

Encontrar fatorial

Agora podemos reutilizar o código



**Exemplo 2:** Escreva um programa que permite calcular o fatorial de N ( $1 \leq N \leq 10$ ) utilizando uma função

```
import java.util.*;
public class FuncFact
{
    static Scanner sc = new Scanner(System.in);

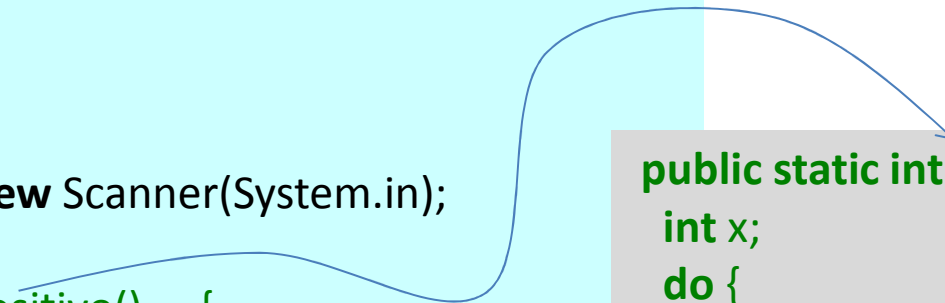
    public static int lerPositivo() {
        // .....

    }

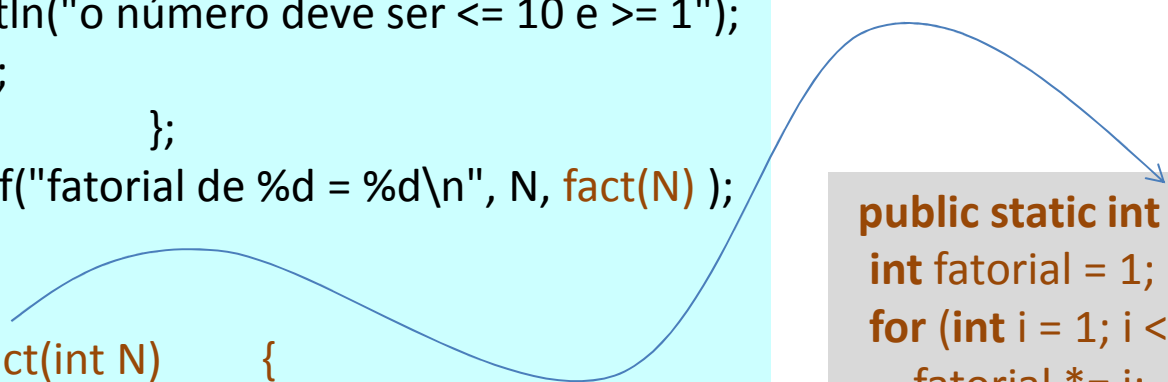
    public static void main(String[] args)
    {
        int N = lerPositivo();
        while(N > 10 || N < 1) {
            System.out.println("o número deve ser <= 10 e >= 1");
            N = lerPositivo();
        };
        System.out.printf("fatorial de %d = %d\n", N, fact(N) );
    }

    public static int fact(int N) {
        // .....

    }
}
```



```
public static int lerPositivo() {
    int x;
    do {
        System.out.print("Valor positivo: ");
        x = sc.nextInt();
    } while(x < 0);    return x;
}
```



```
public static int fact(int N) {
    int fatorial = 1;
    for (int i = 1; i <= N; i++)
        fatorial *= i;
    return fatorial;
}
```

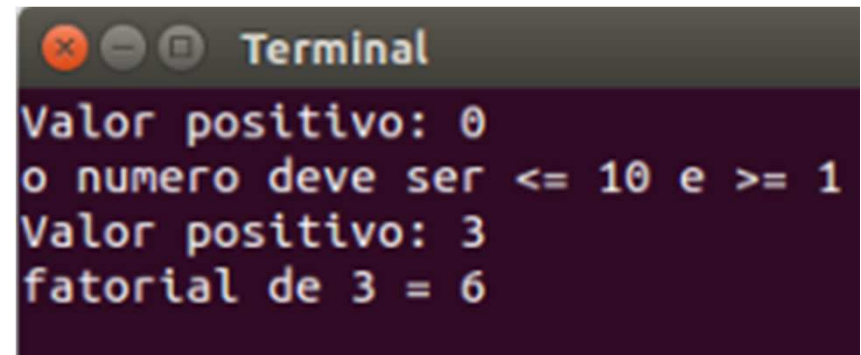
```

import java.util.*;
public class FuncFact
{
    static Scanner sc = new Scanner(System.in);
    public static int lerPositivo() {
        int x;
        do {
            System.out.print("Valor positivo: ");
            x = sc.nextInt();
        } while(x < 0);
        return x;
    }

    public static void main(String[] args)
    {
        int N = lerPositivo();
        while(N > 10 || N < 1) {
            System.out.println("o número deve ser <= 10 e >= 1");
            N = lerPositivo();
        };
        System.out.printf("fatorial de %d = %d\n", N, fact(N) );
    }

    public static int fact(int N) {
        int fatorial = 1;
        for (int i = 1; i <= N; i++)
            fatorial *= i;
        return fatorial;
    }
}

```



```

Terminal
Valor positivo: 0
o numero deve ser <= 10 e >= 1
Valor positivo: 3
fatorial de 3 = 6

