

Programação 1

Aula 3

Valeri Skliarov, Prof. Catedrático

Email: skl@ua.pt

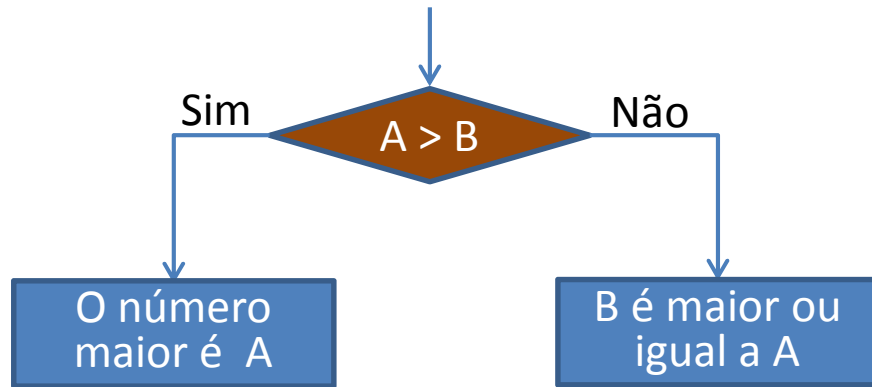
URL: <http://sweet.ua.pt/skl/>

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

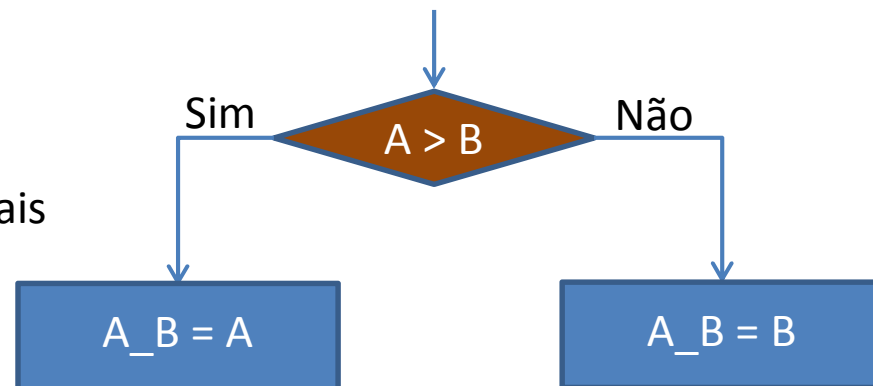
```
System.out.println(A > B ? "O número maior é A" : "B é maior ou igual a A");
```

Estas operações
são iguais

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

Estas operações são iguais

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



Exemplos:

```
public static double max(double x, double y) {  
    return (x>y) ? x : y;  
}
```

```
public static double min(double x, double y) {  
    return (x<y) ? x : y;  
}
```

```
public static int max(int x, int y) {  
    return (x>y) ? x : y;  
}
```

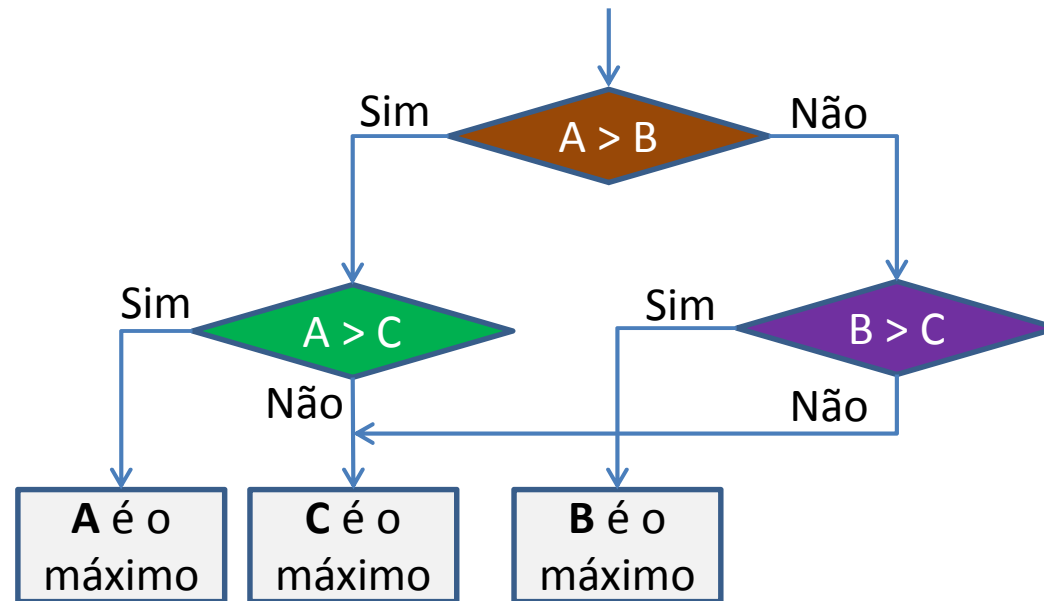
```
public static int min(int x, int y) {  
    return (x<y) ? x : y;  
}
```

```
public static void print_max(int x, int y) {  
    System.out.println((x>y) ? x : y);  
}
```

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

```
print_max(10,20);
```

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 ");
```

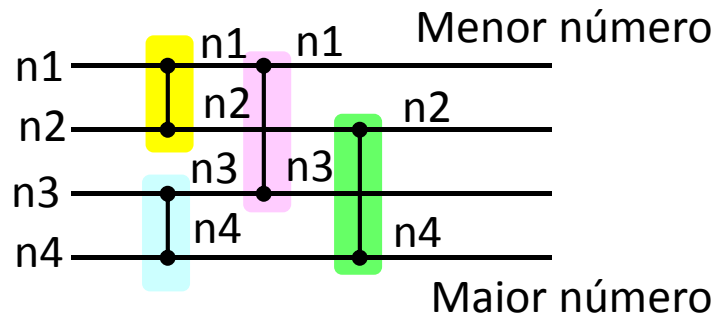
```
System.out.println(A > B ?
```

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

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

Estas operações
são iguais

Instrução if



```
if (n1 > n2) { tmp = n1; n1 = n2; n2 = tmp; }
```

```
if (n3 > n4) { tmp = n3; n3 = n4; n4 = tmp; }
```

```
if (n1 > n3) { tmp = n1; n1 = n3; n3 = tmp; }
```

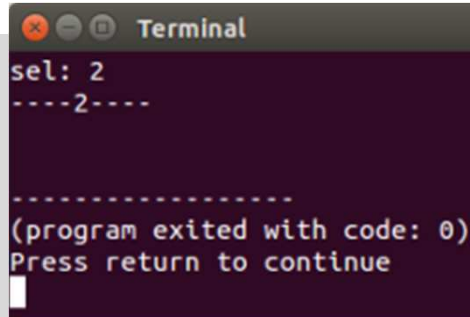
```
if (n2 > n4) { tmp = n2; n2 = n4; n4 = tmp; }
```

