CURSO DE PROGRAMAÇÃO EM JAVA

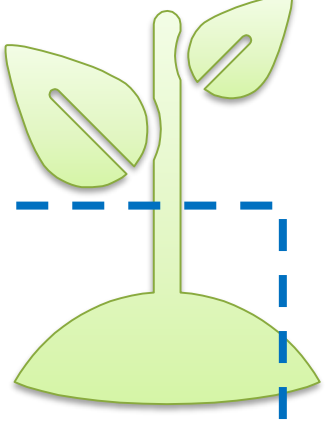
Aula 8

Estruturas de repetição II

As estruturas de repetição também são conhecidas como laços (loops) e são utilizados para executar, repetidamente, uma instrução ou bloco de instrução enquanto determinada condição estiver sendo satisfeita.

Revisando...

Instrução while



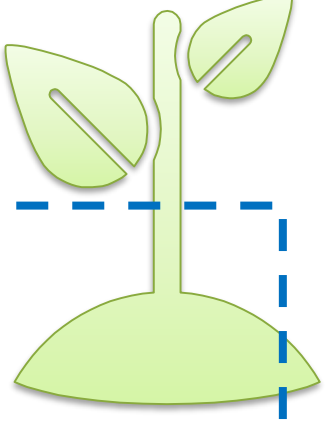
```
while(CONDIÇÃO){
    COMANDO(S);
}
```

```
public class contando {
    public static void main(String[] args) {
        int count=1;

        while(count<=10){
            System.out.println(count);
            count++;
        }
    }

}
```

Instrução do-while

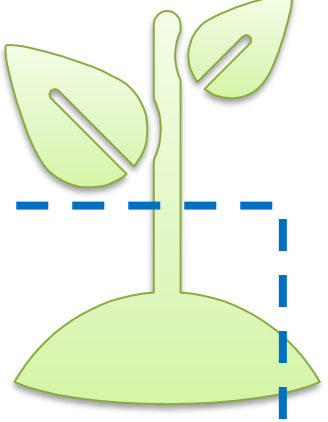


```
do{
    COMANDO(S);
} while(CONDIÇÃO);
```

```
public class somaValores {
    public static void main(String[] args) {
        int soma=0;
        int aux=0;
        Scanner input = new Scanner(System.in);
        do{
              soma += aux;
              aux = input.nextInt();
        }while(aux!=-1);


        System.out.println(soma);
    }
}
```

Instrução break

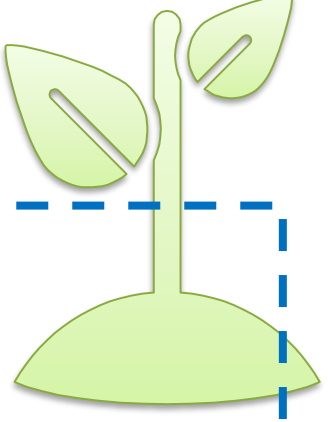


```
break;
```

```
public class somaValores {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int soma=0;
        int i=0;
        int aux=0;
        while(i<10);{
              aux = input.nextInt();
              if(aux == -1) break;
              soma += aux;
              i++;
        }
        System.out.println(soma);
    }
}
```

Instrução continue

```
continue;
```



```
public class teste {
    public static void main(String[] args) {
        int i=0;
        while(i<10);{
              i++;
              if(i%2==0)
                    continue;
              System.out.println(i+" ");
        }
    }
}
```