```

Uma vantagem de modularidade é a reutilização do código

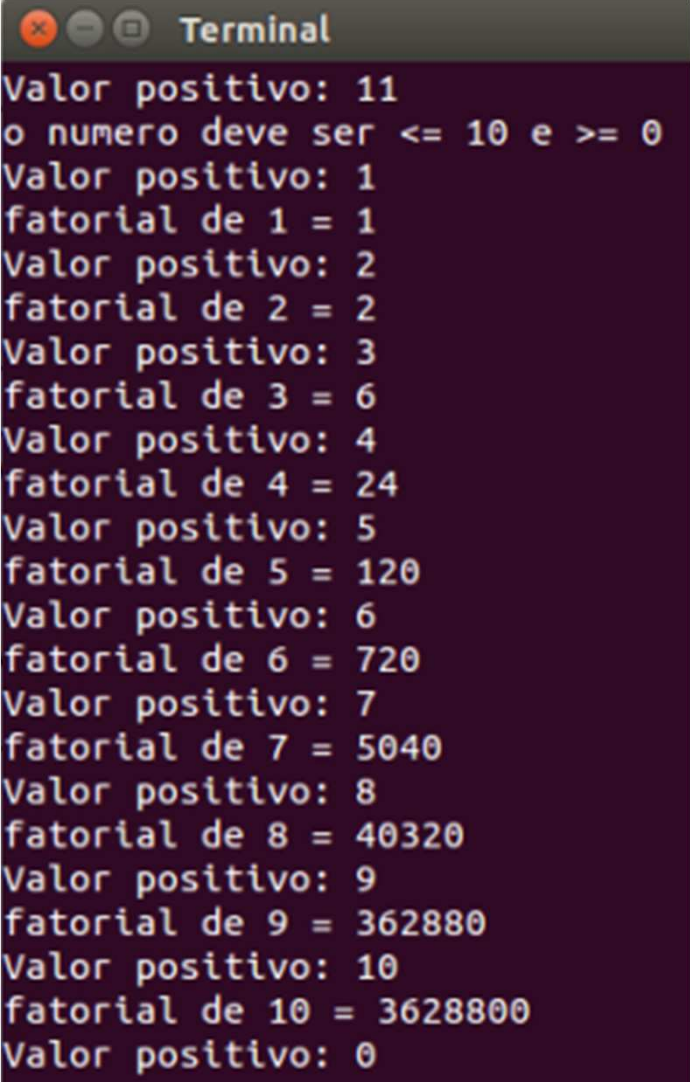
```
import java.util.*;
public class FuncFact
{
    static Scanner sc = new Scanner(System.in);
    public static int lerPositivo() {
        // .....
    }
    public static void main(String[] args)
    {
        int N = lerPositivo();
        while(N != 0) {
            while(N > 10 || N < 0) {
                System.out.println("o número deve ser <= 10 e >= 0");
                N = lerPositivo();
            }
            System.out.printf("fatorial de %d = %d\n", N, fact(N));
            N = lerPositivo();
        }
    }
    public static int fact(int N) {
        // .....
    }
}
```

As funções podem ser chamadas várias vezes com argumentos diferentes

```

import java.util.*;
public class FuncFactReutil
{
    static Scanner sc = new Scanner(System.in);
    public static int lerPositivo() {
        int x;
        do {
            System.out.print("Valor positivo: ");
            x = sc.nextInt();
        } while(x < 0);
        return x;
    }
    public static void main(String[] args)
    {
        int N = lerPositivo();
        while(N != 0) {
            while(N > 10 || N < 0) {
                System.out.println("o número deve ser <= 10 e >= 0");
                N = lerPositivo();
            }
            System.out.printf("fatorial de %d = %d\n", N, fact(N));
            N = lerPositivo();
        }
    };
    public static int fact(int N) {
        int fatorial = 1;
        for (int i = 1; i <= N; i++)
            fatorial *= i;
        return fatorial;
    }
}

```



```

Terminal
Valor positivo: 11
o numero deve ser <= 10 e >= 0
Valor positivo: 1
fatorial de 1 = 1
Valor positivo: 2
fatorial de 2 = 2
Valor positivo: 3
fatorial de 3 = 6
Valor positivo: 4
fatorial de 4 = 24
Valor positivo: 5
fatorial de 5 = 120
Valor positivo: 6
fatorial de 6 = 720
Valor positivo: 7
fatorial de 7 = 5040
Valor positivo: 8
fatorial de 8 = 40320
Valor positivo: 9
fatorial de 9 = 362880
Valor positivo: 10
fatorial de 10 = 3628800
Valor positivo: 0

```

## Tipos primitivos como argumentos

- Tomando como exemplo o programa do slide 22 (soma de dois positivos), podemos ver que a função `lerPositivo` não recebe argumentos e devolve um valor inteiro.
- Quando uma função é chamada várias vezes, o seu valor de retorno pode ser armazenado nas variáveis.
- A função `soma` recebe dois argumentos do tipo inteiro e devolve também um valor inteiro.
- Quanto executada, o conteúdo das variáveis `a` e `b` são passados para “dentro” da função `soma` através dos argumentos.
- Sempre que uma função é usada, o programa “salta” para dentro da função, executa o seu código e quando termina continua na instrução que usa a chamada da função.

# Visibilidade das variáveis

- Vimos que um programa pode conter várias funções, sendo obrigatoriamente uma delas a função `main`.
- As **variáveis locais** apenas são visíveis no corpo da função onde são declaradas.
- As variáveis declaradas dentro de um bloco (ou seja dentro do conjunto de instruções delimitado por `{ ... }` têm visibilidade limitada ao bloco (por exemplo ciclo **for**).
- Podemos também ter **variáveis globais**, sendo estas declaradas fora da função `main`, dentro da classe que implementa o programa (têm de ser precedidas da palavra reservada **static**).

## Exemplo:

A função main pode ser representada como:

```
import java.util.*;
public class FuncFact
{
    static Scanner sc = new Scanner(System.in);
    public static int lerPositivo() {
        // .....
    }
    public static void main(String[] args)
    {
        int N = lerPositivo();
        while(N != 0) {
            while(N > 10 || N < 0) {
                System.out.println("o número deve ser <= 10 e >= 0");
                N = lerPositivo();
            }
            System.out.printf("fatorial de %d = %d\n", N, fact(N) );
            N = lerPositivo();
        }
    }
    // .....
}
```

```
public static void main(String[] args)
{
    for(int N = lerPositivo(); N != 0; ) {
        while(N > 10 || N < 0) {
            System.out.println("o número deve ser <= 10 e >= 0");
            N = lerPositivo();
        }
        System.out.printf("fatorial de %d = %d\n", N, fact(N) );
        N = lerPositivo();
    }
}
```

## Exemplo:

```
public static void main(String[] args)
{
    for(int N = lerPositivo(); N != 0;) {
        while(N > 10 || N < 0) {
            System.out.println("o número deve ser <= 10 e >= 0");
            N = lerPositivo();
        }
        System.out.printf("fatorial de %d = %d\n", N, fact(N) );
        N = lerPositivo();
    }
    System.out.printf("N = %d\n", N);
}
```

DrJava

Error: cannot find symbol  
symbol: variable N  
location: class ForGlobal

Geany

ForGlobal.java:26: error: cannot find symbol  
System.out.printf("N = %d\n", N);

symbol: variable N  
location: class ForGlobal

1 error  
Compilation failed.

```
public static void main(String[] args)
{
    int N;
    for(N = lerPositivo(); N != 0;) {
        while(N > 10 || N < 0) {
            System.out.println("o número deve ser <= 10 e >= 0");
            N = lerPositivo();
        }
        System.out.printf("fatorial de %d = %d\n", N, fact(N) );
        N = lerPositivo();
    }
    System.out.printf("N = %d\n", N);
}
```

Ok

```
Terminal
Valor positivo: 4
fatorial de 4 = 24
Valor positivo: 0
N = 0
```