```
n1 ??? 23
n1 ??? 11
n1 ??? 57
n1 ??? 34
O numero menor e 11
O numero maior e 57
Press any key to continue . . . -
```

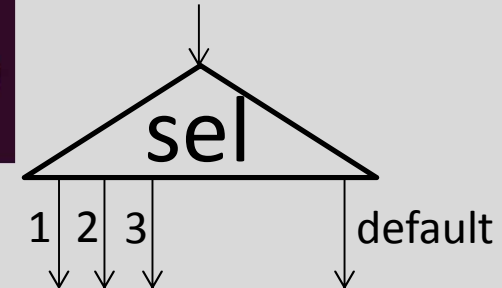
```
import java.util.*;
public class max_min {
    public static void main(String[] args) {
        Scanner ob = new Scanner(System.in);
        int n1,n2,n3,n4;
        System.out.print("n1 ??? ");
        n1 = ob.nextInt();
        System.out.print("n1 ??? ");
        n2 = ob.nextInt();
        System.out.print("n1 ??? ");
        n3 = ob.nextInt();
        System.out.print("n1 ??? ");
        n4 = ob.nextInt();
        int tmp;
        if (n1 > n2) { tmp = n1; n1 = n2; n2 = tmp; }
        if (n3 > n4) { tmp = n3; n3 = n4; n4 = tmp; }
        if (n1 > n3) { tmp = n1; n1 = n3; n3 = tmp; }
        System.out.println("O número menor é " + n1);
        if (n2 > n4) { tmp = n2; n2 = n4; n4 = tmp; }
        System.out.println("O número maior é " + n4);
        ob.close();
    }
}
```

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);
}
```

```
it
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

```
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++ );
```

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

```
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;`*

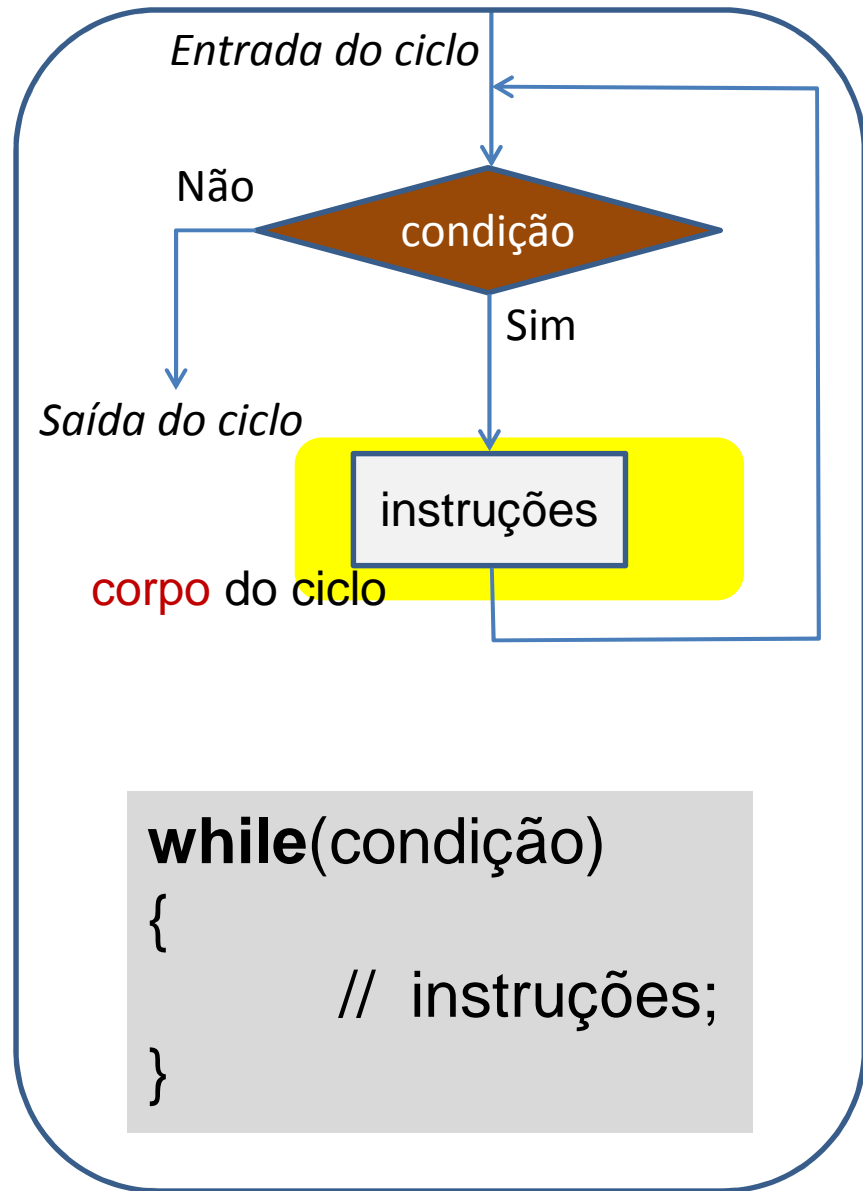
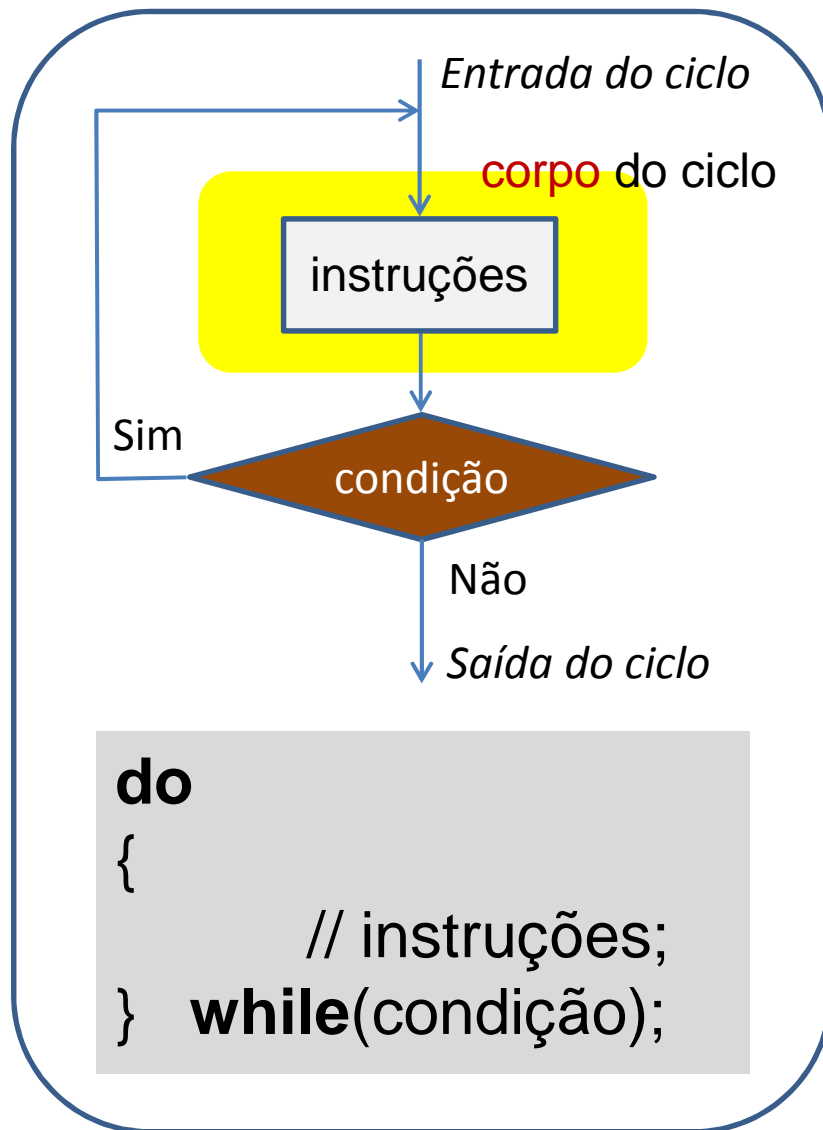
Instrução repetitiva **while** e **do...while**

```
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**...

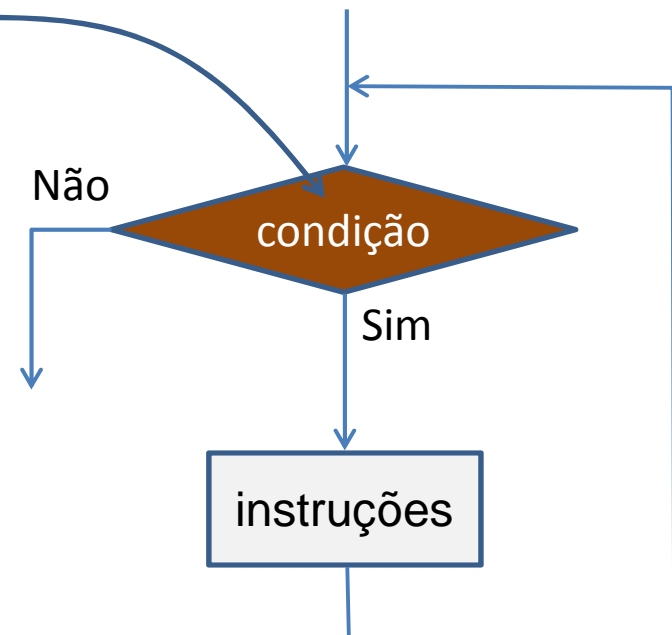
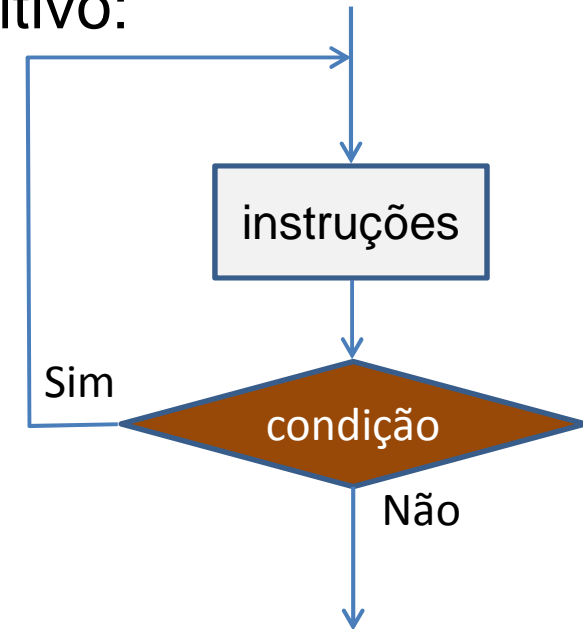
Ciclos



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

Instrução repetitiva for

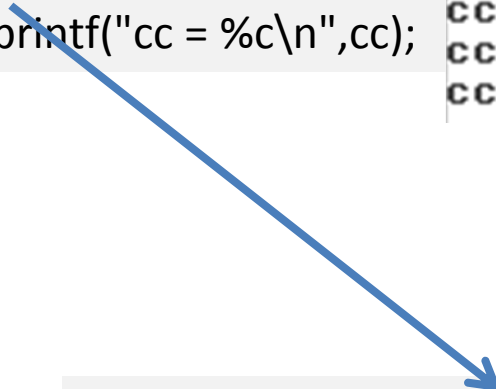
Exemplos:

```
for(double j = 5.; j < 20; j += 4.5)
    System.out.printf("j = %.2f\n",j);
```

```
j. = 5.00
j. = 9.50
j. = 14.00
j. = 18.50
```

```
for(char cc = 'A'; cc < 'H'; ++cc)
    System.out.printf("cc = %c\n",cc);
```

```
cc = A
cc = B
cc = C
cc = D
cc = E
cc = F
cc = G
```



```
for(char cc = 'A'; cc < 'h'; ++cc)
    System.out.printf("cc = %c\n",cc);
```

```
cc = A
cc = B
cc = C
cc = D
cc = E
cc = F
cc = G
cc = H
cc = I
cc = J
cc = K
cc = L
cc = M
cc = N
cc = O
cc = P
cc = Q
cc = R
cc = S
cc = T
cc = U
cc = V
cc = W
cc = X
cc = Y
cc = Z
cc = [
cc = \
cc = ^
cc = _
cc = a
cc = b
cc = c
cc = d
cc = e
cc = f
cc = g
```

Instrução repetitiva for

Exemplos:

```
char cc; int k;  
for(cc = 'H', k = 0; cc >= 'A'; cc--, k++)  
{  
    System.out.printf("k = %d    ", k);  
    System.out.printf("CC = %c\n", cc);  
}
```

k = 0	CC = H
k = 1	CC = G
k = 2	CC = F
k = 3	CC = E
k = 4	CC = D
k = 5	CC = C
k = 6	CC = B
k = 7	CC = A

for(char cc = 'H', int k = 0; cc < 'A'; cc--, k++) // ERRO !!!

```
for(cc = 'H', k = 0; (cc >= 'A' && k < 4); cc--, k++)  
{  
    System.out.printf("k = %d    ", k);  
    System.out.printf("CC = %c\n", cc);  
}
```

k = 0	CC = H
k = 1	CC = G
k = 2	CC = F
k = 3	CC = E

```
for(int k = 0; k < 0; k++)
```

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

Não faz nada

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

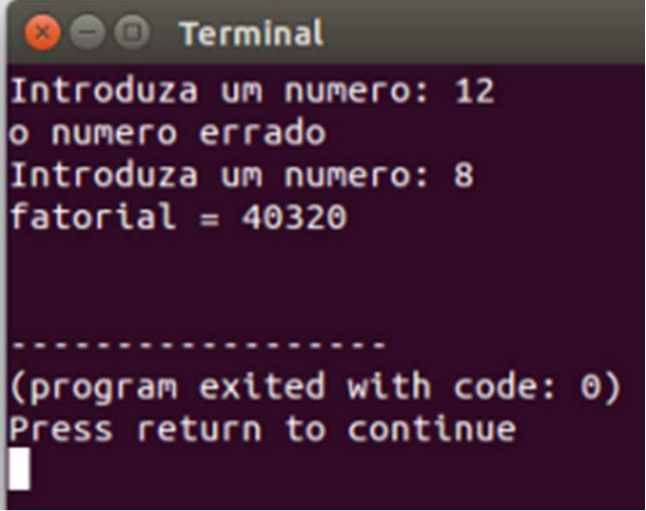
```
int x; int n = 0;
for(x = 0;x>=0;) {
    System.out.print("Introduza um numero: ");
    x = sc.nextInt(); n++;
}
System.out.println("n = "+n);
```

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);
```

