iscte
INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Módulo 1: Programação em Linguagem Java

Aula 4
Ciclos e Vetores

UPskill

Porque se utilizam **Ciclos?**

- O Java permite a utilização de um tipo de estruturas (ciclos) que permitem repetir uma operação ou uma sequência de operações sucessivas.

```
Enquanto i < 100 faz:
    System.out.println("Aula 4");
```

- Quando se constrói um ciclo define-se uma guarda cujo valor dá continuidade ou interrompe o ciclo.

```
i < 100
```

- Sendo um conceito fundamental para a programação em Java são definidos três tipos formatos:
  - Ciclo **`while`**;
  - Ciclo **`do-while`**;
  - Ciclo **`for`.**

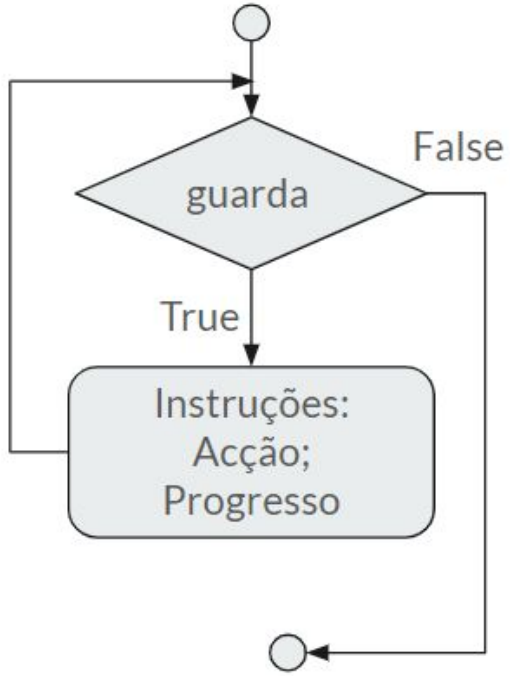
Ciclo While

- Sintaxe:

```
while (condição) {
    Instruções;
}
```

- No exemplo anterior:

```
int i = 0;
while ( i < 100 ) {
    System.out.println("Aula 4");
    i = i + 1;
}
```



Importante para obrigar a guarda a passar a **false**!

Ciclo While

- Qual o resultado esperado do seguinte código.

```
int soma = 0;
int i = 0;
while ( i < 10 ) {
    soma = soma +i;
}
System.out.println("O valor é:" + soma);
```

- Seria um ciclo infinito pois a variável **i** é sempre **<10**.

Ciclo While

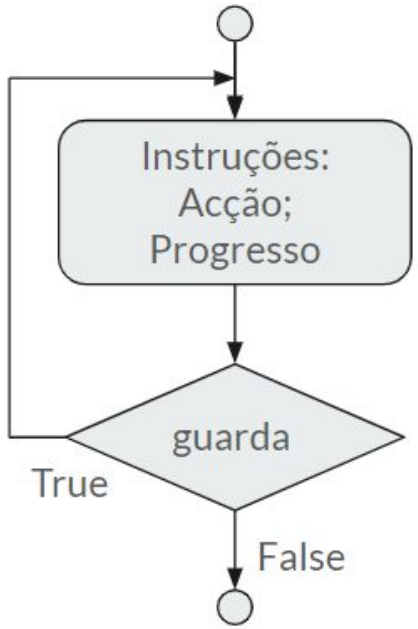
- A guarda é sempre verdadeira porque a variável **i** é sempre **<10**, devia-se ter incrementado a variável **i**.

```
int soma = 0;
int i = 0;
while ( i < 10 ) {
    soma = soma +i;
    i++;
}
System.out.println("O valor é:" + soma);
```

Ciclo Do-While

- Variação da estrutura do ciclo **`while`**. Sintaxe:

```
do {
    Instruções;
} while (condição);
```



- No exemplo inicial:

```
int i = 0;
do {
    System.out.println("Aula 4");
    i++;
} while ( i < 100 );
```