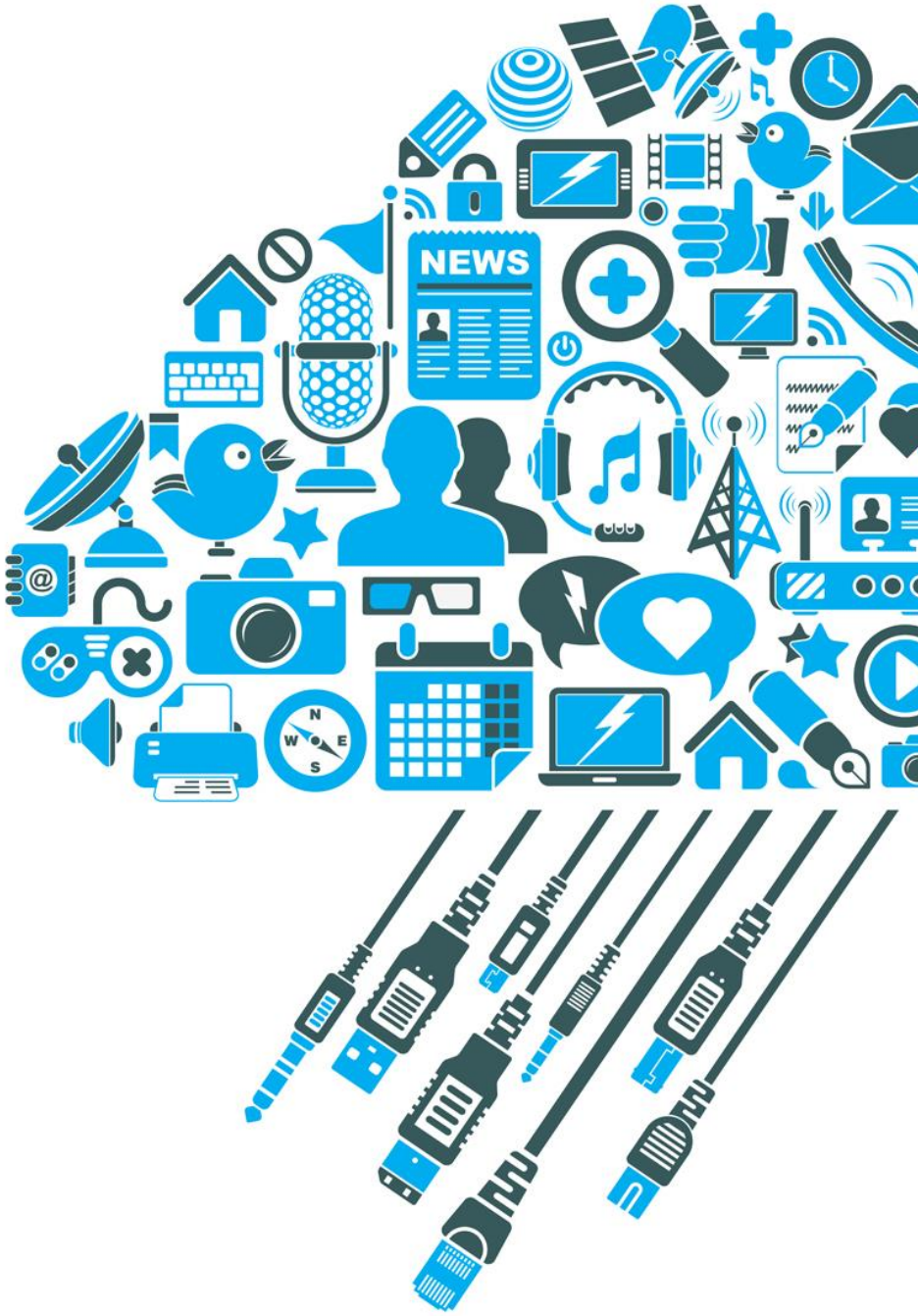
1.

For

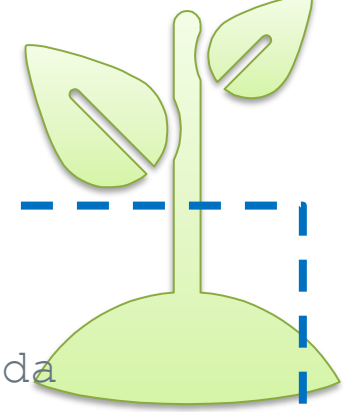
Instrução For

Outro comando de loop extremamente utilizado é o for. A ideia é a mesma do while, fazer um trecho de código ser repetido enquanto uma condição continuar verdadeira. Mas além disso, o for reserva um espaço para inicialização de variáveis e o modificador dessas variáveis. Isso faz com que fique mais legível as variáveis que são relacionadas ao loop.

O é recomendado usar o comando for quando sabemos quantas vezes um loop será executado.