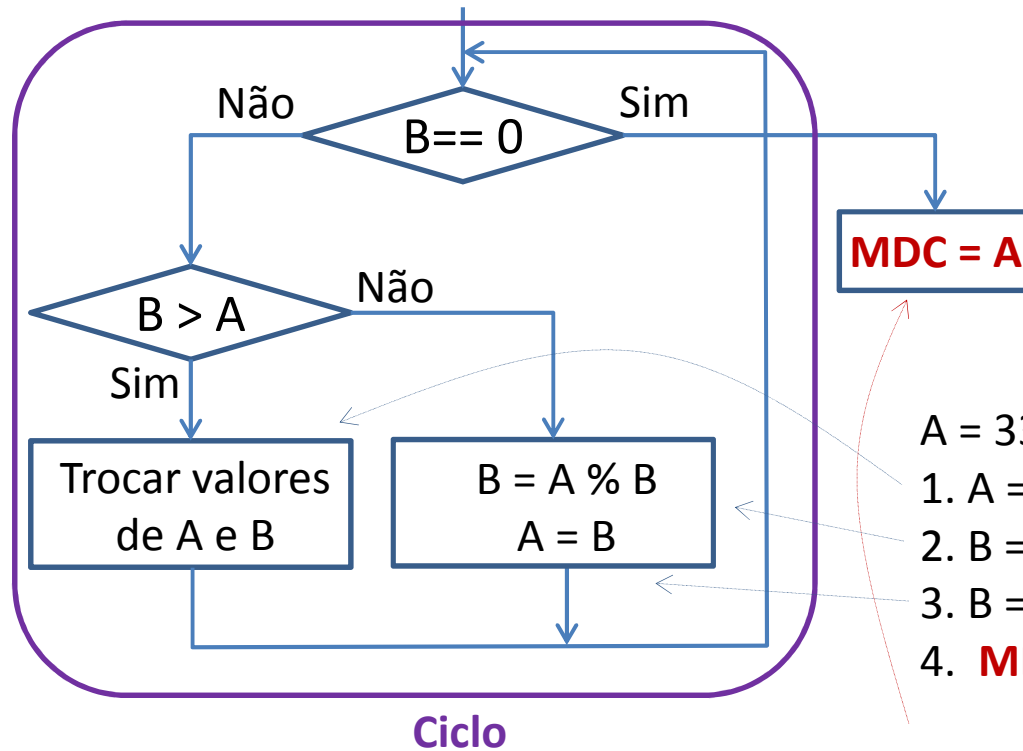
A screenshot of a terminal window titled "Terminal". It shows the execution of a Java program. The user enters "12", and the program outputs "o numero errado". The user then enters "8", and the program outputs "fatorial = 40320". Below this, there is a separator line of dashes, followed by the text "(program exited with code: 0)" and "Press return to continue".

```
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

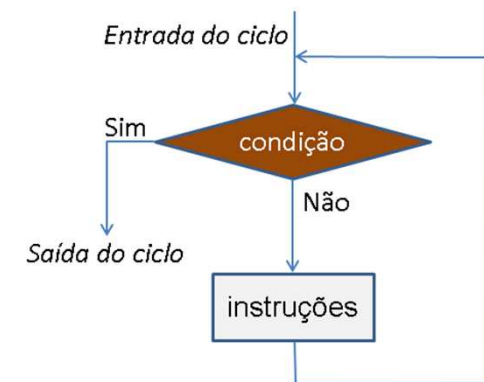
1. A = 77; B = 33 // trocar valores de A e B

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

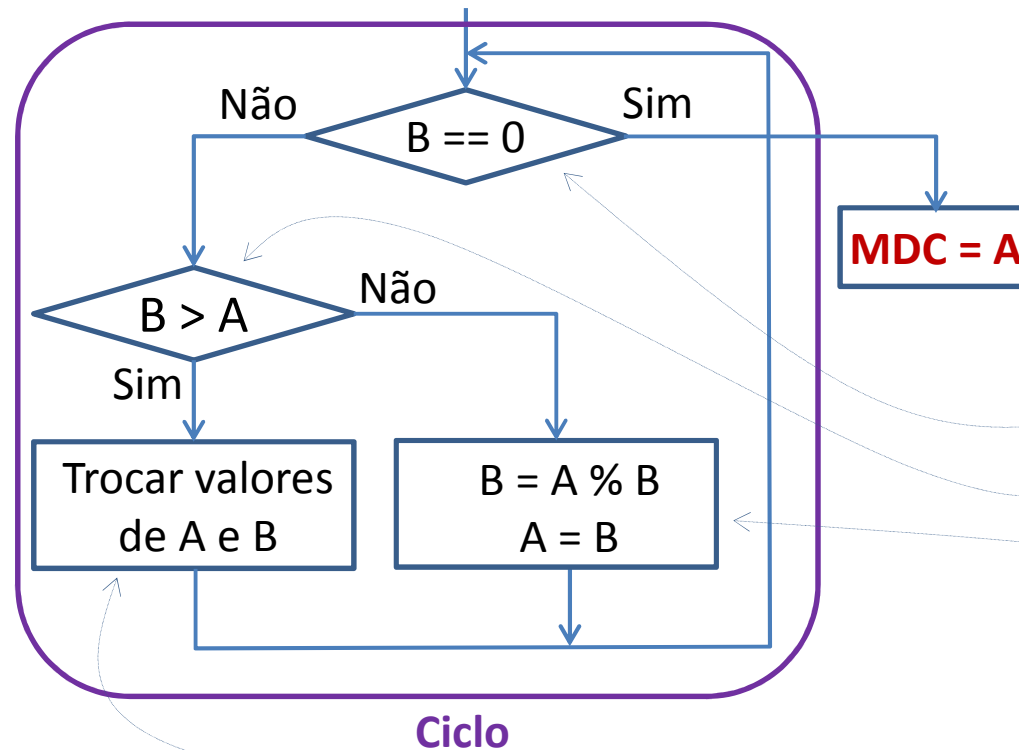
3. B = A % B = 0; A = 11.

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);
```

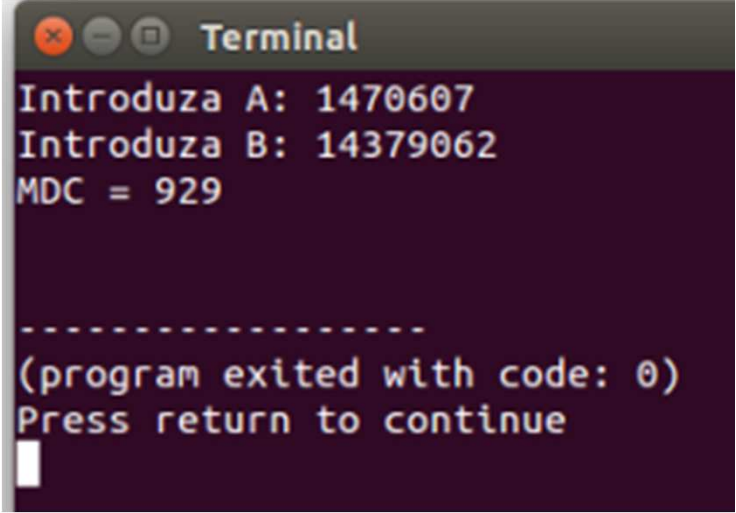
A = 33; B = 77

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

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.

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**

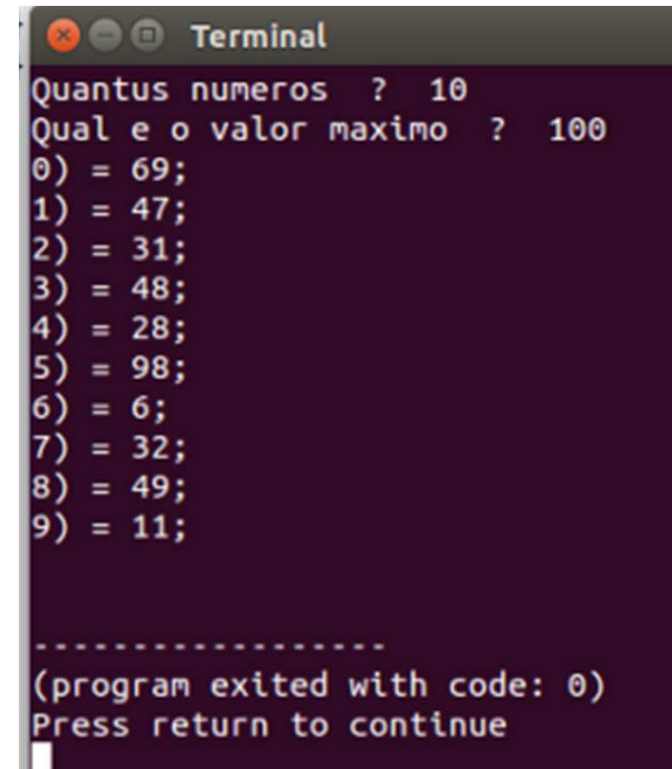
```
int i, n, soma = 0;
do {
    System.out.print("Valor de N [1 ... 99]: ");
    n = sc.nextInt();
} while(n < 1 || n > 100);
for(i = 1 ; i <= n ; i++)
    if(i % 2 != 0) soma += i;
System.out.printf("A soma dos ímpares é %d\n", soma);
```

```
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)+" ");
    }
}
```



```
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
```

