

```
statement(s)
}
```

```
statement(s)
last_statement;
}
```



## Ciclos

Apontamentos sobre ciclos em Java: while, for, do - while

Page

- Os programas vistos até agora são executados sequencialmente com algumas instruções a serem executadas de forma condicionada
- Mas por vezes é necessário repetir certos procedimentos:
  - Enquanto o utilizador não introduzir um valor válido
  - Esperar que uma condição seja atingida
- Em Java podem encontrar-se 3 tipos de ciclos:
  - `while`
  - `do`
  - `do - while`

## While

- A sintaxe do `while` é a seguinte:

```
JavaScript ▾
while (condição)
    instrução;
```

- Enquanto a condição for verdadeira, a instrução é executada
- A condição é avaliada no início de cada ciclo
- Quando a condição for falsa, o ciclo termina e a execução retoma a seguir ao ciclo
- Não esquecer que uma instrução pode ser um bloco de código

```
JavaScript ▾
while (condição) {
    instrução1;
    instrução2;
}
```

```
JavaScript ▾
int num;

System.out.println("Introduza um número positivo: ");
num = input.nextInt();

while( num < 0 ){
    System.out.println("Eu disse POSITIVO, introduza novamente: ");
    num = input.nextInt();
}
```

```

}

System.out.println("obrigado");

// Introduza um número positivo: 12 -> obrigado
// Introduza um número positivo: -12 -> Eu disse POSITIVO, introduza novamente: 5 -> obrigado

```

- Reparar que a condição é sempre avaliada antes do ciclo começar
- Se a condição for falsa logo à partida, o ciclo nem chega a ser executado

```

JavaScript ▾
while (condição)
    instrução;

```

⚠ Não esquecer de alterar a variável que controla o ciclo!! Senão fica com ciclo infinito

## For

- A sintaxe do ciclo `for` é a seguinte:

```

JavaScript ▾
for (inicializações; condição; pós-instruções)
    instrução

```

- Primeiro são executadas as inicializações
- Depois, enquanto condição for verdadeira a instrução é executada
- No final do ciclo são executadas as pós-instruções
- A condição é avaliada no início de cada ciclo
- Quando condição for falsa o ciclo termina e a execução retoma a seguir ao ciclo
- Não esquecer que instrução pode ser um bloco de código:

```

JavaScript ▾
for (inicializações; condição; pós-instruções) {
    instrução1;
    instrução2;
}

```

- No bloco de inicializações devem ser colocadas as inicializações das variáveis de controlo do ciclo
- No bloco de pós-instruções devem ser colocadas as atualizações das variáveis de controlo de ciclo

```

JavaScript ▾
for (int i = 1; i <= 10; i++) {
    System.out.println(i);
}

// 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10

```

- O ciclo `for` começa por declarar e inicializar a variável `i` a 1
- Depois verifica se `i <= 10`
- Depois executa as instruções (imprime o `i`)

- No final do ciclo executa a pós-instrução ( `i++` )
- Volta a verificar a condição ( `i <= 10` ), e repete o ciclo
- O ciclo `for` é uma forma mais compacta de se escrever um ciclo `while`
- É mais utilizado quando se sabe à partida quantas iterações se vão efetuar
- Podem-se converter ciclos `for` para `while` e vice-versa:

JavaScript ▾

```
for (inicializações; condição; pós instruções) {
    instrução;
}
```

JavaScript ▾

```
inicializações;
while (condição) {
    instrução;
    pós-instrução;
}
```

- Quando há mais que uma inicialização a fazer elas devem ser separadas por , (vírgula)
- O mesmo quando há mais que uma pós-instrução

JavaScript ▾

```
for (i = 2, j = 3, i < 20; i++, j += 2) {
    System.out.printf("%d, %d"), i, j);
}
```

## Do-While

- A sintaxe do ciclo do-while é a seguinte:

JavaScript ▾

```
do
    instrução;
while (condição);
```

- Primeiro é executada a instrução
- Depois, enquanto condição for verdadeira a instrução é executada
- A condição é avaliada no final de cada ciclo
- Quando a condição for falsa, o ciclo termina e a execução retoma a seguir ao ciclo
- Não esquecer que a instrução pode ser um bloco de código:

JavaScript ▾

```
do {
    instrução1;
    instrução2;
} while (condição);
```

- Este ciclo é diferente dos outros já que executa sempre, pelo menos, uma vez

JavaScript ▾

```
int soma = 0;

do {
    System.out.println("Escreva um número (zero para terminar): ");
```

```

    int num = input.nextInt();
    soma += num;
} while( num != 0 );
    System.out.println("\n\nTotal = ", soma);

// Escreva um número (zero para terminar): 1
// Escreva um número (zero para terminar): 10
// Escreva um número (zero para terminar): 20
// Escreva um número (zero para terminar): 3
// Escreva um número (zero para terminar): 12
// Escreva um número (zero para terminar): 0

// Total = 46

```

## Uso do break

- Por vezes, há situações que obrigam a terminar a execução de um ciclo, sem que a condição de paragem seja verdadeira
- Para isso pode-se usar o `break`
- O `break` força a saída do ciclo continuando a execução após este

```

JavaScript ▾
int soma = 0;
for( int i=0; i < 5; i++ ){
    System.out.println("Introduza um número: ");
    int num = input.nextInt();

    if( num == 0 )
        break;
    soma += num;
}

System.out.println ("Total = " + soma);

// Escreva um número: 2
// Escreva um número: 4
// Escreva um número: 3
// Escreva um número: 10
// Escreva um número: 7

// Total = 26

```

## Uso do continue

- O `continue` termina a execução da iteração atual e passa para a próxima iteração do ciclo

```

JavaScript ▾
int soma = 0;
for( int i=0; i < 5; i++ ){
    System.out.println("Introduza um número: ");
    int num = input.nextInt();

    if( num < 0 )
        continue;
    soma += num;
}

System.out.println("Soma positivos = " + soma);

// Introduza um número: 2
// Introduza um número: 4

```

```
// Introduza um número: 3
// Introduza um número: 10
// Introduza um número: 7

// Soma positivos = 26
```

## Ciclos infinitos

- Um ciclo infinito é um ciclo que não tem fim
- São usados quando a partida não se sabe quando terminam:
  - Para os terminar usa-se a instrução `break`
- Os ciclos infinitos são usados, por vezes, para controlar a saída de uma aplicação, quando a mesma pode ser interrompida de várias maneiras
- O uso de ciclos infinitos deve ser evitado

```
Java ▾
while (true) {
    // ciclo infinito
    if (user_pressionaESC)
        break;

    if (user_usaBotaoSair)
        break;

    if (erroFatal)
        break;
}
```