




# CURSO DE PROGRAMAÇÃO EM JAVA

Aula 7   
Estruturas de  
repetição



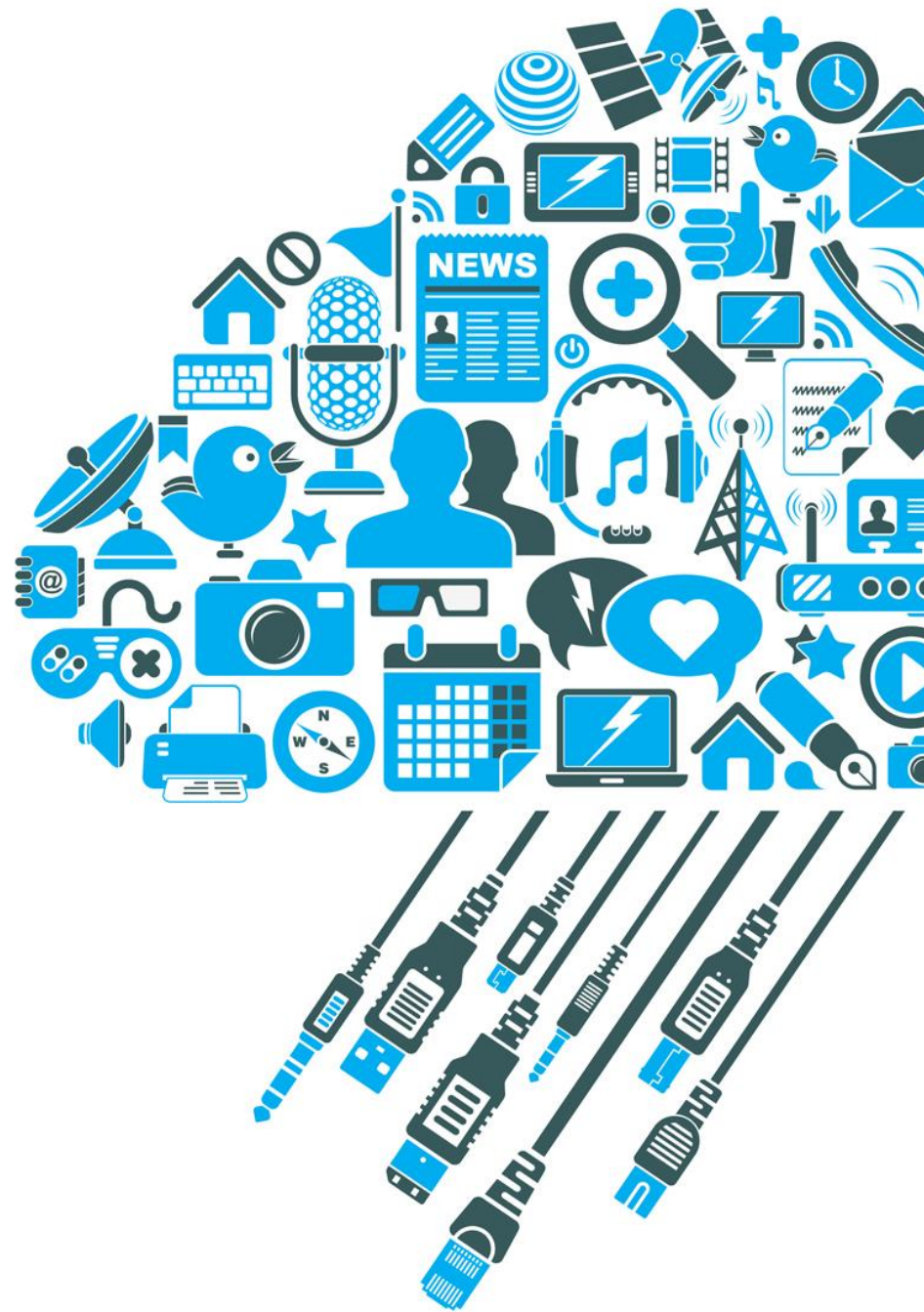
*As estruturas de repetição também são conhecidas como laços (loops) e são utilizados para executar, repetidamente, uma instrução ou bloco de instrução enquanto determinada condição estiver sendo satisfeita.*

# 1. While

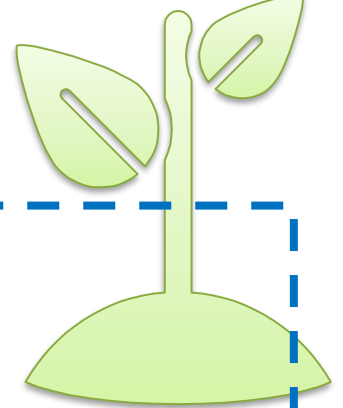
## Instrução while

O termo `while` pode ser traduzido para o português como “enquanto”. Este termo é utilizado para construir uma estrutura de repetição que executa, repetidamente, uma única instrução ou um bloco delas “enquanto” uma expressão booleana for verdadeira.