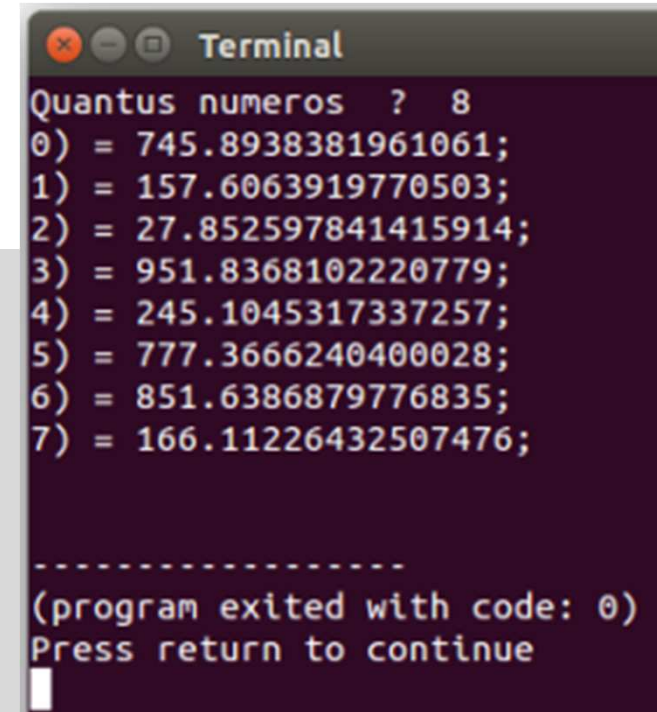
```
for(int i = 0; i < N; i++)
    System.out.printf("%d) = %d;\n", i, rand.nextInt(M));
```

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()+"; ");
```



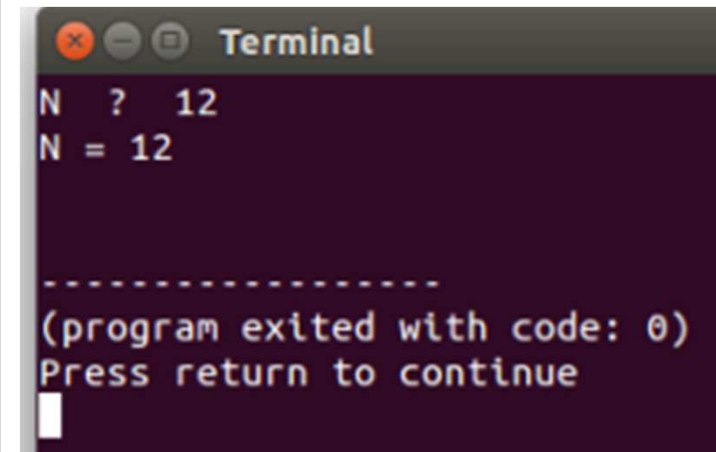
```
Terminal
Quantus numeros ? 8
0) = 745.8938381961061;
1) = 157.6063919770503;
2) = 27.852597841415914;
3) = 951.8368102220779;
4) = 245.1045317337257;
5) = 777.3666240400028;
6) = 851.6386879776835;
7) = 166.11226432507476;

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

```
Quantus numeros ? ?
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);
    }
}
```

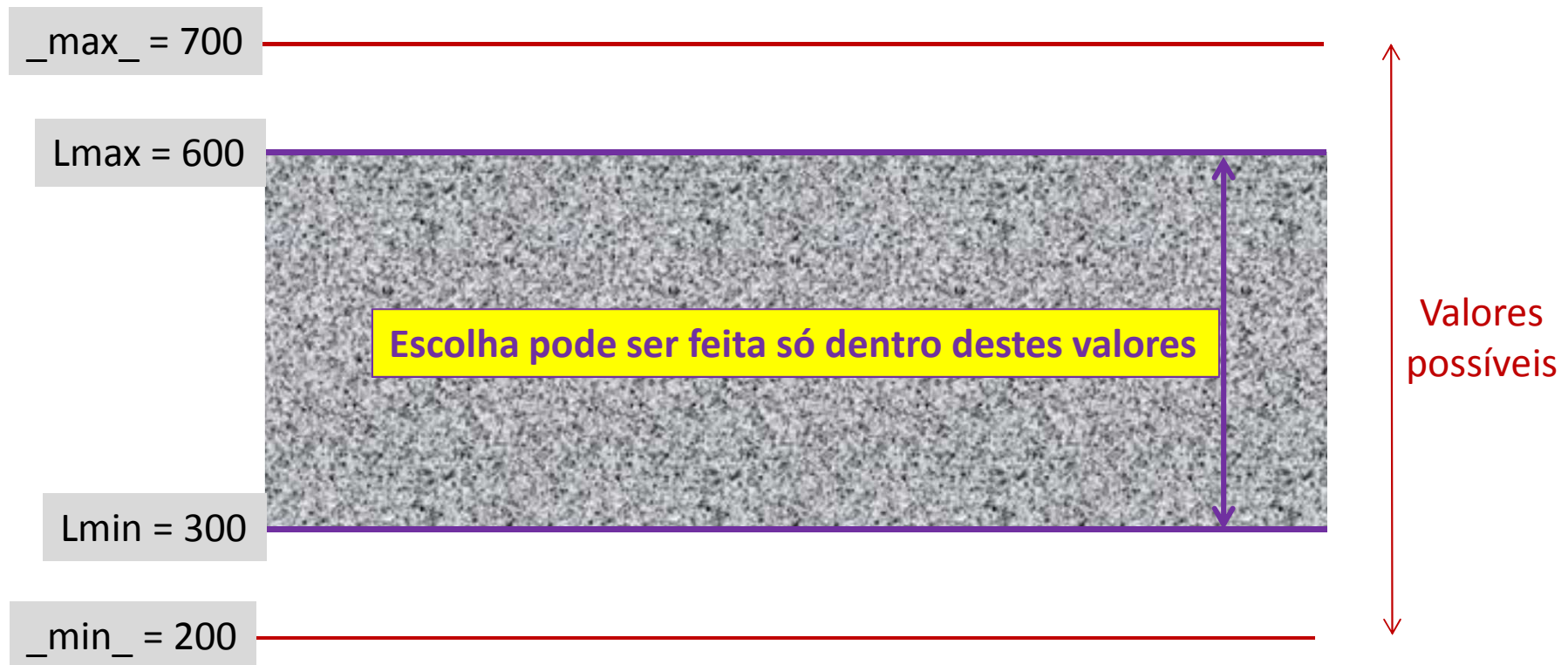


```
Terminal
N ? 12
N = 12

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

Exemplo 10: Filtragem de números entre `_min_` e `_max_` permite encontrar todos os números entre `Lmin` e `Lmax` (`_min_ < Lmin < Lmax < _max_`). O programa escolhe um número aleatoriamente. Vamos assumir que devem ser escolhidos `N` números.