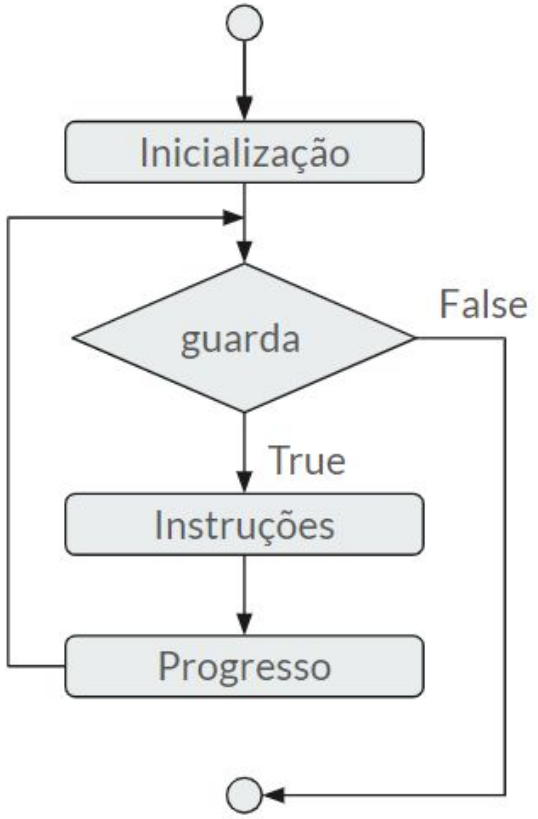
Ciclo For

- Sintaxe:

```
for ( inicialização; guarda; progresso; ) {
    Instruções;
}
```

- No exemplo inicial:

```
for ( int i = 0; i < 100; i++ ) {
    System.out.println("Aula 4");
}
```



Que ciclo usar?

- Os três tipos de ciclos apresentados são equivalentes nas suas funções.

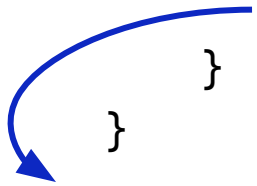
- Os ciclos **`for`** e **`while`** são *pretest loops* pois a condição da guarda é validada antes das instruções serem executadas. O ciclo **`do-while`** é *posttest loop* pois valida a guarda após as instruções serem executadas.

- Por norma o ciclo **`for`** pode ser utilizado quando se conhece, no início, o número de repetições pretendidas. Já o ciclo **`while`** pode ser utilizado quando não se sabe quantas repetições vão ser realizadas.

Break, é possível usar em ciclos?

- À semelhança das estruturas **`switch`** também nos ciclos é possível utilizar a expressão **`break`**. A utilização desta expressão termina no imediato o ciclo.

```
int soma = 0;
int i = 0;
while ( i < 20 ) {
    soma = soma + i;
    i++;
    if ( soma >= 100) {
        break;
    }
}

System.out.println("O valor é:" + soma);
```



Exercício 1

- Escreva os números pares de 1 a 20, usando as três estruturas de ciclos apresentadas.

```
while (condição) {
    Instruções;
}

do {
    Instruções;
} while (condição);

for ( inicialização; guarda; progresso; ) {
    Instruções;
}
```

Exercício 1 - Resolução

```
public static void main(String[] args) {
    System.out.println("while");
    int a =1;
    while( a<=20){
        if(a%2==0){
            System.out.println(a);
        }
        a++;
    }
    System.out.println("do-while");
    int b =1;
    do{
        if(b%2==0){
            System.out.println(b);
        }
        b++;
    } while(b <=20);
    System.out.println("for");
    for(int c =1; c<=20; c++){
        if(c%2==0){
            System.out.println(c);
        }
    }
}
```


```
System.out.println("do-while");
int b =1;
do{
    if(b%2==0){
        System.out.println(b);
    }
    b++;
} while(b <=20);
```

Recursividade

- Quando uma função contém **invocações a si própria**, esta é considerada uma função recursiva.

- Recursividade é um conceito fundamental em computação. Em alguns paradigmas de programação (por exemplo, no paradigma funcional), a recursividade assume uma importância fulcral.

- Dada a sua proximidade com as definições matemáticas, a “elegância" das definições de funções recursivas tornam-as atrativas em certos contextos.

Exemplo - Sucessão de Fibonacci

- Definição **matemática** da função para obter o n-ésimo número da sucessão de Fibonacci

$$f(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ f(n-1) + f(n-2) & \text{outros casos} \end{cases}$$