O comando while deve ser usado sempre que não sabemos quantas vezes um loop será executado.



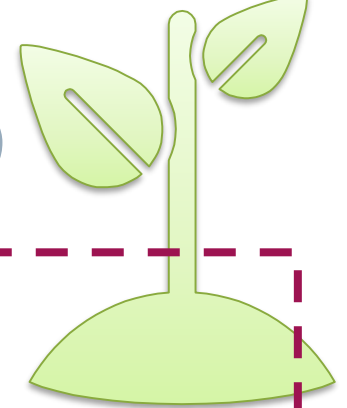
# Instrução while



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

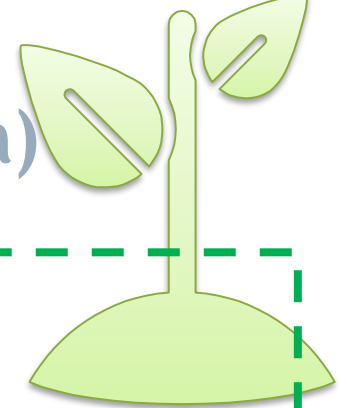
```
public class contando {  
    public static void main(String[] args) {  
        int count=1;  
  
        while (count<=10) {  
            System.out.println(count);  
            count++;  
        }  
    }  
}
```

# Fazendo uma PA (progressão aritmética)



```
public class pa {  
    public static void main(String[] args) {  
        int inicial=1,  
            razao=3,  
            an=inicial,  
            valor_max=20;  
  
        System.out.printf("Elementos da PA, de valor  
inicial %d e razão %d, menores que %d\n", inicial, razao,  
valor_max );  
        while(an<=valor_max) {  
            System.out.println(an);  
            an += razao;  
        }  
    }  
}
```

# Fazendo uma PG (progressão geométrica)



```
public class pg {  
    public static void main(String[] args) {  
        int inicial=1,  
            quociente=2,  
            gn=inicial,  
            valor_max=32;  
  
        System.out.printf("Elementos da PG, de valor  
inicial %d e razão %d, menores que %d\n", inicial,  
quociente, valor_max );  
        while(gn<=valor_max) {  
            System.out.println(gn);  
            gn *= quociente;  
        }  
    }  
}
```