## Exemplo:

Mal

```
import java.util.*;
public class Global
{
    static Scanner sc = new Scanner(System.in);
    static int x, N;
    public static int lerPositivo() {
        do {
            System.out.print("Valor positivo: ");
            x = sc.nextInt();
        } while(x < 0);
        return x;
    }
    public static void main(String[] args)
    {
        for(N = lerPositivo(); N != 0;) {
            while(N > 10 || N < 0) {
                System.out.println("o número deve ser <= 10 e >= 0");
                N = lerPositivo();
            }
            System.out.printf("fatorial de %d = %d\n", N, fact());
            N = lerPositivo();
        }
        System.out.printf("N = %d\n", N);
    }

    public static int fact() {
        int fatorial = 1;
        for (int i = 1; i <= N; i++)
            fatorial *= i;
        return fatorial;
    }
}
```

Geralmente, Ok

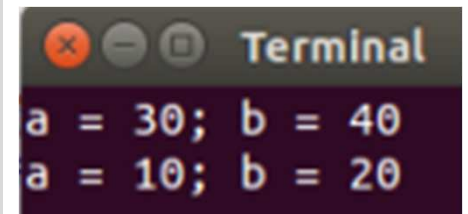
Deve evitar  
variáveis globais

Não use variáveis  
(objetos) globais  
sem necessidade

## Exemplo:

```
public class Bloco
{
    public static void main(String[] args)
    {
        { int a = 30, b = 40;
          System.out.printf("a = %d; b = %d\n", a,b); // a = 30; b = 40
        }
        int a = 10, b = 20;
        System.out.printf("a = %d; b = %d\n", a,b);    // a = 10; b = 20
    } }
```

← Um bloco



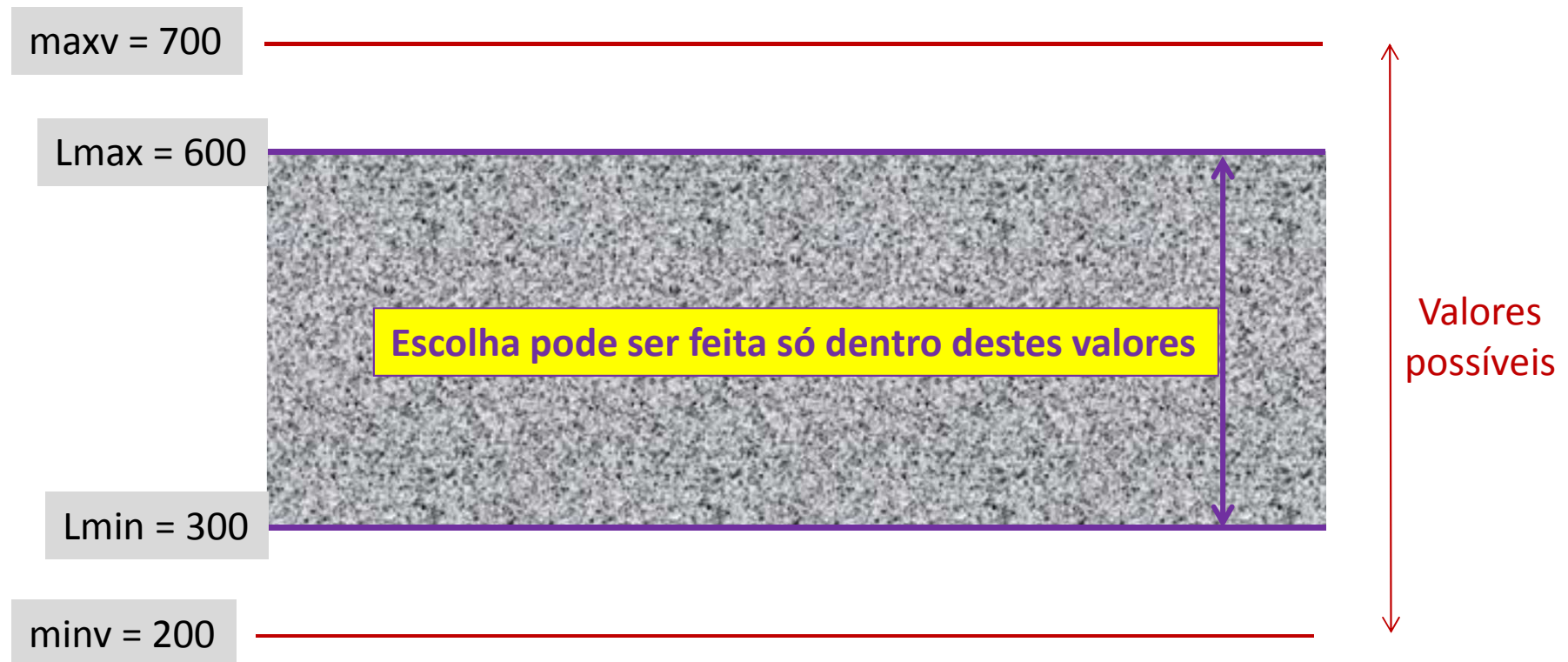
```
Terminal
a = 30; b = 40
a = 10; b = 20
```

```
public class Bloco
{
    public static void main(String[] args)
    { int a = 10, b = 20;
      { int a = 30, b = 40;
        System.out.printf("a = %d; b = %d\n", a,b); // a = 30; b = 40
      }
      System.out.printf("a = %d; b = %d\n", a,b);    // a = 10; b = 20
    } }
```

Error: a is already defined in main(java.lang.String[])  
Error: b is already defined in main(java.lang.String[])

**Exemplo 3:** Filtragem de números entre minv e maxv permite encontrar todos os números entre Lmin e Lmax ( $\text{minv} < \text{Lmin} < \text{Lmax} < \text{maxv}$ ). O programa escolhe um número aleatoriamente. Vamos assumir que devem ser escolhidos N números.

```
Int maxv = 700, minv = 200, Lmin = 300, Lmax = 600, N = 1000;
```



**Exemplo 3:** Filtragem de números entre minv e maxv permite encontrar todos os números entre Lmin e Lmax ( $\text{minv} < \text{Lmin} < \text{Lmax} < \text{maxv}$ ). O programa escolhe um número aleatoriamente. Vamos assumir que devem ser escolhidos N números.

```
public class filtragem
{
    public static void main(String[] args)
    {
        int maxv = 700, minv = 200, Lmin = 300, Lmax = 600, N = 1000;
        int data;
        int cont = 0;
        for(int i = 1; i <= N; i++)
        {
            data= (int)((maxv - minv)*Math.random()) + 200;
            if (data>= Lmin && data <= Lmax) { cont++; System.out.print(data+" "); }
        }
        System.out.println("\ncont = " + cont);
    }
}
```

