JAVA.EstruturasDeControlo.Condicionais

- *Selecção simples (if)*

```
if (condição) Instruções_Se_Condição_Verdadeira
```

- *Selecção em alternativa (if/else)*

```
if (condição) ISCV else ISCFalsa
```

Máximo de 3 números (3 estilos)

```
if ((a>b)&&(a>c))
    maior = a;
else if (b>c)
    maior = b;
else
    maior = c;
```

```
if ((a>b)&&(a>c)){
    maior = a;
}else{
    if (b>c){
        maior = b;
    }else{
        maior = c;
    }
}
```

```
if ((a>b)&&(a>c)) maior = a;
else if (b>c) maior = b;
else maior = c;
```

- *Selecção múltipla (switch-case)*

```
switch (expressão){
    case valor1: Instruções1; [break;]
    ...
    case valorN: InstruçõesN; [break;]
    default: InstruçõesCasoOmisso; [break;]
}
```

O valor da expressão terá de ser inteiro ou carácter (`byte, char, short, int, long`).

A instrução `break` é necessária para que os ramos de instruções não sejam executados sequencialmente.

A instrução switch-case pode ser sempre substituída por um aninhamento de instruções if-else mas...

...o recíproco não se verifica!

```
switch(cAlfaNum){
    case 'A': case 'E': case 'I': case 'O': case 'U':
    case 'a': case 'e': case 'i': case 'o': case 'u':
            System.out.println("vogal"); ...;  break;
    case '0': case '1': case '2': case '3': case '4':
    case '5': case '6': case '7': case '8': case '9':
            System.out.println("dígito"); ...;  break;
    default: System.out.println("consoante"); ...;  break;
}
```

JAVA.EstruturasDeControlo.Repetitivas

- **Ciclo *for***

```
for(inicialização; condição_de_continuação; iteração)
        instruções_a_iterar;
```

A secção de inicialização contém usualmente a declaração! e inicialização da variável de controlo do ciclo.

`condição_de_continuação` é uma expressão booleana avaliada antes de cada iteração e que determina a saída do ciclo logo que o seu valor seja `false`.

O bloco `iteração` é executado após as `instruções_a_iterar` e serve usualmente para actualizar o valor da variável de controlo.

Todos os blocos são opcionais: `for(;;){...}` é válido!

Para que serve o seguinte código?

```
for(int tmp, i=0, j=v.length-1;i<j;++i,--j){
    tmp = v[i]; v[i] = v[j]; v[j] = tmp;
}
for(char i=65; i< 91; ++i){
    System.out.print(i);
}
```

Ciclo *while*

```
while(condição_de_continuação)
        instruções_a_iterar;
```

O modo de funcionamento é muito simples: enquanto a `condição_de_continuação` for verdadeira as `instruções_a_iterar` são executadas.

A seguinte estrutura é equivalente ao ciclo `for`:

```
inicialização;
while(condição_de_continuação){
        instruções_a_iterar;
        iteração;
}
```

- **Ciclo *do/while***

```
do{
        instruções_a_iterar;
}while(condição_de_continuação)
```

Esta estrutura deve ser executada quando se pretende que as instruções_a_iterar sejam executadas pelo menos uma vez.
No final da execução, a `condição_de_continuação` é testada e se o seu valor for `false` a iteração termina.

Exercício: faça um programa capaz de gerar uma chave do Totoloto (6 números inteiros distintos entre 1 e 49). Sugestão: utilize o gerador de números pseudo-aleatórios da classe Math: Math.random(). Este método devolve um valor do tipo *double* pertencente ao intervalo [0, 1[.