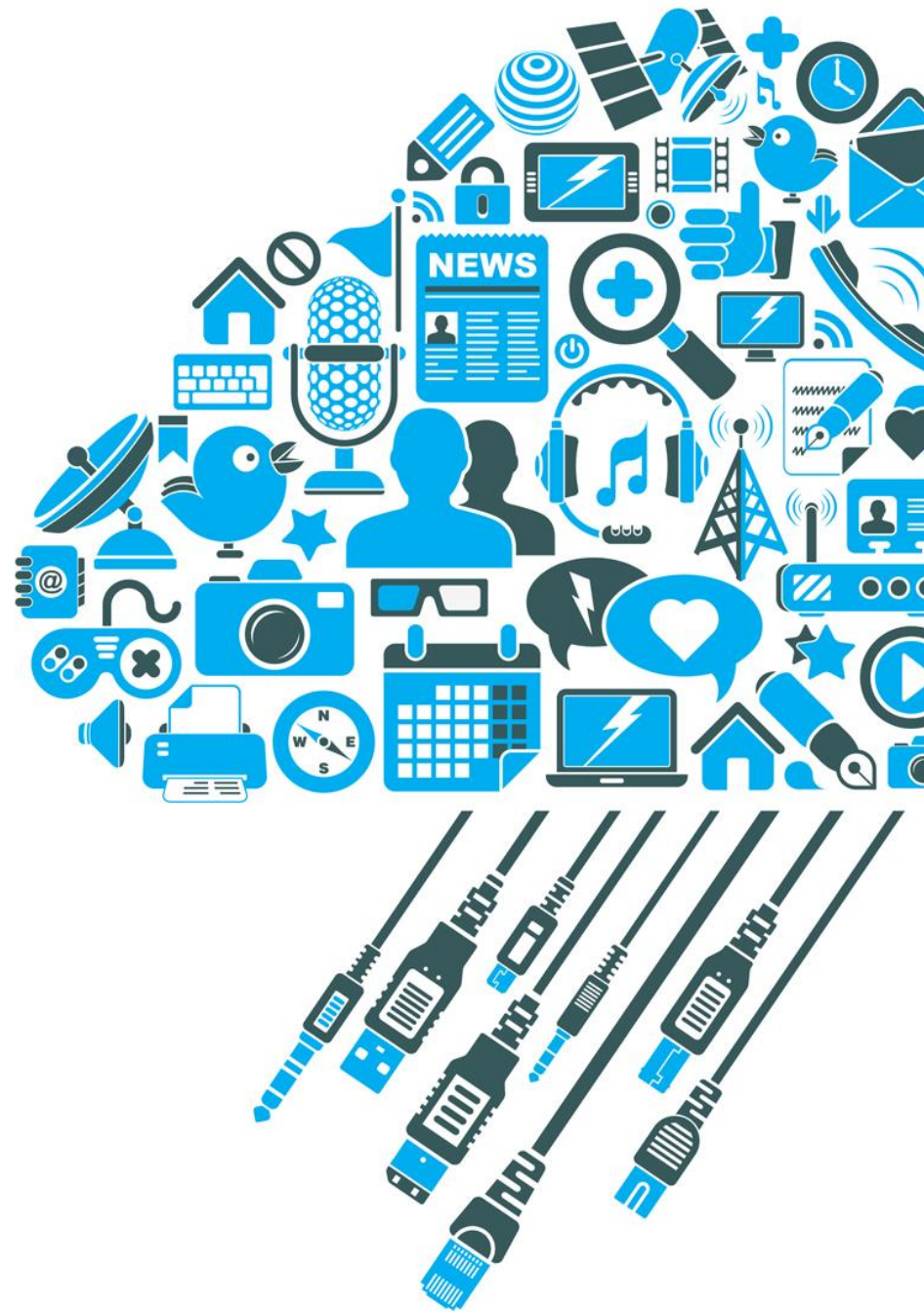
2.

Do While

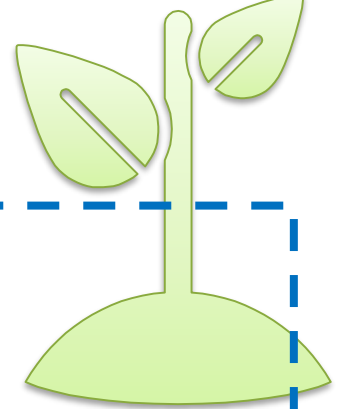


## Instrução while

A estrutura de repetição do-while é uma variação da estrutura do-while. Em um laço while, a condição é testada antes da primeira execução das instruções que compõem seu corpo. Desse modo, se a condição for falsa na primeira vez em que for avaliada, as instruções desse laço não serão executadas nenhuma vez. Em um laço **do-while**, por outro lado, a condição somente é avaliada depois que suas instruções são executadas pela primeira vez, assim, mesmo que a condição desse laço seja falsa antes de ele iniciar, suas instruções serão executadas pelo menos uma vez.



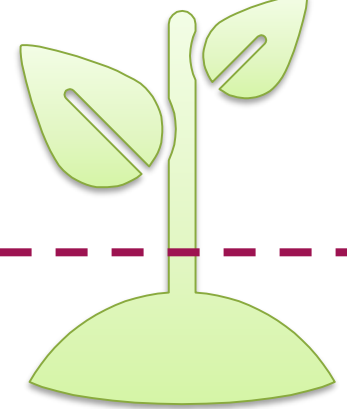
# Instrução do-while



```
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);  
    }  
}
```

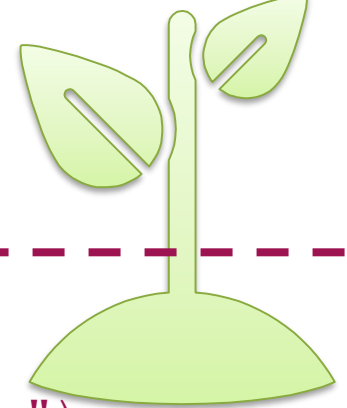
# Menu de opções



```
import java.util.Scanner;

public class DoWhile {
    public static void main(String[] args) {
        boolean continuar=true;
        int opcao;
        Scanner entrada = new Scanner(System.in);
        do
        {
            System.out.println("\t\tMenu de opções do curso
Java Progressivo:");
            System.out.println("\t1. Ver o menu");
            System.out.println("\t2. Ler o menu");
            System.out.println("\t3. Repetir o menu");
            System.out.println("\t4. Tudo de novo");
            System.out.println("\t5. Não li, pode repetir?");
        }
```

