

# Programação 1

## Aula 3

Valeri Skliarov, Prof. Catedrático

Email: [skl@ua.pt](mailto:skl@ua.pt)

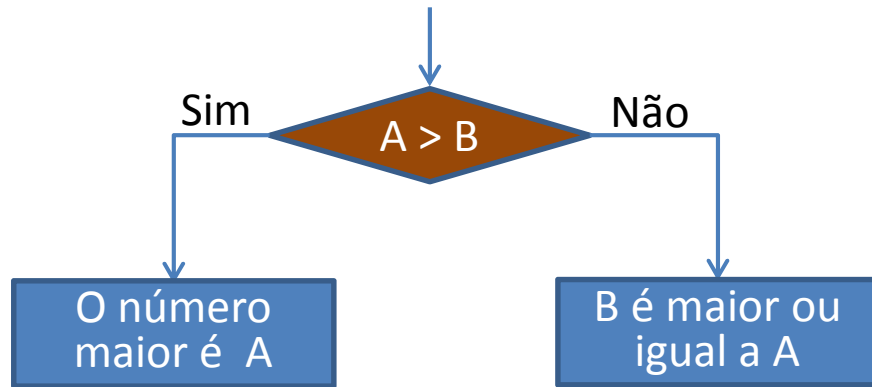
URL: <http://sweet.ua.pt/skl/>

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

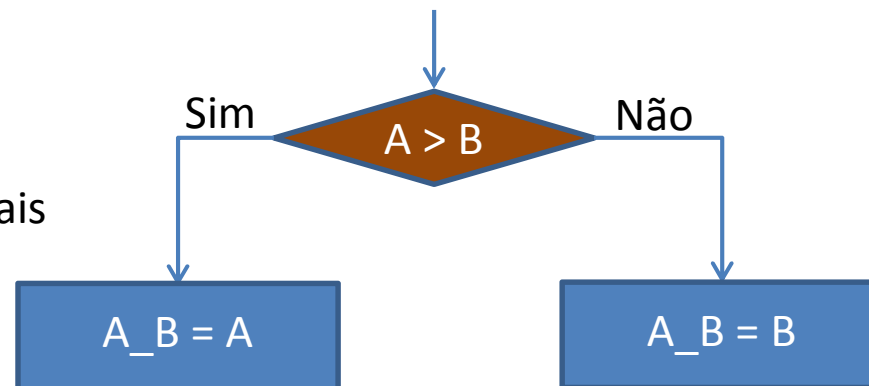
Estas operações  
são iguais

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

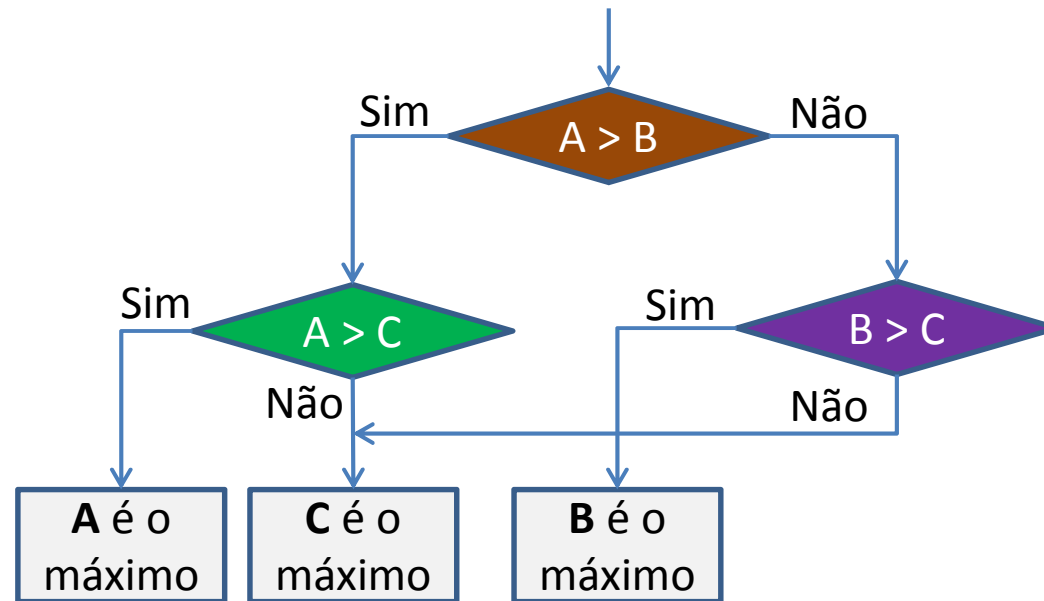
```
A_B = A > B ? A : B;
```

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 ?
```

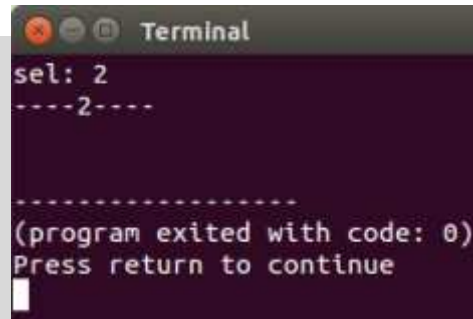
```
    (A > C ? A : C) :
```

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

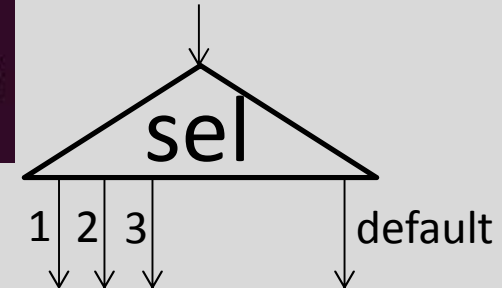
Estas operações  
são iguais

# 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

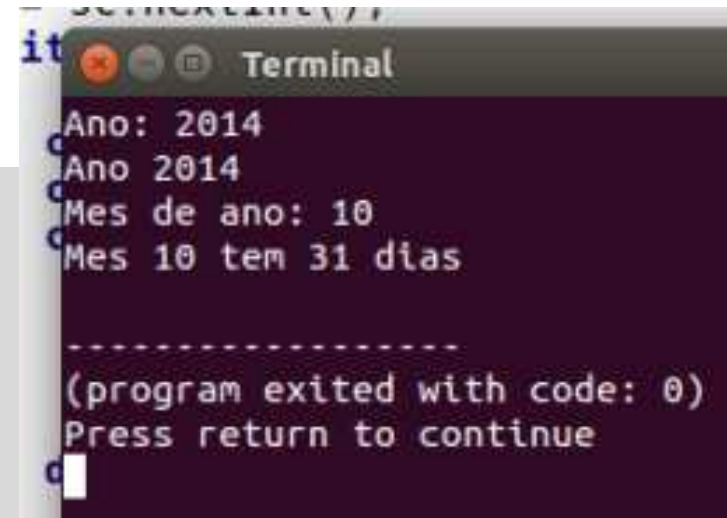
```
double sel;  
System.out.print("sel: ");  
sel = sc.nextInt();  
switch((int)sel)
```

Ok

Só são permitidos valores convertíveis a inteiro

## Exemplo:

```
int A,M;
System.out.print("Ano: ");
A = sc.nextInt();
System.out.println("Ano " + A);
System.out.print("Mês de ano: ");
M = sc.nextInt();
switch(M)
{
    case 1: case 3: case 5: case 7: case 8: case 10: case 12:
        System.out.printf("Mês %d tem 31 dias", M);           break;
    case 4: case 6: case 9: case 11: System.out.printf("Mês %d tem 30 dias", M);           break;
    case 2:
        if( ( A % 4 == 0 ) && !(A % 100 == 0) ) || (A % 400 == 0) )
            System.out.printf("Mês %d tem 29 dias", M);
        else System.out.printf("Mês %d tem 28 dias", M);
                                                                break;
    default: System.out.printf("Mês %d não existe", M);
}
```



```
Terminal
Ano: 2014
Ano 2014
Mes de ano: 10
Mes 10 tem 31 dias

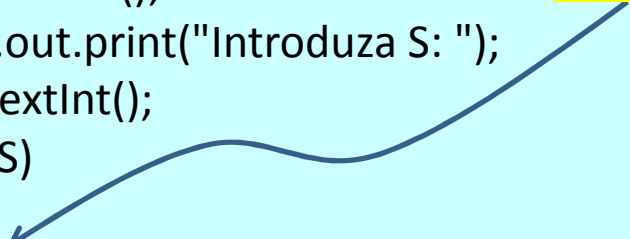
-----
(program exited with code: 0)
Press return to continue
```

January, 2014							February, 2014							March, 2014							April, 2014							May, 2014							June, 2014								
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa		
29	30	31	1	2	3	4	26	27	28	29	30	31	1	23	24	25	26	27	28	1	30	31	1	2	3	4	5	27	28	29	30	1	2	3	25	26	27	28	29	30	31		
5	6	7	8	9	10	11	2	3	4	5	6	7	8	9	10	11	12	13	14	15	6	7	8	9	10	11	12	4	5	6	7	8	9	10	8	9	10	11	12	13	14		
12	13	14	15	16	17	18	9	10	11	12	13	14	15	16	17	18	19	20	21	22	13	14	15	16	17	18	19	11	12	13	14	15	16	17	15	16	17	18	19	20	21		
19	20	21	22	23	24	25	16	17	18	19	20	21	22	23	24	25	26	27	28	29	20	21	22	23	24	25	26	18	19	20	21	22	23	24	22	23	24	25	26	27	28		
26	27	28	29	30	31	1	23	24	25	26	27	28	1	30	31	1	2	3	4	5	27	28	29	30	1	2	3	25	26	27	28	29	30	31	29	30	1	2	3	4	5	6	7

# Erros potenciais

```
int A, B, C, D, S;  
System.out.print("Introduza A: ");  
A = sc.nextInt();  
System.out.print("Introduza B: ");  
B = sc.nextInt();  
System.out.print("Introduza C: ");  
C = sc.nextInt();  
System.out.print("Introduza D: ");  
D = sc.nextInt();  
System.out.print("Introduza S: ");  
S = sc.nextInt();  
switch(S)  
{  
    case A: System.out.println("S = A"); break;  
    case B: System.out.println("S = B"); break;  
    case C: System.out.println("S = C"); break;  
    case D: System.out.println("S = D"); break;  
    default: System.out.println("S != A e S != B e S != C e S != D");  
}
```

ERRO!!!  
Tem que ser constante



# **Programação 1**

## **Aula 3**



- Estruturas de controlo – repetição
- Operadores aritméticos unários
- Instrução de atribuição com operação
- Instrução repetitiva **while** e **do...while**
- Instrução repetitiva **for**
- Instruções de salto **break** e **continue**

- Para além da execução condicional de instruções, por vezes existe a necessidade de executar instruções repetidamente.
- A um conjunto de instruções que são executadas repetidamente designamos por **ciclo**.
- Um **ciclo** é constituído por uma estrutura de controlo que determina quantas vezes as instruções vão ser repetidas.
- As estruturas de controlo podem ser dos tipos **while**, **do...while** e **for**.
- Normalmente utilizamos as estruturas do tipo condicional quando o número de iterações é desconhecido e as estruturas do tipo contador quando sabemos à partida o número de iterações.

# Operadores aritméticos unários

- incremento de 1: ++ (++x, x++)
- decremento de 1: -- (--x, x--)
- Os operadores de incremento e decremento atualizam o valor de uma variável com mais ou menos uma unidade.
- Colocados antes são pré-incremento e pré-decremento. Neste caso a variável é primeiro alterada antes de ser usada.
  - `y = ++x; // equivalente a: x = x + 1; y = x;`
- Colocados depois são pós-incremento e pós-decremento e neste caso a variável é primeiro usada na expressão onde está inserida e depois atualizada.
  - `y = x++; // equivalente a: y = x; x = x + 1;`

# Operadores aritméticos unários

## Exemplos:

```
int a=5, b=6, c;
c = a++ + b; // ???
```

c = 11, a = 6

```
int a=5, b=6, c;
c = a+++ b; // ???
```

```
int a=5, b=6, c;
c = a++ - b; // ???
```

c = -1, a = 6

```
int a=5, b=6, c;
c = ++a + b; // ???
```

c = 12, a = 6

```
int a=5, b=6, c;
c = ++a - b; // ???
```

c = 0, a = 6

```
int a=5, b=6, c;
c = ++a - b++; // ???
```

c = 0, a = 6, b = 7

c = 0	a = 6	b = 7
c = 0	a = 7	b = 7

```
int a=5, b=6, c;
c = ++a - b++; // ???
System.out.printf(" c = %d      a = %d      b = %d\n",c,a,b );
System.out.printf(" c = %d      a = %d      b = %d\n",c,++a,b++ );
```

```
int a=9, b=17;
double c;
c = Math.sqrt(a++) + Math.sqrt(--b);
System.out.printf(" c = %f      a = %d      b = %d\n",c,a,b );

a=9; b=17;
c = Math.sqrt(++a) + Math.sqrt(b--);
System.out.printf(" c = %f      a = %d      b = %d\n",c,a,b );
```

c = 7.000000	a = 10	b = 16
c = 7.285383	a = 10	b = 16

```
int a=5, b=6, c;
c = ++a - b++; // ???
System.out.printf(" c = %d      a = %d      b = %d\n",c,a,b );
```

# Instrução de atribuição com operação

- É comum usar uma versão compacta do operador de atribuição (=) onde este é precedido de uma operação (por exemplo +=, -=, \*=, /=, %=, ...).
- A instrução resultante é equivalente a uma instrução normal de atribuição em que a mesma variável aparece em ambos os lados do operador =.
- A importância desta notação tem a ver com a simplificação do código e com a clareza da operação a realizar.
  - `int x, y, z;`
  - `...`
  - `y += 5;`     *// equivalente a `y = y + 5;`*
  - `z *= 5 + x;`     *// equivalente a `z = z * (5 + x);`*
  - `y += ++x;`     *// `x = x + 1;` `y = y + x;`*

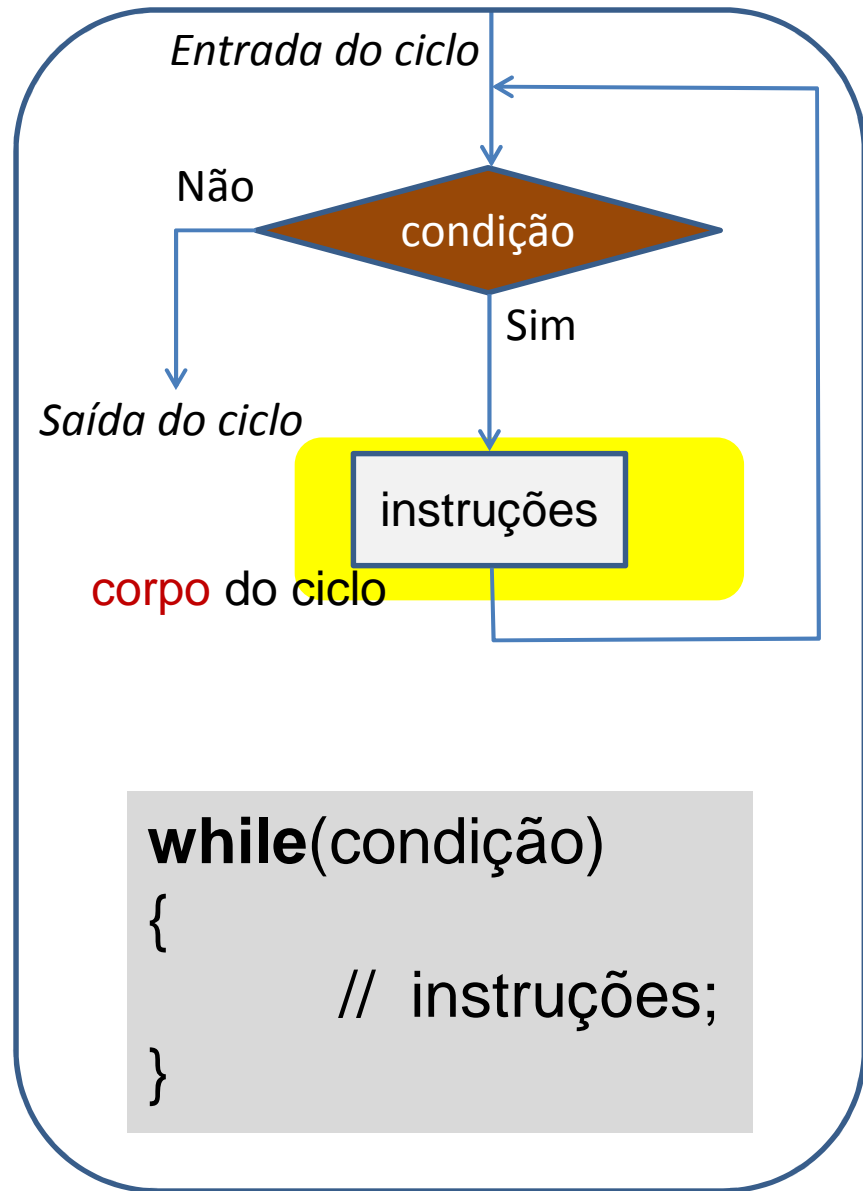
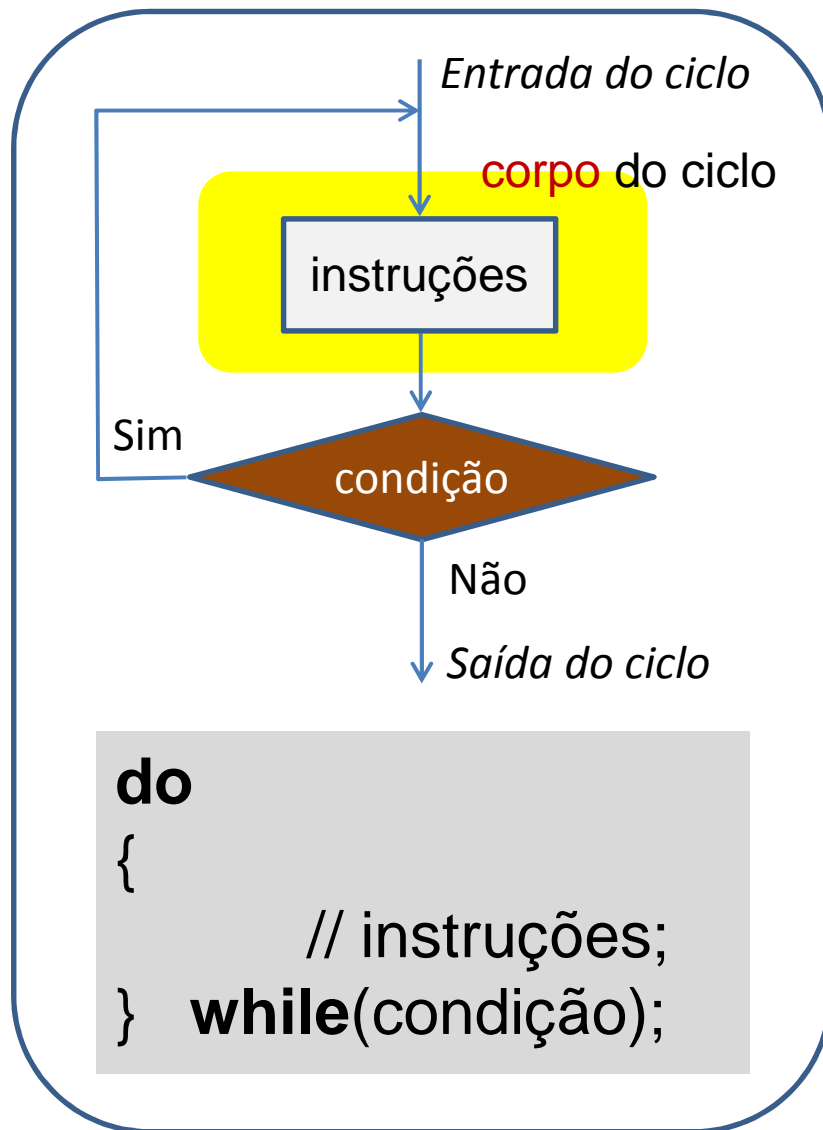
# Instrução repetitiva **while** e **do...while**