- Função recursiva **em Java**

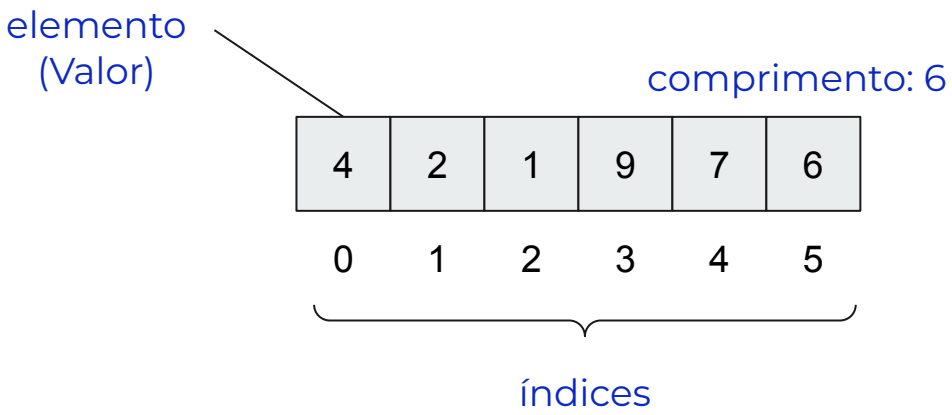
```
static int fibonacci(int n) {
     if(n <= 1) {
          return n;
      } else {
          return fibonacci(n - 1) + fibonacci(n - 2);
      }
 }
```

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

Vetores

- Estrutura elementar que permite armazenar dados de forma organizada.

- Os acessos aos elementos do vetor são efectuados mediante um índice.



Criação de Vetores

- Em Java, os vectores têm comprimento fixo. Desta forma, ao ser criado um vector é necessário indicar um comprimento, o qual não pode ser alterado após a criação.

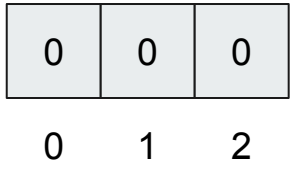
```
int[] v1 = new int[3];
boolean[] v2 = new boolean[2];
```

- Quando os vectores são criados, os elementos tomam um valor por omissão. No caso de um vector de inteiros esse valor é **0**, já no caso de um vetor de *boolean* este valor é *false*.

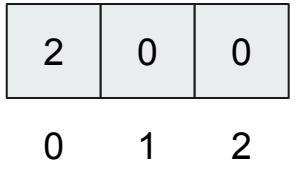
Manipulação de vetores: Modificação

- A modificação dos elementos do vector é feita mediante um índice.

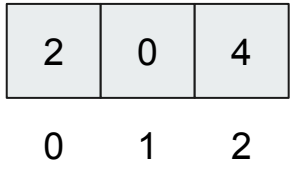
```
int[] v = new int[3];
```




```
v[0] = 2;
```

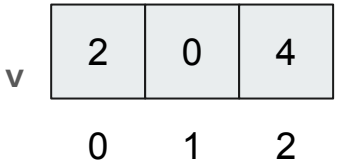



```
v[2] = 4;
```



Manipulação de vetores: Acesso

- O comprimento de um vector pode ser obtido através do atributo *length*.



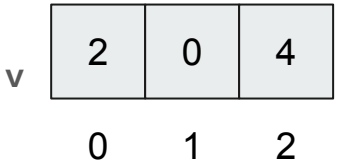

```
int size = v.length;
int first = v[0];
int last = v[v.length-1];

System.out.println("O tamanho do vector é: " + size);
System.out.println("O primeiro elemento do vector é: " + first);
System.out.println("O último elemento do vector é: " + last);
```

Resultado na consola:

```
O tamanho do vector é: 3
O primeiro elemento do vector é: 2
O último elemento do vector é: 4
```

Manipulação de vetores: Iteração

- O processo de percorrer os elementos de um vector pode ser designado por iteração. Tipicamente, a iteração faz uso de uma variável que toma sucessivamente os valores dos índices do vector que se quer aceder (variável **i**).

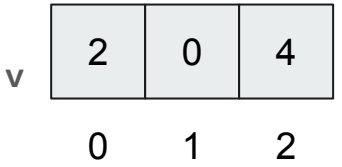



```
public static int sum(int[] v) {
    int i = 0;
    int sum = 0;
    while(i != v.length) {
        sum = sum + v[i];
        i = i + 1;
    }
    return sum;
}
```

Manipulação de vetores: Iteração

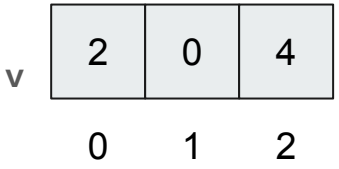
- O que faz a função que se segue?

```
public static int sum(int[] v) {
    int i = 0;
    int sum = 0;
    while(i != v.length) {
        sum = sum + v[i];
        i = i + 1;
    }
    return sum;
}
```



Manipulação de vetores: Iteração

- O que faz a função que se segue?