# Menu de opções



```
        System.out.println("\t0. Sair");

        System.out.print("\nInsira sua opção: ");
        opcao = entrada.nextInt();

        if(opcao == 0){
            continuar = false;
            System.out.println("Programa finalizado.");
        }
        else{
            System.out.printf("\n\n\n\n\n\n");
        }

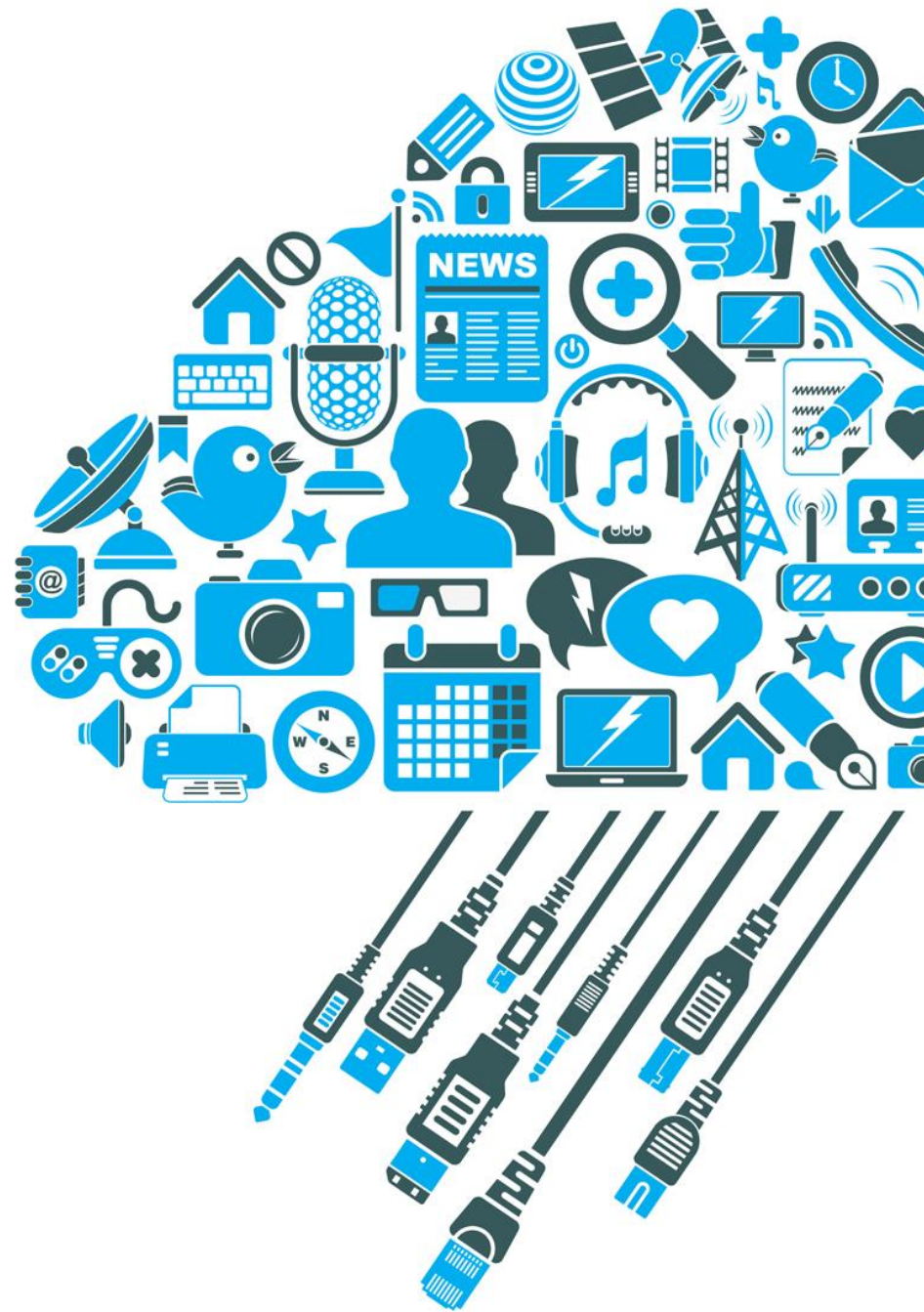
    } while( continuar );
}
```

3.