```
do  
{  
    // instruções;  
} while(condição);
```

```
while(condição)  
{  
    // instruções;  
}
```

- A sequência de instruções colocadas no **corpo** do ciclo são executadas enquanto a **condição** for verdadeira.
- Quando a condição for falsa, o ciclo termina e o programa continua a executar o que se **seguir**.
- A diferença principal entre as duas instruções repetitivas reside no facto de no ciclo **do ... while** a sequência de instruções é executada pelo menos uma vez.
- Muito cuidado na definição da **condição**...

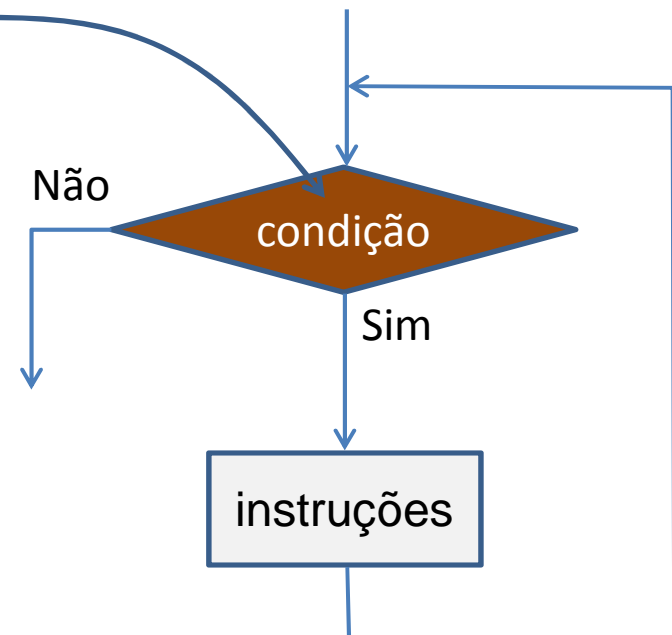
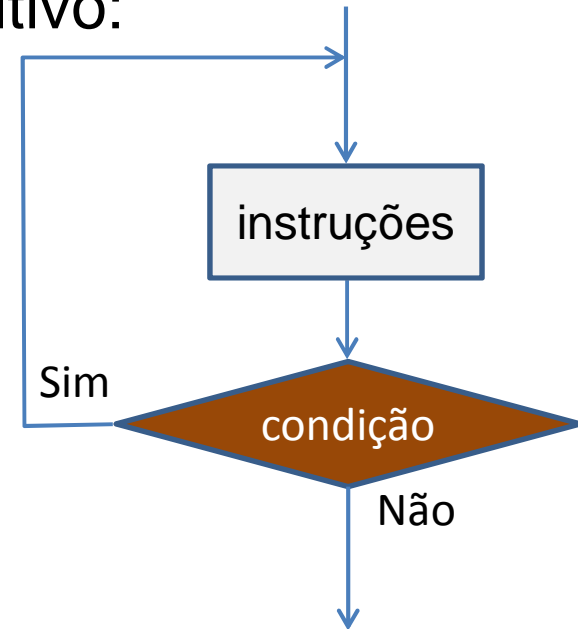
# Ciclos



## Exemplo de leitura de um valor inteiro positivo:

```
int x, cont = 0;  
do {  
    System.out.print("Um valor inteiro positivo: ");  
    x = sc.nextInt();  
    cont++;  
} while(x <= 0);  
System.out.printf("Valor %d lido em %d tentativas\n",x,cont);
```

```
int x = -1, cont = 0; // Atenção à inicialização de x  
while(x <= 0) {  
    System.out.print("Um valor inteiro positivo: ");  
    x = sc.nextInt();  
    cont++; }  
System.out.printf("Valor %d lido em %d tentativas\n",x,cont);
```





# Instrução repetitiva **for**

```
for(inicialização ; condição ; atualização)
```

```
{
```

```
    // instruções;
```

```
}
```

- A inicialização é executada em primeiro lugar e apenas uma vez.
- A condição é avaliada no início de todos os ciclos e as instruções são executadas enquanto a condição for verdadeira.
- A parte da atualização é feita no final de todas as iterações.
- Em geral, a função da inicialização e da atualização é manipular variáveis de contagem utilizadas dentro do ciclo.

# Instrução repetitiva for

## Exemplos:

```
for(int i = 0; i < 5; i++)  
    System.out.printf("i = %d\n",i);
```

```
i = 0  
i = 1  
i = 2  
i = 3  
i = 4
```

```
for(int i = 0; i < 5; i++)  
    System.out.printf("i = %d\n",i);  
  
System.out.printf("i = %d\n",i);
```

IncDec.java:27: error: cannot find symbol

System.out.printf("i = %d\n",i);

^

symbol: variable i

location: class IncDec

1 error

Compilation failed.

Erro !!!