Instrução for

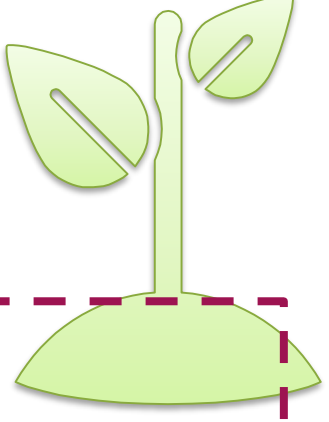


```
//Sintaxe:

for ( inicialização; expressões booleanas; passo da
repetição )
        Instrução_simples;


for ( inicialização; expressões booleanas; passo da
repetição )
{
        instruções;
}
```

Exemplo



```
for(int i=1;i<=10;i++)
    System.out.println(i+" ");
```

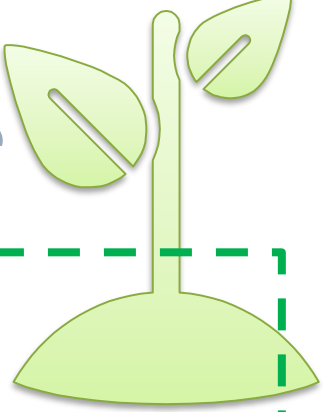
```
// Calculando o fatorial de um número:
int numero = 10;
int fatorial = 1;
for (int i = numero; i > 0; i--)
{
      fatorial = fatorial * i;
}
System.out.println("fatorial de " + valor + " = " +
fatorial);
```

2.

Variações do laço for

O laço for é umas das instruções mais versáteis da linguagem Java porque permite muitas variações.

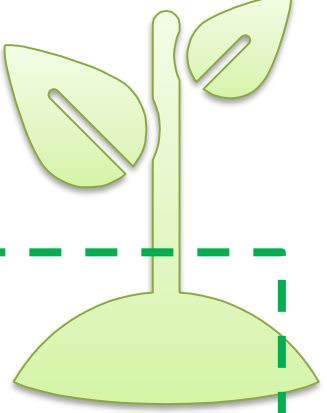
Sem inicializar o contador externamente



```
// Calculando o fatorial de um número:
int numero = 10;
int fatorial = 1;

for (int i = numero; i > 0; i--)
{
        fatorial = fatorial * i;
}
System.out.println("fatorial de " + valor + " = " +
fatorial);
```

Diversas variáveis de controle

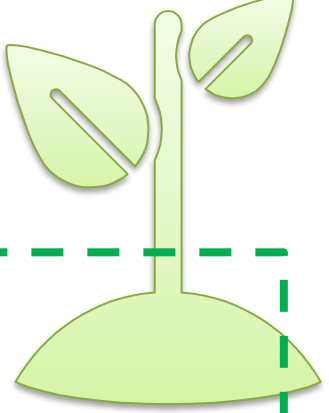


```
//Use vírgulas em uma instrução for.
Class VariasVar{
   public static void main(String args[]){
       for(int sobe=1, desce=10 ; sobe<=10 && desce>=1;
sobe++, desce--){
            System.out.printf("%d \t %d \n", sobe, desce);
        }
    }

}
```

```
Saída:
1       10                      6       5
2       9                       7       4
3       8                       8       3
4       7                       9       2
5       6                       10      1
```

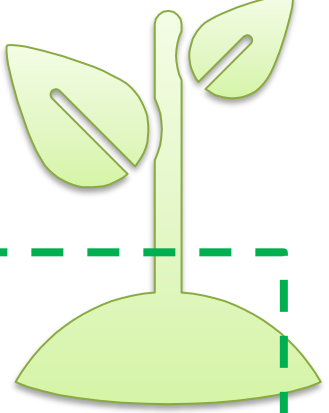
Partes ausentes



```
//Partes de for podem estar vazias.
Class Empty{
   public static void main(String args[]){
        int i;
        for(i=0; i<10){
           System.out.println("Pass #" + 1);
           i++; //Incrementa a variável de controle do laço
        }
   }
}
```

```
Saída:
Pass #0                    Pass #5
Pass #1                    Pass #6
Pass #2                    Pass #7
Pass #3                    Pass #8
Pass #4                    Pass #9
```

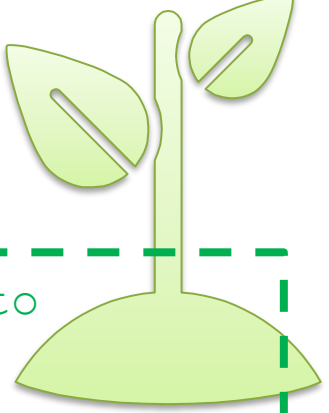
Partes ausentes II



```
//Partes de for podem estar vazias.
Class Empty2{
   public static void main(String args[]){
       int i;
       i=0; //move a inicialização para fora do laço
       for(; i<10;){
          System.out.println("Pass #" + i);
          i++; //Incrementa a variável de controle do laço
       }
   }
}
```

```
Saída:
Pass #0
Pass #1
Pass #2
Pass #3
Pass #4
Pass #5
Pass #6
Pass #7
Pass #8
Pass #9
```