```
int _max_ = 700, _min_ = 200, Lmin = 300, Lmax = 600, N = 1000;
```



Exemplo 7: Filtragem de números entre `_min_` e `_max_` permite encontrar todos os números entre `Lmin` e `Lmax` (`_min_ < Lmin < Lmax < _max_`). O programa escolhe um número aleatoriamente. Vamos assumir que devem ser escolhidos `N` números.

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

```

public class filtragem1
{
    public static void main(String[] args)
    {
        int _max_ = 700, _min_ = 200, Lmin = 300, Lmax = 600, N = 1000;
        int data;
        int cont = 0;
        for(int i = 1; i <= N; i++)
        {
            data= (int)((_max_ - _min_)*Math.random()) + 200;
            if (data < Lmin) continue;
            else if (data > Lmax) continue;
            else { cont++; System.out.print(data+" "); }
        }
        System.out.println("\ncont = " + cont);
    }
}

```

```

Terminal
518 362 308 477 401 373 347 310 534 523 335 562 334 543 420 417
441 387 317 324 571 540 578 321 530 402 505 418 349 329 432 485
347 549 386 512 452 421 444 355 570 445 452 390 305 340 532 383
429 412 518 439 445 468 404 432 396 462 371 433 385 531 476 435
499 548 361 380 564 542 483 416 432 584 475 426 510 542 441 378
402 301 451 301 330 320 455 430 580 542 349 463 547 379 414 411
397 498 302 319 525 480 560 519 303 375 417 368 588 367 457 572
433 595 470 470 518 431 367 489 486 436 316 369 315 387 556 442
378 570 385 596 369 585 387 417 318 580 397 532 372 301 387 559
599 480 472 592 347 525 504 594 401 373 519 496 356 577 522 342
596 430 368 562 350 368 457 592 523 405 495 336 331 322 420 513
302 518 597 594 506 302 496 363 345 391 471 461 397 591 401 320
383 357 506 580 324 498 529 471 368 480 479 413 425 535 472 374
469 390 579 373 300 307 576 505 508 516 431 496 428 553 584 506
502 420 589 592 570 420 324 436 536 445 566 382 448 596 341 433
392 486 522 358 352 578 342 447 363 451 401 347 600 502 319 366
496 506 517 455 594 476 428 395 349 562 326 542 366
cont = 605

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

```


Exemplo 11:

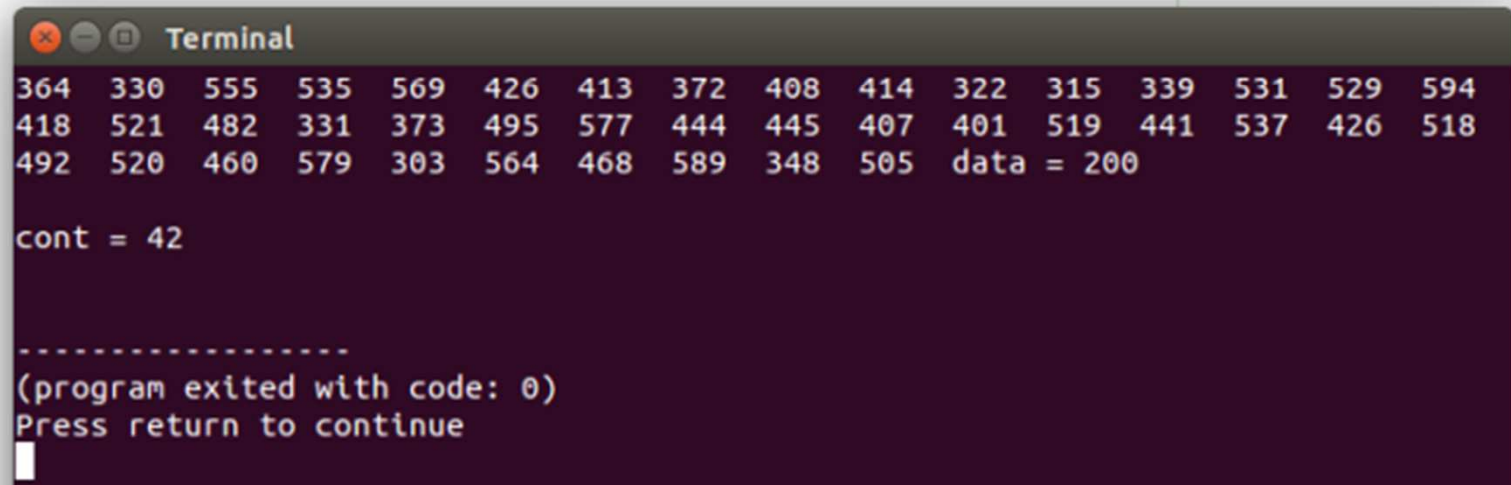
```
public class filtragem
{
    public static void main(String[] args)
    {
        int _max_ = 700, _min_ = 200, Lmin = 300, Lmax = 600, N = 1000;
        int data;
        int cont = 0;
        for(int i = 1; i <= N; i++)
        {
            data= (int)((_max_ - _min_)*Math.random()) + 200;
            if (data % 100 == 0) break; // terminar se aparecer um valor 200,300,400,500,600
            if (data < Lmin) continue;
            else if (data > Lmax) continue;
            else { cont++; System.out.print(data+" "); }
        }
        System.out.println("\ncont = " + cont);
    }
}
```



```

public class filtragem2
{
    public static void main(String[] args)
    {
        int _max_ = 700, _min_ = 200, Lmin = 300, Lmax = 600, N = 1000;
        int data;
        int cont = 0;
        for(int i = 1; i <= N; i++)
        {
            data= (int)((_max_ - _min_)*Math.random()+200;
            if (data % 100 == 0) { System.out.println("data = " + data); break; }
            if (data < Lmin) continue;
            else if (data > Lmax) continue;
            else { cont++; System.out.print(data+" "); }
        }
        System.out.println("\ncont = " + cont);
    }
}

```



```

Terminal
364 330 555 535 569 426 413 372 408 414 322 315 339 531 529 594
418 521 482 331 373 495 577 444 445 407 401 519 441 537 426 518
492 520 460 579 303 564 468 589 348 505 data = 200

cont = 42

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

```

Exemplo 12:

```
int ii = 0, jj = 10, count = 0;
```

```
do {
```

```
    if (ii++ == jj--) break;
```

```
    else { count++; continue; }
```

```
} while(count < 20);
```

```
if (count == 20)    System.out.printf("???? ii = %d; jj = %d\n",ii,jj);
```

```
else                System.out.printf("ii = %d; jj = %d\n",ii,jj);
```

ii = 6; jj = 4

```
int ii = 0, jj = 9, count = 0;
```

```
do {
```

```
    if (ii++ == jj--) break;
```

```
    else { count++; continue; }
```

```
} while(count < 20);
```

```
if (count == 20)    System.out.printf("???? ii = %d; jj = %d\n",ii,jj);
```

```
else                System.out.printf("ii = %d; jj = %d\n",ii,jj);
```

???? ii = 20; jj = -11

Exemplo 12:

```
int ii = 0, jj = 10, count = 0;
```

```
do {
```

```
    if (ii++ == jj--) break;
```

```
    else { count++; continue; }
```

```
} while(count < 20);
```

```
if (count == 20)    System.out.printf("???? ii = %d; jj = %d\n",ii,jj);
```

```
else                System.out.printf("ii = %d; jj = %d\n",ii,jj);
```

ii = 6; jj = 4

```
int ii = 0, jj = 10, count = 0;
```

```
do {
```

```
    if (++ii == --jj) break;
```

```
    else { count++; continue; }
```

```
} while(count < 20);
```

```
if (count == 20)    System.out.printf("???? ii = %d; jj = %d\n",ii,jj);
```

```
else                System.out.printf("ii = %d; jj = %d\n",ii,jj);
```

ii = 5; jj = 5

Exemplo 12:

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

```
int ii, jj, count;
for (ii = 0, jj = 10, count = 0; ii != jj; ++ii, --jj, count++)
    if (count >= 20) break;
if (count == 20)    System.out.printf("???? ii = %d; jj = %d\n", ii, jj);
else                System.out.printf("ii = %d; jj = %d\n", ii, jj);
```

ii = 5; jj = 5

???? ii = 20; jj = -11

```
int ii, jj, count;
for (ii = 0, jj = 9, count = 0; ii != jj; ++ii, --jj, count++)
    if (count >= 20) break;
if (count == 20)    System.out.printf("???? ii = %d; jj = %d\n", ii, jj);
else                System.out.printf("ii = %d; jj = %d\n", ii, jj);
```