```
int i;  
for(i = 0; i < 5; i++)  
    System.out.printf("i = %d\n",i);  
  
System.out.printf("i = %d\n",i);
```

```
i = 0  
i = 1  
i = 2  
i = 3  
i = 4  
i = 5
```

Ok

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

- Podemos terminar a execução de um bloco de instruções com duas instruções especiais: **break** e **continue**.
- A instrução **break** permite a saída imediata do bloco de código que está a ser executado. É usada normalmente no **switch** e em estruturas de repetição, terminando-as.
- A instrução **continue** permite terminar a execução do bloco de instruções dentro de um ciclo, forçando a passagem para a iteração seguinte (não termina o ciclo).
- A aplicação destas instruções em conjunto com os ciclos permite reduzir a complexidade dos mesmos, aumentando clareza e legibilidade do código.

# 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--)

**Exemplo 1:** Escreva um programa que leia uma série de números inteiros. Quando for introduzido um número negativo, o programa deve escrever quantos números foram introduzidos e terminar

```
int x;    int n = 0;
do        {
    System.out.print("Introduza um numero: ");
    x = sc.nextInt(); n++;
    } while(x >= 0);
System.out.println("n = "+n);
```

**do ... while**

```
int x = 0; int n = 0;
for(;x>=0;) {
    System.out.print("Introduza um numero: ");
    x = sc.nextInt(); n++;
}
System.out.println("n = "+n);
```

**for**

```
int x = 0; int n = 0;
while(x >= 0) {
    System.out.print("Introduza um numero: ");
    x = sc.nextInt(); n++;
};
System.out.println("n = "+n);
}
```

**while**

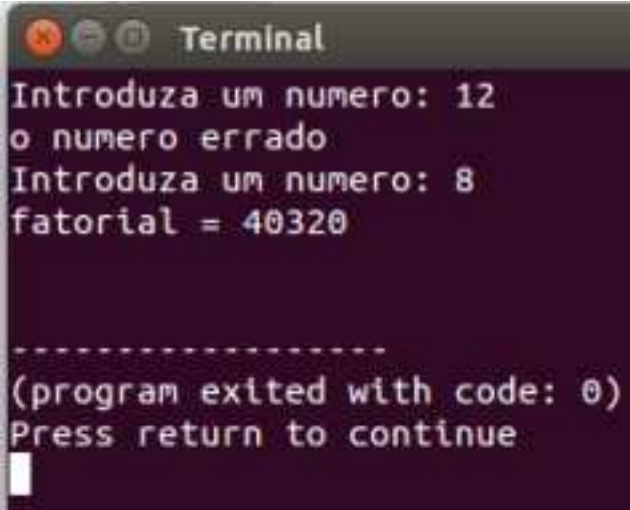
```
int x; int n = 0;
for(x = 0;x>=0;) {
    System.out.print("Introduza um numero: ");
    x = sc.nextInt(); n++;
}
System.out.println("n = "+n);
```

**for**

Embora qualquer dos três ciclos pode ser usado, provavelmente **do ... while** é o melhor porque **do ... while** é o mais natural para a tarefa considerada

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

```
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);
```

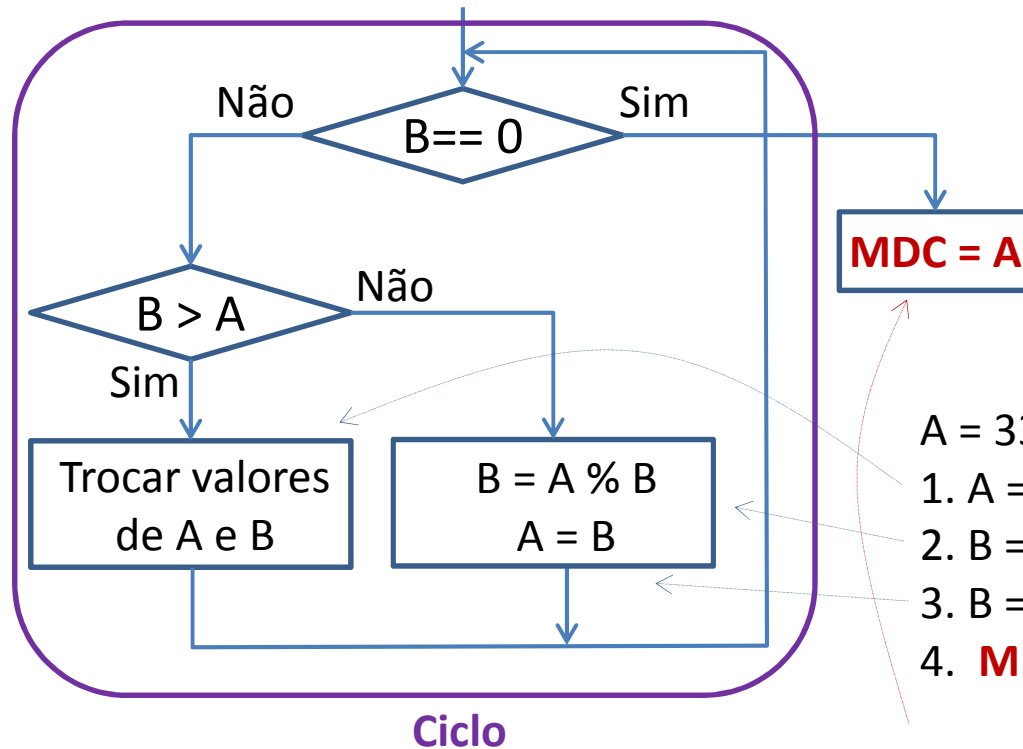


```
Terminal
Introduza um numero: 12
o numero errado
Introduza um numero: 8
fatorial = 40320

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

Para este exemplo ciclo **do ... while** é o melhor para verificar dados de entrada e ciclo **for** é o melhor para calcular o fatorial

**Exemplo 3:** Escreva um programa que leia dois números inteiros e determine o seu divisor máximo comum (MDC) através do algoritmo de Euclides.