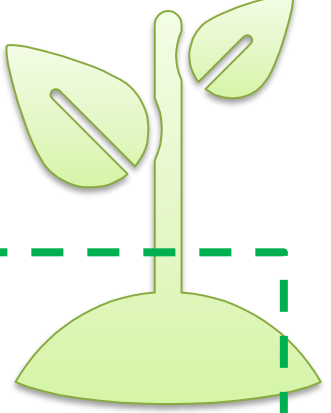
Laço infinito



```
//Necessário o uso break para parar o laço infinito
Class Empty3{
   public static void main(String args[]){
       int i = 0;
       for(;;){
          System.out.println("Pass #" + i);
          i++; //Incrementa a variável de controle do laço
          if(i>=10) break;
       }
   }
}
```

```
Saída:
Pass #0
Pass #1
Pass #2
Pass #3
Pass #4
Pass #5
Pass #6
Pass #7
Pass #8
Pass #9
```

Laço sem corpo



```
//O corpo do laço pode estar vazio
Class Empty3{
   public static void main(String args[]){
        int i;
        int sum = 0;
        for(i = 1; i <= 5; sum += i++);

        System.out.println("Sum é " + sum);
   }
}
```

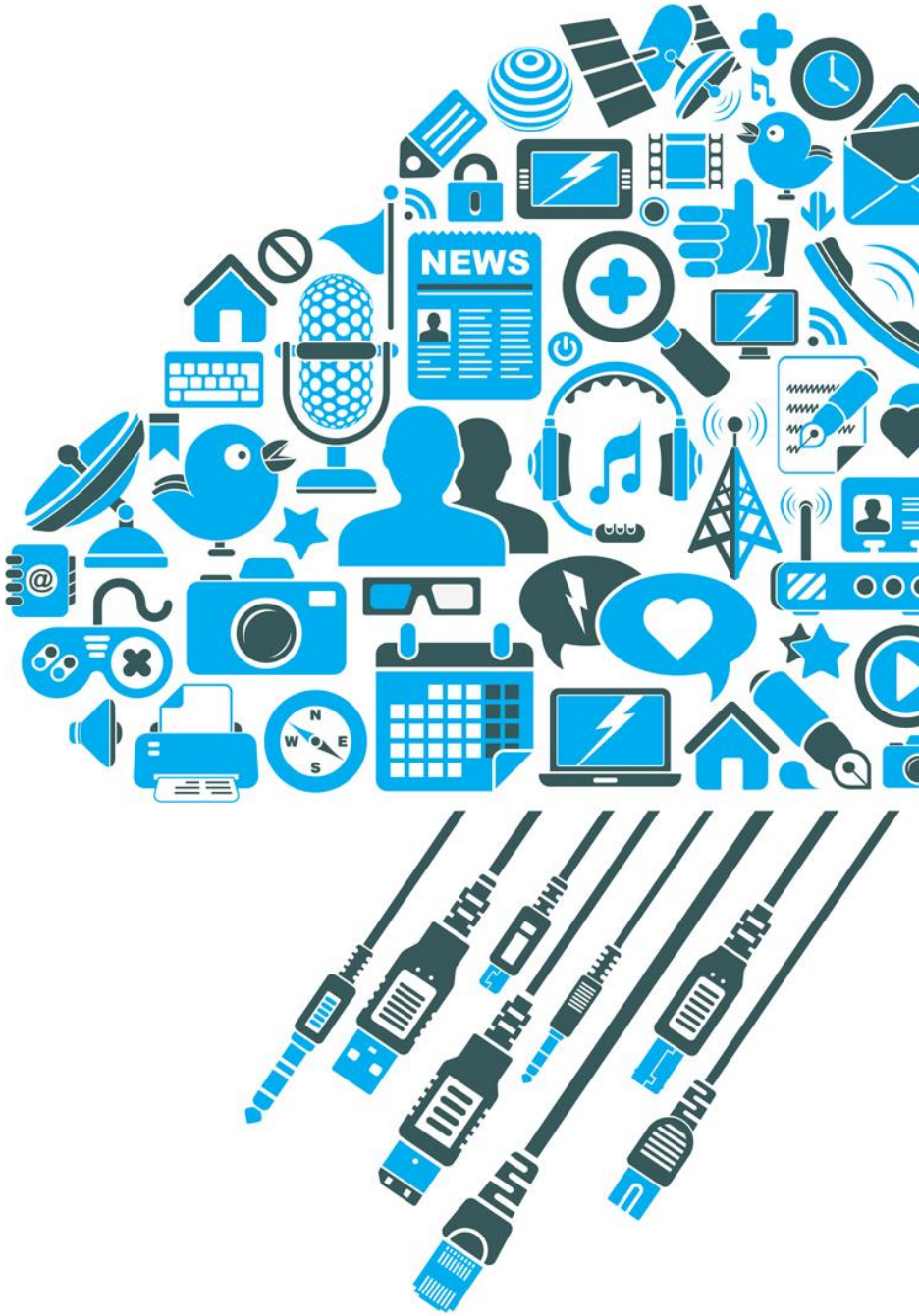
```
Saída:

Sum é 15
```

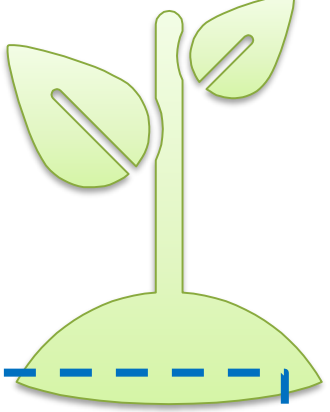
1.
Enhanced-for

Enhanced-for

O enhanced-for foi introduzido a partir do Java 5, e é utilizado para realizar as varreduras em collections. Para cada iteração do for, o elemento da iteração é atribuído à variável. Utilizando o enhanced-for, você é obrigado a percorrer um array por exemplo.



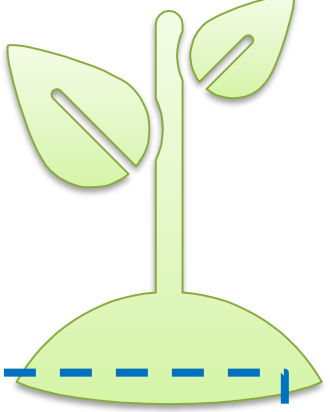
Instrução enhanced-for



```
Public class teste{
   public static void main(String args[]){
       int[] array = {1, 2, 3, 4, 5};

       for (int i : array){
         System.out.println(i);
       }
   }
}
```

Enhanced-for em Collections



```
Public class teste{
   public static void main(String args[]){