**Exemplo 3:** Filtragem de números entre minv e maxv permite encontrar todos os números entre Lmin e Lmax ( $\text{minv} < \text{Lmin} < \text{Lmax} < \text{maxv}$ ). O programa escolhe um número aleatoriamente. Vamos assumir que devem ser escolhidos N números.

```
import java.util.*;
public class filtragem
{
    static Scanner sc = new Scanner(System.in);
    public static void main(String[] args)
    { int maxv, minv, Lmin, Lmax, N;
      do {
          System.out.print("maxv ?"); maxv = sc.nextInt();
          System.out.print("minv ?"); minv = sc.nextInt();
          System.out.print("Lmin ?"); Lmin = sc.nextInt();
          System.out.print("Lmax ?"); Lmax = sc.nextInt();
          System.out.print("N ?"); N = sc.nextInt();
          filter(maxv,minv, Lmin, Lmax, N);
          System.out.print("Entrar 0 para terminar");
      }
      while( sc.nextInt() != 0);
    }
    public static void filter(int maxv,int minv, int Lmin, int Lmax, int N)
    // .....
}
```

**Exemplo 3:** Filtragem de números entre minv e maxv permite encontrar todos os números entre Lmin e Lmax ( $\text{minv} < \text{Lmin} < \text{Lmax} < \text{maxv}$ ). O programa escolhe um número aleatoriamente. Vamos assumir que devem ser escolhidos N números.

```
public static void filter(int maxv, int minv, int Lmin, int Lmax, int N)
{
    int data;
    int cont = 0;
    for(int i = 1; i <= N; i++)
    {
        data = (int)((maxv - minv)*Math.random()) + minv;
        if (data >= Lmin && data <= Lmax) { cont++; System.out.print(data+" "); }
    }
    System.out.println("\ncont = " + cont);
}
```

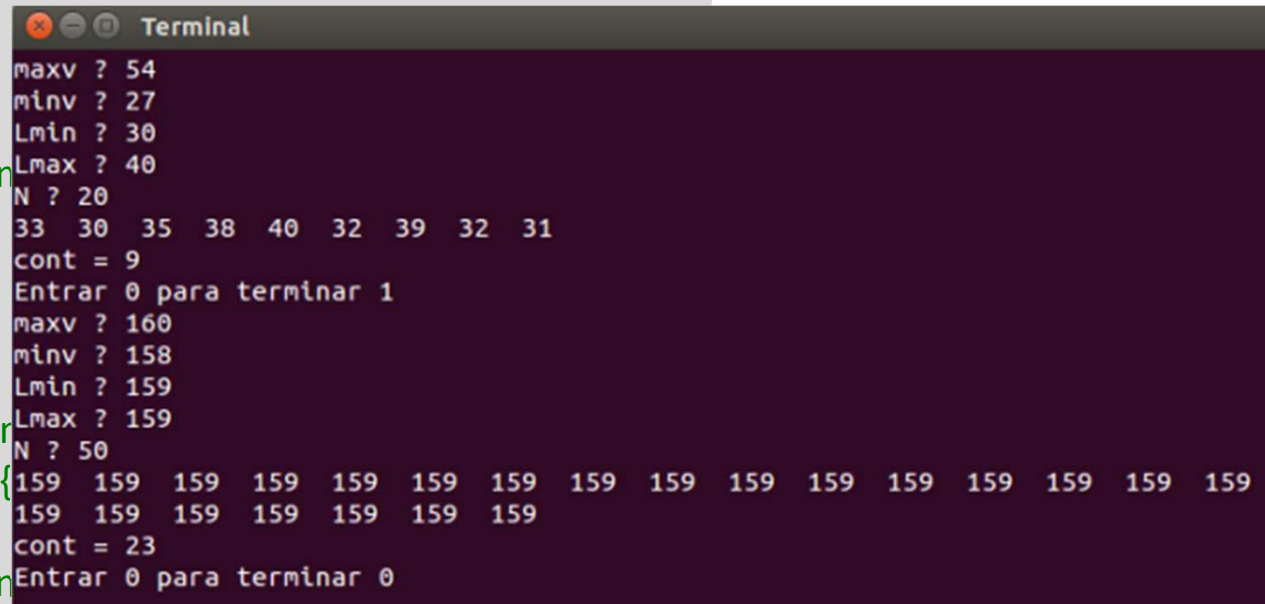
Agora a função *filter* pode ser chamada várias vezes para vários valores de argumentos

```
public class filtragem
{
    public static void main(String[] args)
    {
        int maxv = 700, minv = 200, Lmin = 300, Lmax = 600, N = 1000;
        int data;
        int cont = 0;
        for(int i = 1; i <= N; i++)
        {
            data = (int)((maxv - minv)*Math.random()) + 200;
            if (data >= Lmin && data <= Lmax) { cont++; System.out.print(data+" "); }
        }
        System.out.println("\ncont = " + cont);
    }
}
```

```

import java.util.*;
public class filtragem
{
    static Scanner sc = new Scanner(System.in);
    public static void main(String[] args)
    {
        int maxv, minv, Lmin, Lmax, N;
        do {
            System.out.print("maxv ? "); maxv = sc.nextInt();
            System.out.print("minv ? "); minv = sc.nextInt();
            System.out.print("Lmin ? "); Lmin = sc.nextInt();
            System.out.print("Lmax ? "); Lmax = sc.nextInt();
            System.out.print("N ? "); N = sc.nextInt();
            filter(maxv,minv, Lmin, Lmax, N);
            System.out.print("Entrar 0 para terminar ");
        }
        while( sc.nextInt() != 0);
    }
    public static void filter(int maxv,int minv, int Lmin, int Lmax, int N)
    {
        int data;
        int cont = 0;
        for(int i = 1; i <= N; i++)
        {
            data= (int)((maxv - minv)*Math.random() + minv);
            if (data >= Lmin && data <= Lmax) {
                cont++;
            }
        }
        System.out.println("\ncont = " + cont);
    }
}

```



```

Terminal
maxv ? 54
minv ? 27
Lmin ? 30
Lmax ? 40
N ? 20
33 30 35 38 40 32 39 32 31
cont = 9
Entrar 0 para terminar 1
maxv ? 160
minv ? 158
Lmin ? 159
Lmax ? 159
N ? 50
159 159 159 159 159 159 159 159 159 159 159 159 159 159 159 159 159 159 159 159
159 159 159 159 159 159 159
cont = 23
Entrar 0 para terminar 0