A = 33; B = 77

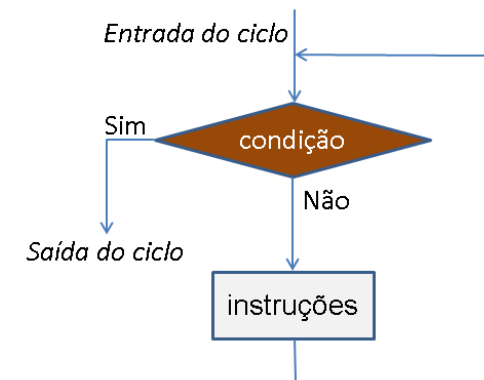
1. A = 77; B = 33 // trocar valores de A e B

2. B = A % B = 11; A = 33

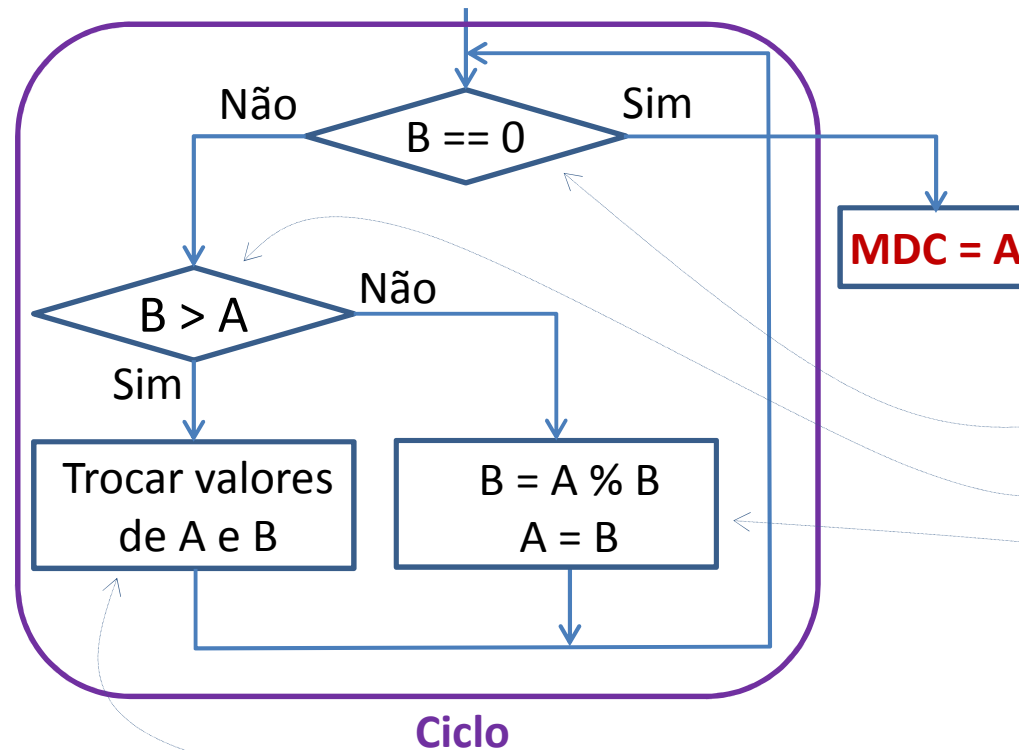
3. B = A % B = 0; A = 11.

4. MDC = A = 11.

Para este exemplo, provavelmente, ciclo **while** é o melhor



**Exemplo 3:** Escreva um programa que leia dois números inteiros e determine o seu divisor máximo comum (MDC) através do algoritmo de Euclides.



```
int tmp;
int A,B;
System.out.print("Introduza A: ");
A = sc.nextInt();
System.out.print("Introduza B: ");
B = sc.nextInt();
while (B>0)
{ if (B > A) { tmp=A; A=B; B=tmp;}
  else      { tmp=B; B=A%B; A=tmp;}
}
System.out.println("MDC = "+A);
```

$A = 33; B = 77$

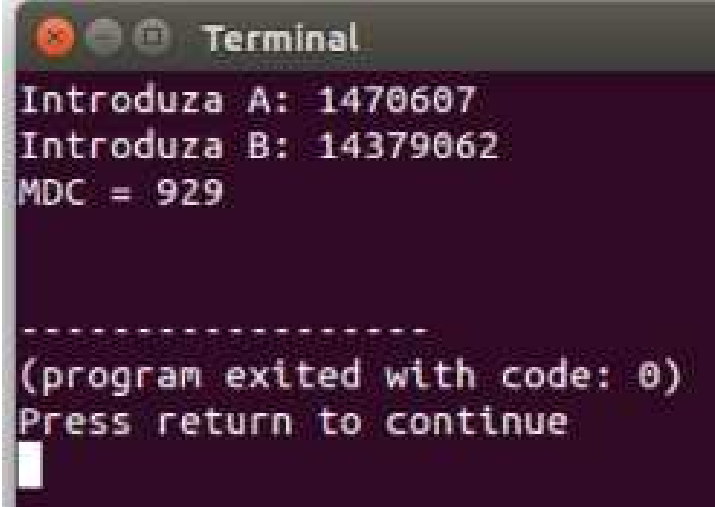
1.  $A = 77; B = 33$  // trocar valores de A e B
2.  $B = A \% B = 11; A = 33$
3.  $B = A \% B = 0; A = 11$ .
4. **MDC = A = 11.**