       String stringArray = {“one”, “two”, “three”};

       for (String i : stringArray){
         System.out.println(i);
       }
   }
}
```

DESAFIO

E aí, vamos praticar?

Fibonacci Fácil

A seguinte sequência de números 0 1 1 2 3 5 8 13 21... é conhecida como série de Fibonacci. Nessa sequência, cada número, depois dos 2 primeiros, é igual à soma dos 2 anteriores. Escreva um algoritmo que leia um inteiro N (N < 46) e mostre os N primeiros números dessa série.

Entrada: O arquivo de entrada contém um valor inteiro N (0 < N < 46).

Saída: Os valores devem ser mostrados na mesma linha, separados por um espaço em branco. Não deve haver espaço após o último valor.

Entrada:	**Saída:**
5	0 1 1 2 3

Número Perfeito

Na matemática, um número perfeito é um número inteiro para o qual a soma de todos os seus divisores positivos próprios (excluindo ele mesmo) é igual ao próprio número. Por exemplo o número 6 é perfeito, pois 1+2+3 é igual a 6. Sua tarefa é escrever um programa que imprima se um determinado número é perfeito ou não.

Entrada: A entrada contém vários casos de teste. A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 20$), indicando o número de casos de teste da entrada. Cada uma das N linhas seguintes contém um valor inteiro X ($1 \leq X \leq 108$), que pode ser ou não, um número perfeito.

Entrada:	**Saída:**
6	6 eh perfeito
5	5 nao eh perfeito
28	28 eh perfeito

Obrigado!

Alguma pergunta?

Você pode me contatar em:
ywassef@hotmail.com