```

**Exemplo 4:** Representar um valor inteiro positivo **v** em binário

```
import java.util.Scanner;
public class inteiro_binario {
    static Scanner sc = new Scanner(System.in);
    public static void main (String args[]) {
        int v;
        System.out.print("Valor de inteiro : ");
        v = sc.nextInt();
        System.out.print("\nValor binário de inteiro " + v + " é ");
        for (int i = 31; i >= 0; i--)
            System.out.print((v >> i) & 1);
    }
}
```

[illegible]

```
inteiro 45
inteiro 255
inteiro 10
inteiro 20
inteiro 30
inteiro 81936
```

binário 101101  
binário 11111111  
binário 1010  
binário 10100  
binário 11110  
binário 10100000000010000

→  $2^0 = 1$   
 $2^1 = 2$   
 $2^2 = 4$   
→  $2^3 = 8$   
 $2^4 = 16$   
→  $2^5 = 32$   
 $2^6 = 64$   
 $2^7 = 128$   
 $2^8 = 256$   
 $2^9 = 512$   
 $2^{10} = 1024$   
 $2^{11} = 2048$   
 $2^{12} = 4096$   
 $2^{13} = 8192$   
 $2^{14} = 16384$   
 $2^{15} = 32768$



**Exemplo 4:** Representar um valor inteiro positivo **v** em binário

```
import java.util.*;

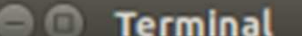
public class inteiro_binario {
    static Scanner sc = new Scanner(System.in);
    public static int lerPositivo() {
        int x = 0;
        for(; x <= 0; System.out.print("Valor inteiro: "), x = sc.nextInt());
        return x;
    }
    public static void conv_imp(int v)
    {
        System.out.print("\nValor binário de inteiro " + v + " é ");
        for (int i = 31; i >= 0; i--)
            System.out.print((v >> i) & 1);
    }
    public static void main (String args[]) {
        int v = lerPositivo();
        conv_imp(v);
    }
}
```

[illegible]

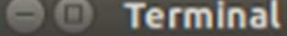


**Exemplo 5:** Representar um valor inteiro positivo **v** em binário *e remover os primeiros zeros*

```
public static void conv_imp(int v)
{
    System.out.print("\nValor binário de inteiro " + v + " é ");
    for (int i = 31; i >= 0; i--)
        if ((v >> i) == 0) continue;
        else System.out.print((v >> i) & 1);
}
```

[illegible]

A terminal window titled "Terminal" with a dark background. It shows the prompt "Valor inteiro: 45" followed by the output "Valor binario de inteiro 45 e 101101".

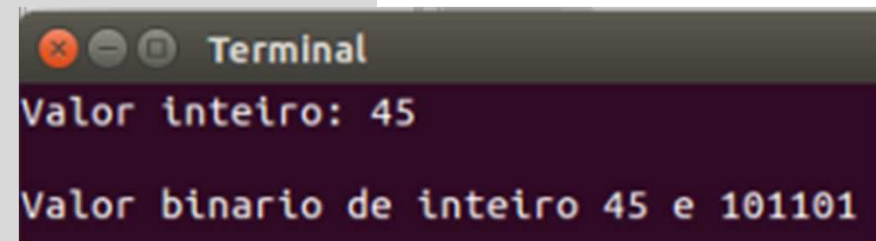


A terminal window titled "Terminal" with standard window controls (close, minimize, maximize). The terminal has a dark purple background and displays two lines of white text: "Valor inteiro: 11" and "Valor binario de inteiro 11 e 1011".

```

import java.util.*;
public class inteiro_binario_remove_zeros {
    static Scanner sc = new Scanner(System.in);
    public static int lerPositivo() {
        int x = 0;
        for(;x <= 0;System.out.print("Valor inteiro: "),x = sc.nextInt());
        return x;
    }
    public static void conv_imp(int v)
    {
        System.out.print("\nValor binário de inteiro " + v + " é ");
        for (int i = 31; i >= 0; i--)
            if ((v >> i) == 0) continue;
            else System.out.print((v >> i) & 1);
    }
    public static void main (String args[]) {
        int v = lerPositivo();
        conv_imp(v);
    }
}

```

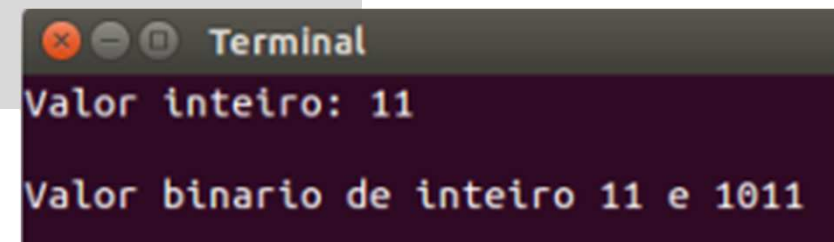


```

Terminal
Valor inteiro: 45

Valor binario de inteiro 45 e 101101

```



```

Terminal
Valor inteiro: 11

Valor binario de inteiro 11 e 1011

```

**Exemplo 6:** Representar o número inteiro positivo  $V$  (expresso em decimal) num sistema posicional  $N$  ( $N = \{2,3,4,5,6,7,8,9\}$ ) e verificar o resultado.

Regras de conversão do valor  $V$ :

1. Se  $V < N$  a conversão já esta feita.
2. Se  $V \geq N$  dividir  $V$  por  $N$  ( $V \neq N$ ) e gravar o resto da divisão.
3. Se  $V < N$ , gravar  $V$  e a conversão já está pronta.

Exemplo para  $V=35$ ,  $N=8$ :

1.  $35 > 8$ , i.e.  $V > N$ .
2.  $V/8 = 4$ ,  $V\%8 = 3$ . Gravar 3.
3.  $V=4$ ,  $V < N$ . Gravar 4

O resultado é 43.

Verificação:  $4 \cdot 8^1 + 3 \cdot 8^0 = 32 + 3 = 35$

Exemplo para  $V=35$ ,  $N=2$ :

1.  $35 > 2$ , i.e.  $V > N$ .
2.  $V/2 = 17$ ,  $V\%2 = 1$ . Gravar 1.
3.  $(V=17)/2 = 8$ ,  $V\%2 = 1$ . Gravar 1.
4.  $(V=8)/2 = 4$ ,  $V\%2 = 0$ . Gravar 0.
5.  $(V=4)/2 = 2$ ,  $V\%2 = 0$ . Gravar 0.
6.  $(V=2)/2 = 1$ ,  $V\%2 = 1$ . Gravar 0.
7. O resultado é 110011.