```
int ii, jj, count;
for (ii = 0, jj = 10, count = 0; ++ii != --jj; count++)
    if (count >= 20) break;
if (count == 20)    System.out.printf("???? ii = %d; jj = %d\n", ii, jj);
else                System.out.printf("ii = %d; jj = %d\n", ii, jj);
```

ii = 5; jj = 5

```
int ii, jj, count;
for (ii = 0, jj = 10, count = 0; ++ii != --jj && count < 20; count++);
if (count == 20)    System.out.printf("???? ii = %d; jj = %d\n", ii, jj);
else                System.out.printf("ii = %d; jj = %d\n", ii, jj);
```

ii = 5; jj = 5

Exemplos adicionais

Exemplo 1: Representar um valor **v** inteiro positivo 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

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

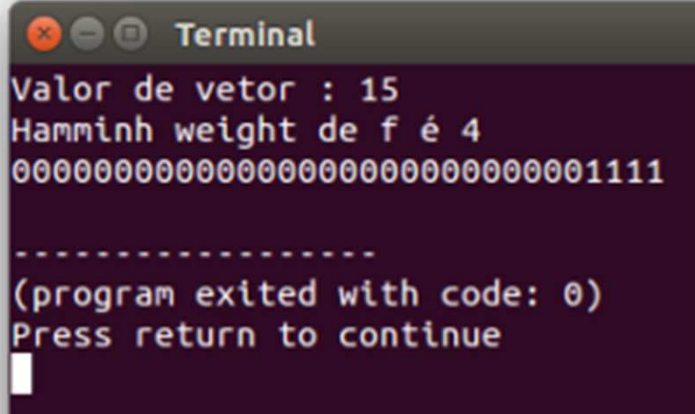
Exemplo 2: Computação de peso de *Hamming*

010111010111 – exemplo de vetor **V** binário (só pode ter valores 1 e 0)

Peso de Hamming $W(V)$ do vetor **V** é o número de valores 1 no vetor **V**

Por exemplo, $W(010111010111) = 8$

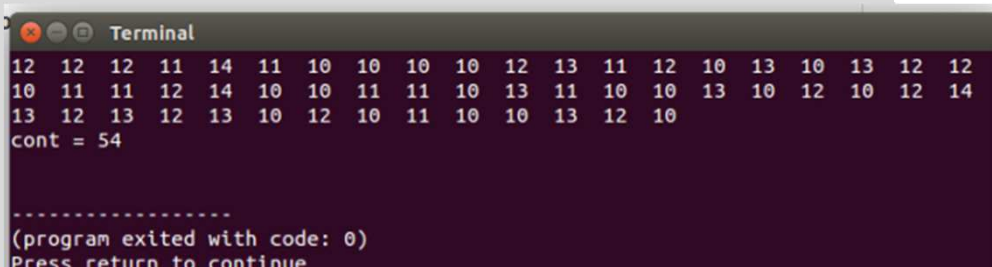
```
import java.util.Scanner;
public class hw {
    static Scanner sc = new Scanner(System.in);
    public static void main (String args[]) {
        int V;
        int HW = 0;
        System.out.print("Valor de vetor : ");
        V = sc.nextInt();
        for (int i = 0; i <= 31; i++) {
            HW += (V >> i) & 1;
        }
        System.out.printf("Hamming weight de %h é %d\n", V, HW);
        for (int i = 31; i >= 0; i--) {
            System.out.print((V >> i) & 1);
        }
    }
}
```



```
Terminal
Valor de vetor : 15
Hamming weight de f é 4
000000000000000000000000000001111
-----
(program exited with code: 0)
Press return to continue
```

Exemplo 3: filtragem de vetores binários

```
public class hw_filtragem
{
    public static void main(String[] args)
    {
        int _max_ = 1000000, _min_ = 0, Lmin = 10, Lmax = 20, N = 100;
        int V, HW;
        int cont = 0;
        for(int i = 1; i <= N; i++)
        {
            V = (int)((_max_ - _min_)*Math.random());
            HW = 0;
            for (int j = 0; j <= 31; j++)
                HW += (V >> j) & 1;
            // System.out.printf("\nHamming weight de %h é %d\n", V, HW); // só para verificação
            // for (int j = 31; j >= 0; j--) // só para verificação
            //     System.out.print((V >> j) & 1); // só para verificação
            if (HW < Lmin) continue;
            else if (HW > Lmax) continue;
            else { cont++; System.out.print(HW+" "); }
        }
        System.out.println("\ncont = " + cont);
    }
}
```



```
Terminal
12 12 12 11 14 11 10 10 10 10 12 13 11 12 10 13 10 13 12 12
10 11 11 12 14 10 10 11 11 10 13 11 10 10 13 10 12 10 12 14
13 12 13 12 13 10 12 10 11 10 10 13 12 10
cont = 54

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


**Exemplos seguintes são
úteis para futuro**

Instrução `for` é usada mais frequentemente com *arrays*.

Um *array* pode ser declarado como: **int[]** a = { 1, 2, 3, 4, 5 };

ou **int** a[] = { 1, 2, 3, 4, 5 };

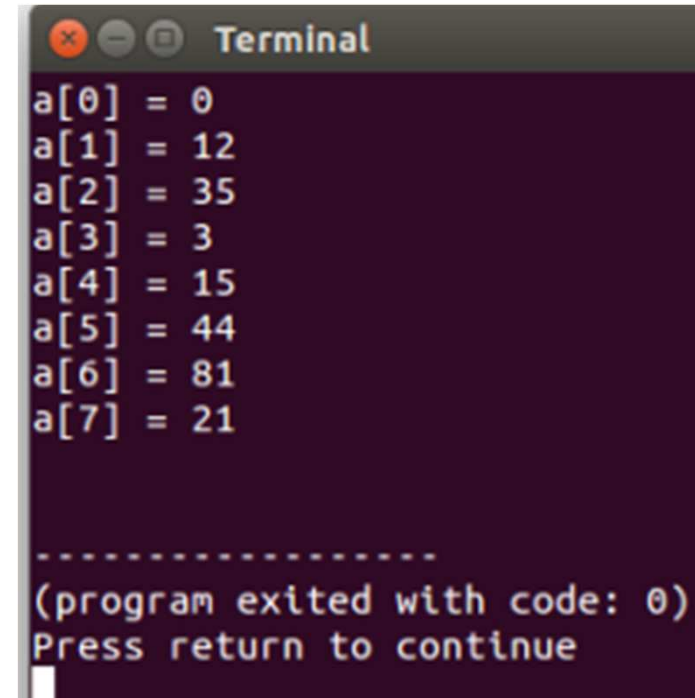
Exemplo 4:

```
public class for_and_array
{
    public static void main(String[] args)
    {
        int[] a = { 1, 2, 3, 4, 5 };
        // pode também declarar array como: int a[] = { 1, 2, 3, 4, 5 };
        for(int i = 0; i < a.length; i++)
            System.out.println("a[" + i + "] = " + a[i]);
        System.out.println("-----");
        for(int i = a.length-1; i >= 0; i--)
            System.out.println("a[" + i + "] = " + a[i]);
    }
}
```

Exemplo 5:

O seguinte exemplo permite gerar dados aleatoriamente:

```
import java.util.*;
public class for_and_array
{
    static Random rand = new Random();
    public static void main(String[] args)
    {
        int a[] = new int[8];
        for(int i = 0; i < a.length; i++)
        { a[i] = rand.nextInt(100);
          System.out.println("a[" + i + "] = " + a[i]);
        }
    }
}
```