Chavetas não são obrigatórias:  
pode remover ou não

```
int tmp;  
int A,B;  
System.out.print("Introduza A: ");  
A = sc.nextInt();  
System.out.print("Introduza B: ");  
B = sc.nextInt();  
while (B>0)  
{ if (B > A) { tmp=A; A=B; B=tmp;}  
  else      { tmp=B; B=A%B; A=tmp;}  
}  
System.out.println("MDC = "+A);
```

```
Welcome to DrJava.  
> run MDC  
Introduza A: 49  
Introduza B: 77  
MDC = 7  
>
```



```
Terminal  
Introduza A: 1470607  
Introduza B: 14379062  
MDC = 929  
  
-----  
(program exited with code: 0)  
Press return to continue
```

## **Exemplo 4** (Exemplo de leitura de um valor inteiro positivo:)

```
int x, cont = 0;
do {
    System.out.print("Um valor inteiro positivo: ");
    x = sc.nextInt();
    cont++;
} while(x <= 0);
System.out.printf("Valor %d lido em %d tentativas\n",x,cont);
```

**do ... while**

```
int x = -1, cont = 0; // Atenção à inicialização de x
while(x <= 0) {
    System.out.print("Um valor inteiro positivo: ");
    x = sc.nextInt();
    cont++; }
System.out.printf("Valor %d lido em %d tentativas\n",x,cont);
```

**while**

```
int x, cont;
for (x = -1, cont = 0; x <= 0; cont++)
    { System.out.print("Um valor inteiro positivo: ");
      x = sc.nextInt(); }
System.out.printf("Valor %d lido em %d tentativas\n",x,cont);
```

**for**

## Exemplo 5:

```
int x, cont = 0;
do {
    System.out.print("Um valor inteiro positivo: ");
    x = sc.nextInt();
    cont++;
    if(cont >= 10) //depois de 10 tentativas, termina o ciclo
        break;
} while(x <= 0);
if(x > 0) System.out.printf("Valor %d lido em %d tentativas\n",x,cont);
else      System.out.printf("Ultrapassadas 10 tentativas\n");
```

```
Um valor inteiro positivo: -3
Um valor inteiro positivo: 5
Valor 5 lido em 2 tentativas
```

## Exemplo 6:

```
int i, n, soma = 0;
do {
    System.out.print("Valor de N [1 ... 99]: ");
    n = sc.nextInt();
    } while(n < 1 || n > 100);
for(i = 1 ; i <= n ; i++){
    // se numero par avança para a iteração seguinte
    if(i % 2 == 0) continue;
    soma += i;
}
System.out.printf("A soma dos impares é %d\n", soma);
```

```
Valor de N [1 ... 99]: 120
Valor de N [1 ... 99]: 111
Valor de N [1 ... 99]: -6
Valor de N [1 ... 99]: 5
A soma dos impares e 9
```

## Exemplo 6: o mesmo código sem instrução **continue**

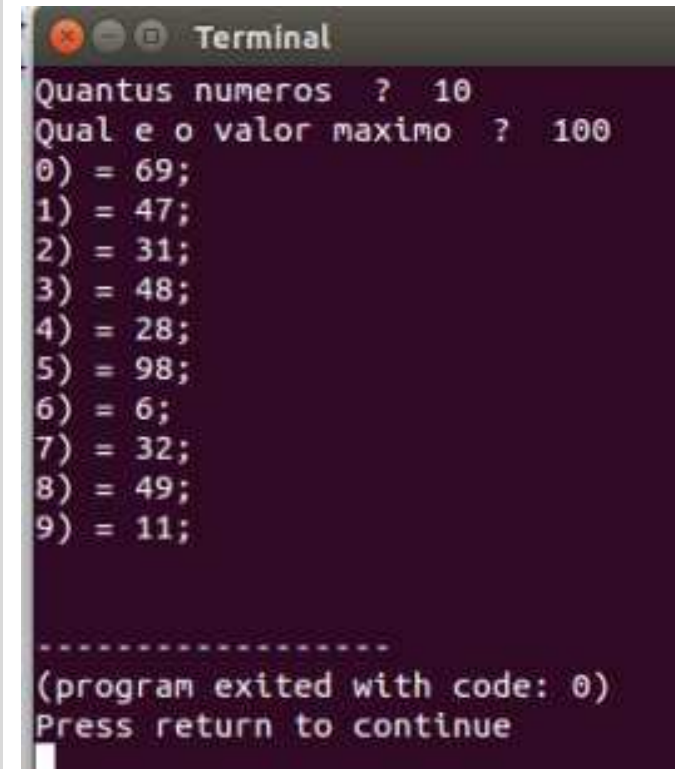
```
int i, n, soma = 0;
do {
    System.out.print("Valor de N [1 ... 99]: ");
    n = sc.nextInt();
} while(n < 1 || n > 100);
for(i = 1 ; i <= n ; i++)
    if(i % 2 != 0) soma += i;
System.out.printf("A soma dos ímpares é %d\n", soma);
```

```
int i, n, soma = 0;
do {
    System.out.print("Valor de N [1 ... 99]: ");
    n = sc.nextInt();
} while(n < 1 || n > 100);
for(i = 1 ; i <= n ; i++) {
    // se numero par avança para a iteração seguinte
    if(i % 2 == 0) continue;
    soma += i;
}
System.out.printf("A soma dos impares é %d\n", soma);
```

Ciclo **for** com instrução **continue**

## Exemplo 7: Gerar N números inteiros entre 0 e M-1 aleatoriamente

```
import java.util.*;
public class inteiros_aleatorios
{
    static Random rand = new Random();
    static Scanner sc = new Scanner(System.in);
    public static void main(String[] args)
    {
        int N,M;
        System.out.print("Quantus números ? ");
        N = sc.nextInt();
        System.out.print("Qual é o valor máximo ? ");
        M = sc.nextInt();
        for(int i = 0; i < N; i++)
            System.out.println(i+" = "+rand.nextInt(M)+" ");
    }
}
```