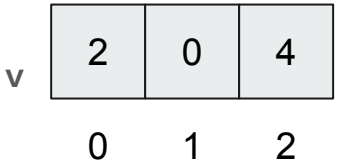



```
public static int sum(int[] v) {
    int i = 0;
    int sum = 0;
    while(i != v.length) {
        sum = sum + v[i];
        i = i + 1;
    }
    return sum;
}
```

- Calcula o somatório de todos os elementos do vector na variável **`sum`**.

Manipulação de vetores: Iteração

- Ao iterar-se sobre um vector é também possível contar-se o número de ocorrência de determinado valor no vetor




```
public static int numberOfOccurrences(int a, int[] v) {
    int i = 0;
    int count = 0;
    while(i != v.length) {
        if ( v[i] == a ){
            count = count + 1;
        }
        i = i + 1;
    }
    return count;
}
```

Exercício 2

- Escreva uma função que pesquisa num vetor de inteiros, a existência de um determinado valor inteiro.

Exercício 2 - Resolução

```
public static boolean exists(int a, int[] v) {
   int i = 0;
   while(i != v.length) {
       if(v[i] == a) {
           return true;
       }
       i = i + 1;
   }
   return false;
}
```

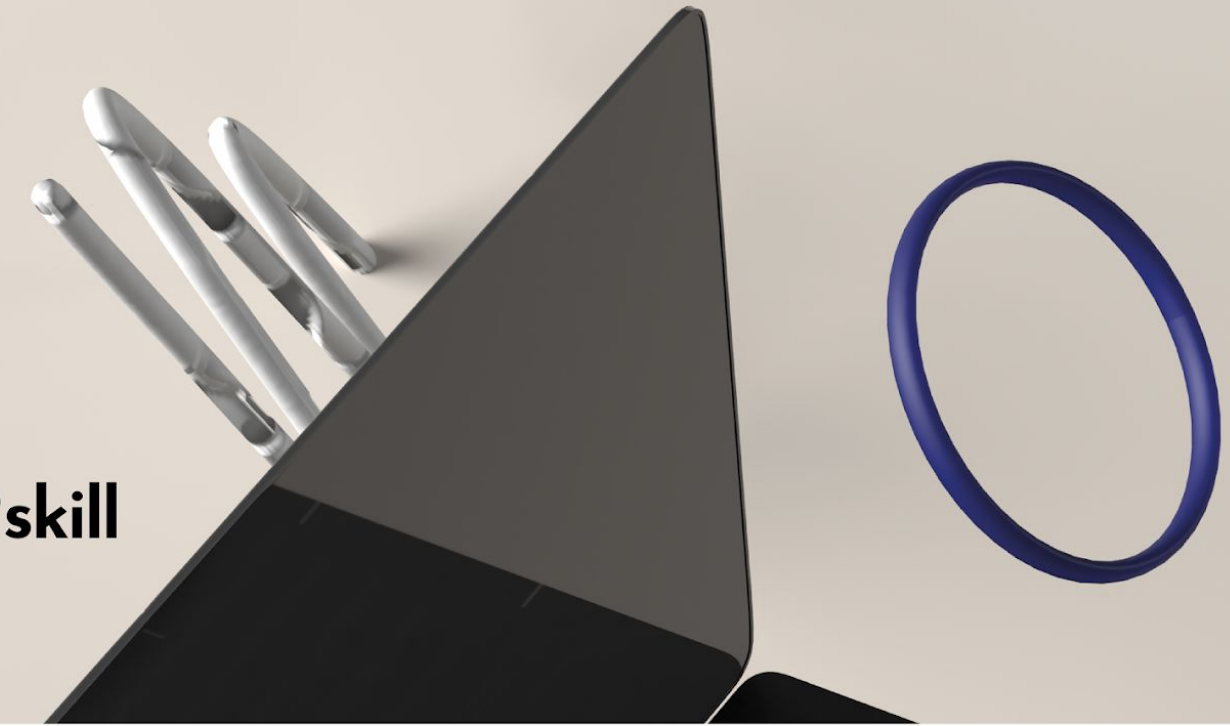
Exercício 3

- Escreva uma função que determina o valor máximo dentro de um vetor de inteiros.

Exercício 3 - Resolução

```java
public static int max(int[] v) {
   int max = v[0];
   int i = 1;
   while(i != v.length) {
       if(v[i] > max) {
           max = v[i];
       }
       i = i + 1;
   }
   return max;
}
```

O futuro profissional começa aqui



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UPskill