Verificação:

$$\begin{aligned} & 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = \\ & = 32 + 0 + 0 + 0 + 2 + 1 = 35 \end{aligned}$$

Ver dígitos com cores iguais

Decimais	Binários	Octais	Hexadecimais
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
<b>20</b>	<b>10100</b>	<b>24</b>	<b>14</b>

## *Alguns exemplos adicionais*

Decimal:  $2 \times 10^1 + 0 \times 10^0 = 20 + 0 = 20$

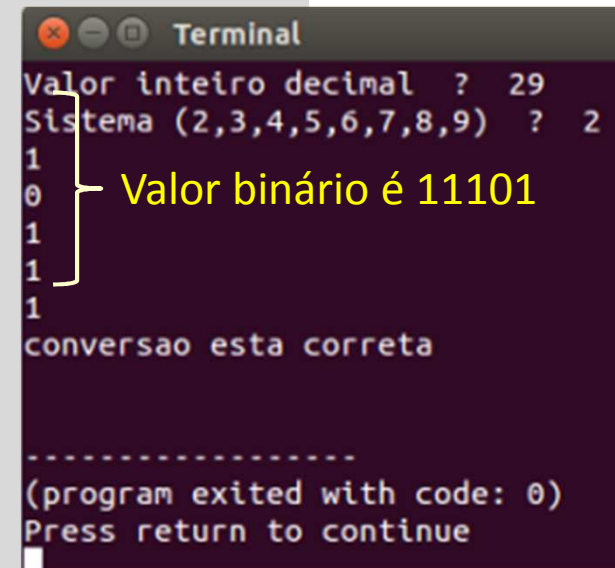
Binário:  $1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$   
 $16 + 0 + 4 + 0 + 0 = 20$

Octal:  $2 \times 8^1 + 4 \times 8^0 = 16 + 4 = 20$

Hexadecimal:  $1 \times 16^1 + 4 \times 16^0 = 16 + 4 = 20$

**Exemplo 6:** Representar o número inteiro positivo V (expresso em decimal) num sistema posicional N ( $N = \{2,3,4,5,6,7,8,9\}$ ) e verificar o resultado.

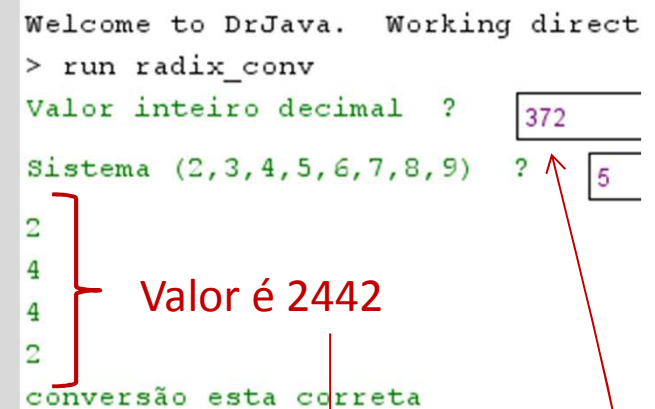
```
import java.util.*;
public class radix_conv
{
    static Scanner sc = new Scanner(System.in);
    public static void main(String[] args)
    { int V_gravado, V, N, tmp=0, indice=0;
      System.out.print("Valor inteiro decimal ? ");
      V = sc.nextInt();
      V_gravado = V;
      System.out.print("Sistema (2,3,4,5,6,7,8,9) ? ");
      N = sc.nextInt();          // é necessário verificar entrada
      for(;;)
      { if(V < N) { System.out.println(V);
                    tmp += (V%N)*Math.pow(N,indice++);
                    break; }
        else { System.out.println(V%N);
                tmp += (V%N)*Math.pow(N,indice++);
                V /= N; }
        if (tmp == V_gravado) System.out.println("conversão está correta");
        else System.out.println("conversão não esta correta " + tmp);
      }
    }
```



```
Terminal
Valor inteiro decimal ? 29
Sistema (2,3,4,5,6,7,8,9) ? 2
1
0
1
1
1
conversao esta correta

-----
(program exited with code: 0)
Press return to continue
```

Valor binário é 11101



```
Welcome to DrJava. Working direct
> run radix_conv
Valor inteiro decimal ? 372
Sistema (2,3,4,5,6,7,8,9) ? 5
2
4
4
2
conversão esta correta
```

Valor é 2442

$$2 \times 5^3 + 4 \times 5^2 + 4 \times 5^1 + 2 \times 5^0 = 250 + 100 + 20 + 2 = 372$$

**Exemplo 6:** Representar o número inteiro positivo V (expresso em decimal) num sistema posicional N ( $N = \{2,3,4,5,6,7,8,9\}$ ) e verificar o resultado.

```
import java.util.*;
public class radix_conv
{
    static Scanner sc = new Scanner(System.in);
    public static int lerPositivo() {
        // .....
    }
    public static void main(String[] args)
    {
        conv_e_imprimir(lerPositivo());
    }
    public static void conv_e_imprimir(int V)
    {
        int V_gravado = V, N, tmp=0, indice=0;
        System.out.print("Sistema (2,3,4,5,6,7,8,9) ? ");
        N = sc.nextInt();          // é necessário verificar entrada
        for(;;)
        {
            if(V < N) { System.out.println(V);
                        tmp += (V%N)*Math.pow(N,indice++);
                        break; }
            else { System.out.println(V%N);
                  tmp += (V%N)*Math.pow(N,indice++);
                  V /= N; }
        }
        if (tmp == V_gravado) System.out.println("conversão está correta");
        else System.out.println("conversão não esta correta " + tmp);
    }
}
```



```

import java.util.*;
public class radix_conv
{ static Scanner sc = new Scanner(System.in);
  public static int lerPositivo() {
    int x = 0;
    for(;x <= 0;System.out.print("Valor inteiro: "),x = sc.nextInt());
    return x;
  }
  public static void main(String[] args)
  {   conv_e_imprimir(lerPositivo());   }
  public static void conv_e_imprimir(int V)
  {
    int V_gravado = V, N, tmp=0, indice=0;
    System.out.print("Sistema (2,3,4,5,6,7,8,9) ? ");
    N = sc.nextInt();          // é necessário verificar entrada
    for(;;)
      if(V < N) { System.out.println(V);
                 tmp += (V%N)*Math.pow(N,indice++);
                 break; }
      else { System.out.println(V%N);
            tmp += (V%N)*Math.pow(N,indice++);
            V /= N; }
    if (tmp == V_gravado) System.out.println("conversão está correta");
    else System.out.println("conversão não esta correta " + tmp);
  } }

```

```

Terminal
Valor inteiro: 45
Sistema (2,3,4,5,6,7,8,9) ? 3
0
0
2
1
1200
conversao esta correta