A screenshot of a terminal window titled "Terminal". It shows the execution of a Java program. The user is prompted to enter the number of integers (N) and the maximum value (M). The program then outputs 10 random integers. At the bottom, it shows the program exited with code 0 and a prompt to press return to continue.

```
Terminal
Quantus numeros ? 10
Qual e o valor maximo ? 100
0) = 69;
1) = 47;
2) = 31;
3) = 48;
4) = 28;
5) = 98;
6) = 6;
7) = 32;
8) = 49;
9) = 11;

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

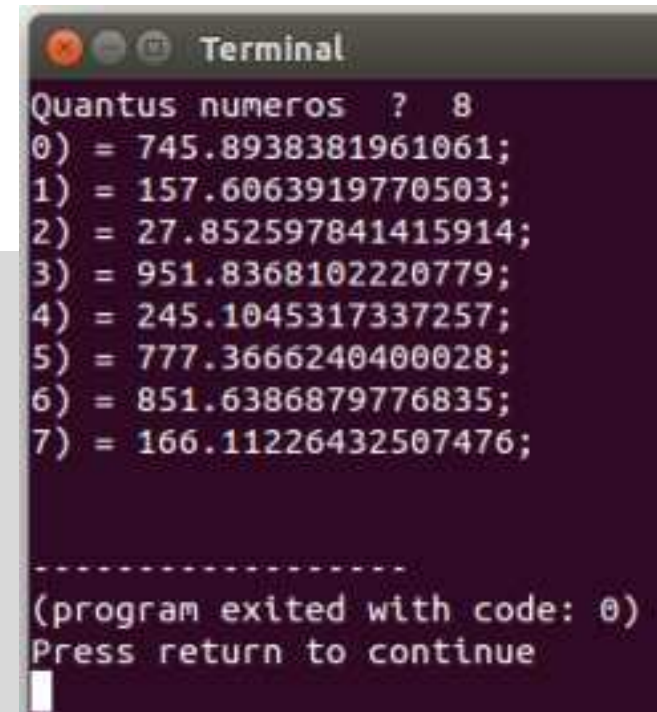
```
for(int i = 0; i < N; i++)
    System.out.printf("%d) = %d;\n", i, rand.nextInt(M));
```

## Exemplo 8:

Gerar N números reais aleatoriamente

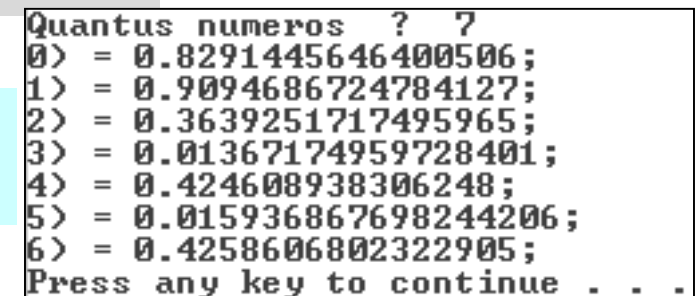
```
import java.util.*;
public class inteiros_reais
{
    static Random rand = new Random();
    static Scanner sc = new Scanner(System.in);
    public static void main(String[] args)
    {   int N;
        System.out.print("Quantus números ? ");
        N = sc.nextInt();
        for(int i = 0; i < N; i++)
            System.out.println(i+" = "+rand.nextDouble()*1000+"; ");
    }
}
```

```
for(int i = 0; i < N; i++)
    System.out.println(i+" = "+rand.nextDouble()+"; ");
```

A terminal window titled "Terminal" with a dark background. It shows the execution of a Java program. The prompt "Quantus numeros ?" is followed by the input "8". The program then outputs eight lines of random numbers, each indexed from 0 to 7. After the output, a dashed line separates it from the exit message "(program exited with code: 0)" and the instruction "Press return to continue".

```
Terminal
Quantus numeros ? 8
0) = 745.8938381961061;
1) = 157.6063919770503;
2) = 27.852597841415914;
3) = 951.8368102220779;
4) = 245.1045317337257;
5) = 777.3666240400028;
6) = 851.6386879776835;
7) = 166.11226432507476;

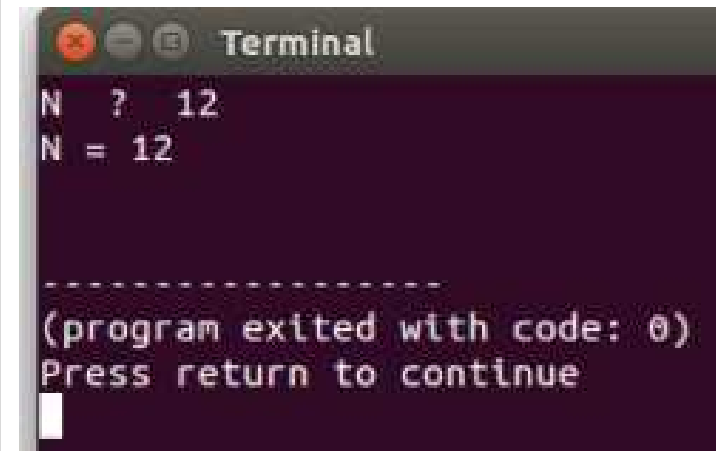
-----
(program exited with code: 0)
Press return to continue
```

A terminal window showing the execution of the same Java program with the input "7". It outputs seven lines of random numbers indexed from 0 to 6. The prompt "Press any key to continue" is visible at the bottom.

```
Quantus numeros ? 7
0> = 0.8291445646400506;
1> = 0.9094686724784127;
2> = 0.3639251717495965;
3> = 0.01367174959728401;
4> = 0.424608938306248;
5> = 0.015936867698244206;
6> = 0.4258606802322905;
Press any key to continue . . .
```

**Exemplo 9:** Entrar um valor inteiro entre 10 e 20 utilizando ciclo for e repetir a entrada se o valor for fora do intervalo 10,...,20

```
import java.util.*;
public class entrada_for
{
    static Scanner sc = new Scanner(System.in);
    public static void main(String[] args)
    {   int N;
        for(;;)
        {
            System.out.print("N ? ");
            N = sc.nextInt();
            if(N >= 10 && N <= 20) break;
        }
        System.out.println("N = "+N);
    }
}
```



```
Terminal
N ? 12
N = 12

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