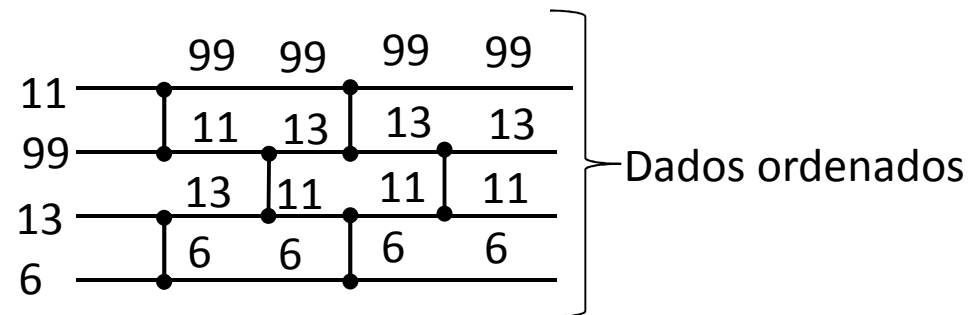


```
Terminal
a[0] = 0
a[1] = 12
a[2] = 35
a[3] = 3
a[4] = 15
a[5] = 44
a[6] = 81
a[7] = 21

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

Exemplo 6:

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



Exemplo 6: O seguinte exemplo permite gerar dados aleatoriamente e ordenar dados utilizando uma rede de ordenação:

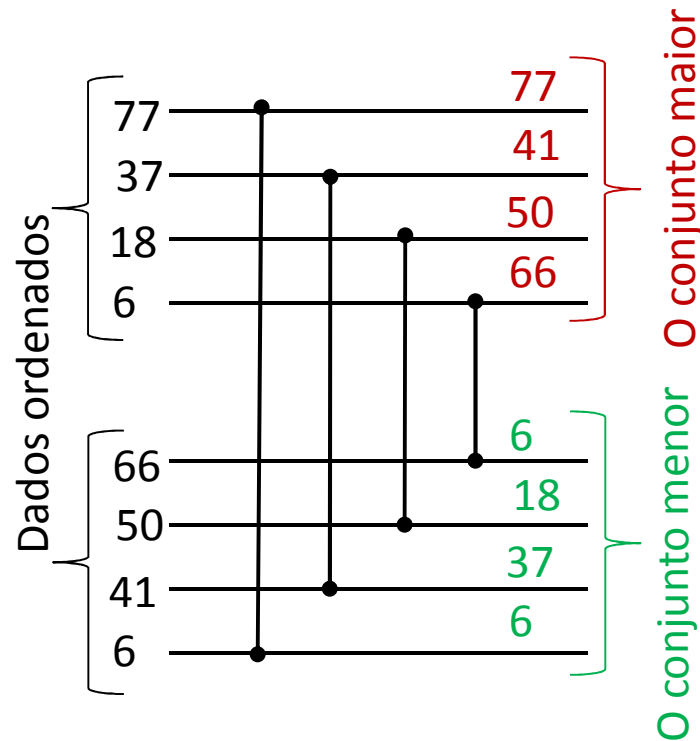
```
import java.util.*;
public class sorting_network
{
    static Random rand = new Random();
    public static void main(String[] args)
    {
        int N = 64, tmp; // N pode ser qualquer valor de  $2^R$ ,  $R = 0,1,2,3,4,5,6,7,8,9,10,\dots$ 
        int a[] = new int[N];
        for(int i = 0; i < a.length; i++)
        {
            a[i] = rand.nextInt(1000);
            System.out.println("a[" + i + "] = " + a[i]);
        }
        System.out.println("-----");
        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("%10d; ", a[i]);
            if (((i+1)%10) == 0) System.out.println();
        }
    }
}
```

```
Terminal
a[0] = 520
a[1] = 526
a[2] = 298
a[3] = 197
a[4] = 562
a[5] = 724
a[6] = 490
a[7] = 386
a[8] = 959
a[9] = 239
a[10] = 775
a[11] = 488
a[12] = 763
a[13] = 557
a[14] = 324
a[15] = 44
a[16] = 238
a[17] = 4
a[18] = 542
a[19] = 437
a[20] = 840
a[21] = 60
a[22] = 802
a[23] = 3
a[24] = 717
a[25] = 262
a[26] = 534
a[27] = 499
a[28] = 886
a[29] = 842
a[30] = 49
a[31] = 695
-----
959; 886; 842; 840; 802; 775; 763; 724; 717; 695;
562; 557; 542; 534; 526; 520; 499; 490; 488; 437;
386; 324; 298; 262; 239; 238; 197; 60; 49; 44;
4; 3;
-----
(program exited with code: 0)
Press return to continue
```

```
Terminal
a[0] = 136
a[1] = 183
a[2] = 164
a[3] = 497
a[4] = 990
a[5] = 483
a[6] = 855
a[7] = 17
a[8] = 445
a[9] = 340
a[10] = 311
a[11] = 801
a[12] = 282
a[13] = 200
a[14] = 535
a[15] = 825
-----
990; 855; 825; 801; 535; 497; 483; 445; 340; 311;
282; 200; 183; 164; 136; 17;
-----
(program exited with code: 0)
Press return to continue
```

Exemplo 7:

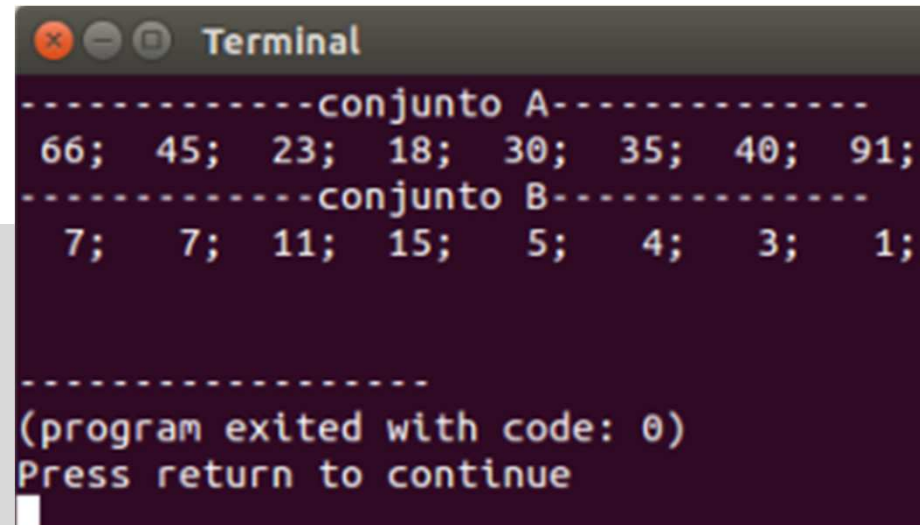
Aplicação da rede em baixo a dois conjuntos ordenados permite encontrar o conjunto maior (em cima) e o conjunto menor (em baixo)



Escrever um programa para verificar esta regra

Exemplo 7:

```
public class sort
{
    public static void main(String[] args)
    {
        int[] A = { 66,45,23,18,15,11,7,7 };
        int[] B = { 91,40,35,30,5,4,3,1 };
        int tmp;
        for (int i = 0, j = A.length-1; i < A.length; i++, j--)
            if (A[i] < B[j]) { tmp = A[i]; A[i] = B[j]; B[j] = tmp; }
        System.out.println("-----conjunto A-----");
        for(int i = 0; i < A.length; i++) System.out.printf("%3d; ",A[i]);
        System.out.println("\n-----conjunto B-----");
        for(int i = 0; i < B.length; i++) System.out.printf("%3d; ",B[i]);
        System.out.println();
    }
}
```



```
Terminal
-----conjunto A-----
66; 45; 23; 18; 30; 35; 40; 91;
-----conjunto B-----
7; 7; 11; 15; 5; 4; 3; 1;

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


**Vamos falar sobre estes
exemplos mais uma vez
quando vamos estudar
arrays**