```

$$1 \times 3^3 + 2 \times 3^2 + 0 \times 3^1 + 0 \times 3^0 = 27 + 18 = 45$$

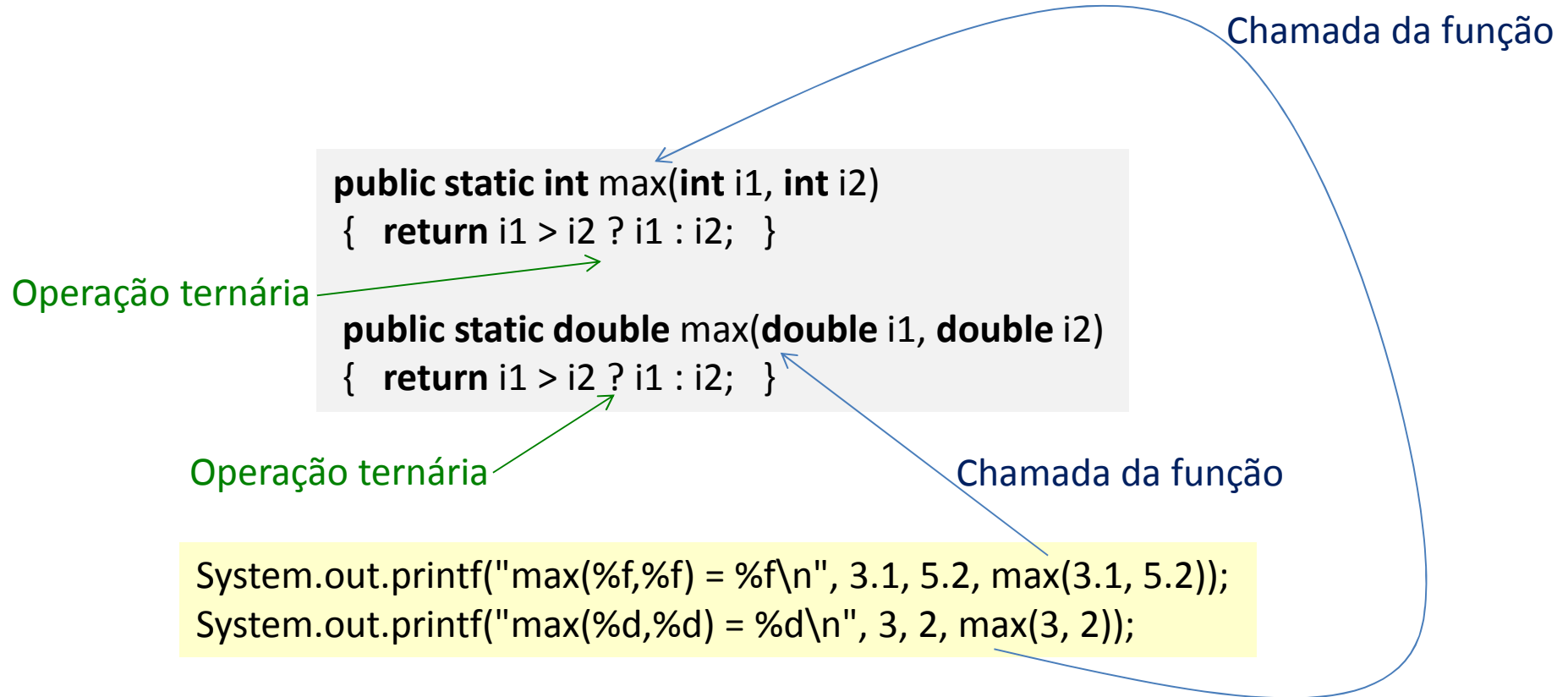
```

Terminal
Valor inteiro: 9
Sistema (2,3,4,5,6,7,8,9) ? 4
1
2
21
conversao esta correta

```

$$2 \times 4^1 + 1 \times 4^0 = 8 + 1 = 9$$

# Sobrecarga de nomes



# Sobrecarga de nomes

```
public static int max(int i1, int i2)
{ return i1 > i2 ? i1 : i2; }
```

```
public static double max(double i1, double i2)
{ return i1 > i2 ? i1 : i2; }
```

```
public static int max(int i1, int i2, int i3)
{ if (i1 > i2 && i1 > i3) return i1;
  else if (i2 > i1 && i2 > i3) return i2;
  return i3;
}
```

Chamada da função

```
System.out.printf("max(%f,%f) = %f\n", 3.1, 5.2, max(3.1, 5.2));
System.out.printf("max(%d,%d) = %d\n", 3, 2, max(3, 2));
System.out.printf("max(%d,%d,%d) = %d\n", 3, 2, 6, max(3, 2, 6));
```

# Sobrecarga de nomes

```
public static int max(int i1, int i2)
{ return i1 > i2 ? i1 : i2; }
```

```
public static double max(double i1, double i2)
{ return i1 > i2 ? i1 : i2; }
```

```
public static int max(int i1, int i2, int i3)
{ if (i1 > i2 && i1 > i3) return i1;
  else if (i2 > i1 && i2 > i3) return i2;
  return i3;
}
```

```
public static char max(int i1, char c)
{ return i1 > (int)c ? (char)i1 : c; }
```

A própria função é selecionada de acordo com o número e tipo de argumentos

Chamada da função

```
System.out.printf("max(%f,%f) = %f\n", 3.1, 5.2, max(3.1, 5.2));
System.out.printf("max(%d,%d) = %d\n", 3, 2, max(3, 2));
System.out.printf("max(%d,%d,%d) = %d\n", 3, 2, 6, max(3, 2, 6));
System.out.printf("max(%d,%c) = %c\n", 48, 'A', max(48, 'A'));
```

# Algumas regras úteis

1. Pode importar a biblioteca `import java.util.*;` em vez de `import java.util.Scanner;` É mais simples para lembrar.
2. Pode declarar um objeto do tipo Scanner `global` e usar este objeto em qualquer função, por exemplo:

```
import java.util.*;  
public class primo  
{  
    static Scanner sc = new Scanner(System.in);  
    public static void main(String[] args)  
    // .....
```

3. Escreva o código mais compacto e compreensível, por exemplo, as funções

```
public static double sqr(double x) {  
    double y;  
    y = x*x;  
    return y;  
}
```

e

```
public static double sqr(double x) { return x*x; }
```

executam a mesma operação mas o último código é melhor porque é mais compacto e não precisa de reservar e libertar memória para a variável y.

# Conclusão

**As funções permitem implementar hierarquia no código do programa e simplificam reutilização de código**

**Uma função pode chamar outras funções**

**As funções podem ser chamadas várias vezes com argumentes diferentes**

**Os objetos declarados numa função são visíveis só dentro desta função**

**Os objetos (variáveis) globais são visíveis dentro de qualquer função**

**Não use variáveis (objetos) globais sem necessidade**

# Exemplos adicionais

## Exemplo A1:

Computação de  
peso de *Hamming*

```
import java.util.*;

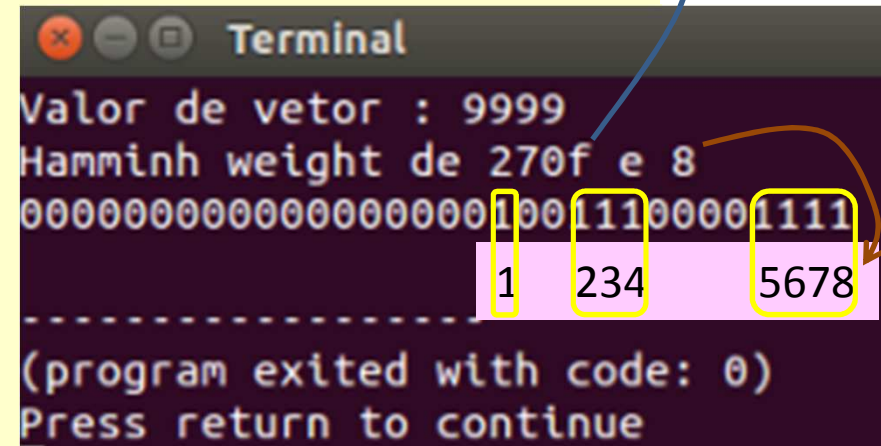
public class hw {

    static Scanner sc = new Scanner(System.in);

    public static void main (String args[]) {
        int V;
        int HW = 0;
        System.out.print("Valor de vetor : ");
        V = sc.nextInt();
        HW(V,HW);
    }

    public static void HW(int V, int HW)
    {
        for (int i = 0; i <= 31; i++)
            HW += (V >> i) & 1;
        System.out.printf("Hamminh weight de %h e %d\n", V,HW);
        for (int i = 31; i >= 0; i--)
            System.out.print((V >> i) & 1);
    }
}
```

$$9999_{10} = 270f_{16}$$



$$270f_{16} = 2 \times 16^3 + 7 \times 16^2 + 0 \times 16^1 + 15 \times 16^0 = 8192 + 1792 + 0 + 15 = 9999_{10}$$



```

import java.util.*;
public class funcoes_booleanas
{
    static Scanner sc = new Scanner(System.in);
    public static void main(String[] args)
    {
        int a[] = { 0, 1, 2, 3 };
        for(char op = '&'; op == '&' || op == '|' || op == '^');
        { System.out.print("operacao ? ");
          op = sc.next().charAt(0);
          AndOrXor(op,a);
        }
        public static void AndOrXor(char operacao,int a[])
        {
            switch(operacao)
            {
                case '^': System.out.println("operacao ^ :");
                    for (int i = 0; i < a.length; i++)
                        System.out.print(((a[i] & 0x2)>>1) + " " + (a[i] & 0x1) + " " + (((a[i] & 0x2)>>1) ^ (a[i] & 0x1)) + '\n');
                    break;
                case '|': System.out.println("operacao | :");
                    for (int i = 0; i < a.length; i++)
                        System.out.print(((a[i] & 0x2)>>1) + " " + (a[i] & 0x1) + " " + (((a[i] & 0x2)>>1) | (a[i] & 0x1)) + '\n');
                    break;
                case '&': System.out.println("operacao & :");
                    for (int i = 0; i < a.length; i++)
                        System.out.print(((a[i] & 0x2)>>1) + " " + (a[i] & 0x1) + " " + (((a[i] & 0x2)>>1) & (a[i] & 0x1)) +
                        '\n');
                    break;
                default: System.out.println("operacao errada");
            }
        }
    }
}

```

```

Terminal
operacao ? &
operacao & :
0 0 0
0 1 0
1 0 0
1 1 1
operacao ? |
operacao | :
0 0 0
0 1 1
1 0 1
1 1 1
operacao ? ^
operacao ^ :
0 0 0
0 1 1
1 0 1
1 1 0
operacao ? !
operacao errada

-----
(program exited with code: 0)
Press return to continue

```

## Exemplo A2:

Imprimir a tabela de verdade para operações booleanas ^, |, & aplicadas a 2 variáveis

## Exemplo A3:

O seguinte exemplo permite gerar dados aleatoriamente e ordenar dados utilizando uma rede de ordenação:

```
import java.util.*;
public class sorting_network
{
    static Random rand = new Random();
    public static void main(String[] args)
    {
        int N = 32; // N pode ser qualquer valor de 2**R, R = 0,1,2,3,4,5,6,7,8,9,10,...
        int a[] = new int[N];
        for(int i = 0; i < a.length; i++)
        {
            a[i] = rand.nextInt(1000);
            System.out.println("a[" + i + "] = " + a[i]);
        }
        System.out.println("-----");
        SN(N,a);
    }

    public static void SN(int N, int a[])
    {
        int tmp;
        for(int k = 0; k < N/2; k++)
        {
            for(int i = 0; i < a.length/2; i++)
            {
                if (a[2*i] < a[2*i+1]) { tmp = a[2*i]; a[2*i] = a[2*i+1]; a[2*i+1] = tmp; }
            }
            for(int i = 0; i < a.length/2-1; i++)
            {
                if (a[2*i+1] < a[2*i+2]) { tmp = a[2*i+1]; a[2*i+1] = a[2*i+2]; a[2*i+2] = tmp; }
            }
            for(int i = 0; i < a.length; i++) { System.out.printf("%5d; ",a[i]);
                if (((i+1)%10) == 0) System.out.println(); }
        }
    }
}
```

## Exemplo A3:

O seguinte exemplo permite gerar dados aleatoriamente e ordenar dados utilizando uma rede de ordenação:

```
Terminal
a[0] = 146
a[1] = 867
a[2] = 979
a[3] = 645
a[4] = 590
a[5] = 887
a[6] = 628
a[7] = 694
a[8] = 284
a[9] = 53
a[10] = 950
a[11] = 327
a[12] = 566
a[13] = 450
a[14] = 336
a[15] = 187
a[16] = 535
a[17] = 581
a[18] = 482
a[19] = 575
a[20] = 500
a[21] = 46
a[22] = 455
a[23] = 53
a[24] = 822
a[25] = 121
a[26] = 742
a[27] = 84
a[28] = 340
a[29] = 762
a[30] = 303
a[31] = 655
-----
979; 950; 887; 867; 822; 762; 742; 694; 655; 645;
628; 590; 581; 575; 566; 535; 500; 482; 455; 450;
340; 336; 327; 303; 284; 187; 146; 121; 84; 53;
53; 46;
-----
(program exited with code: 0)
Press return to continue
```