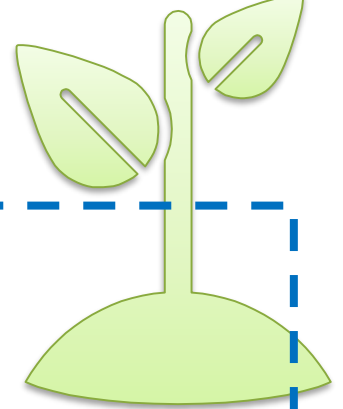
Break

## Instrução break

O comando *break* é um comando bastante importante no desenvolvimento da maior parte dos programas de computador, ele é usado para sair imediatamente de um laço (loop, em inglês), independente do valor de CONDIÇÃO. Ele pode ser executado dentro de um *while*, *for*, *do-while* ou *switch*, fazendo uma saída imediata dessa instrução. Passando para a execução do próximo comando.



# 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);  
    }  
}
```

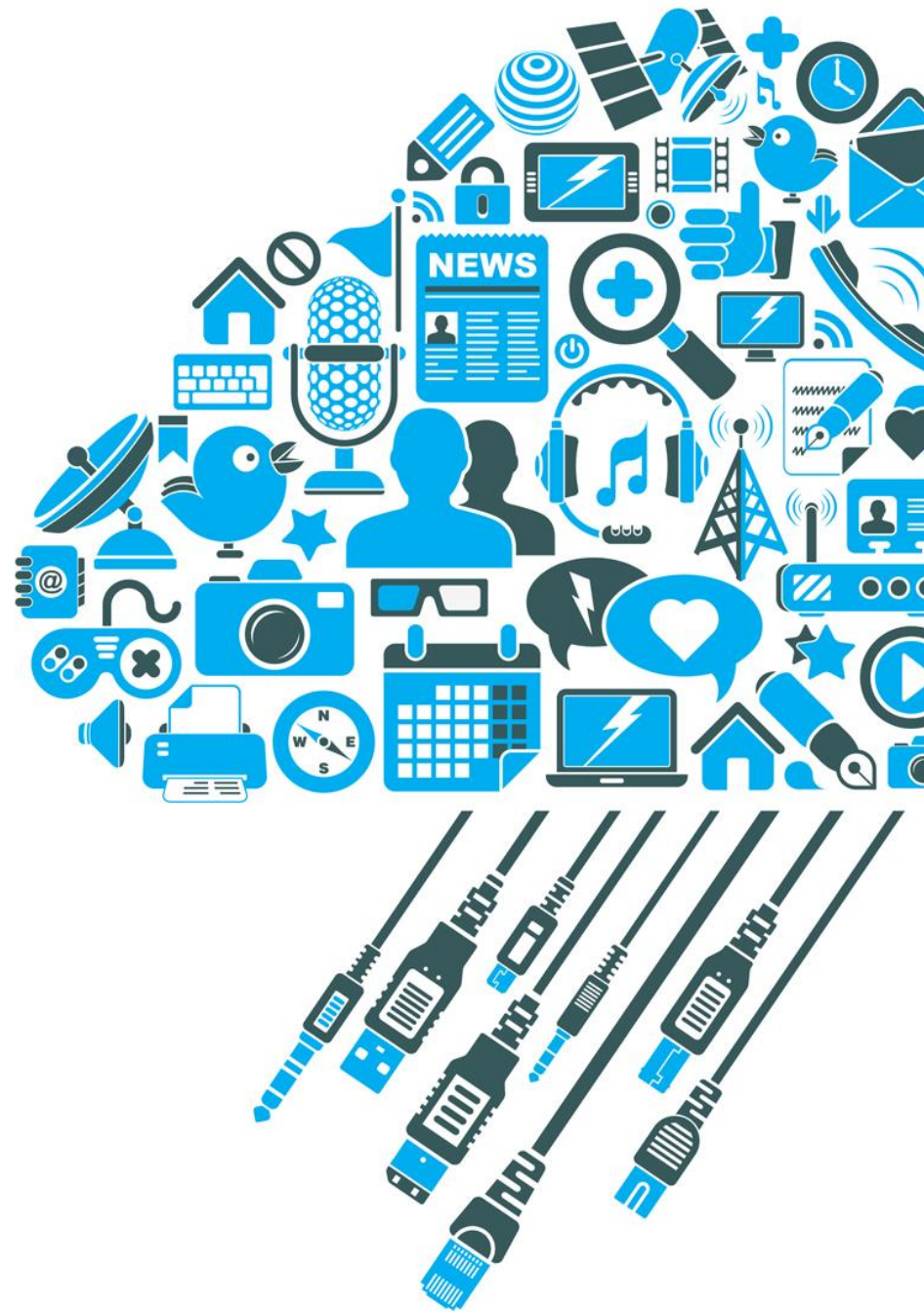
4.

Continue

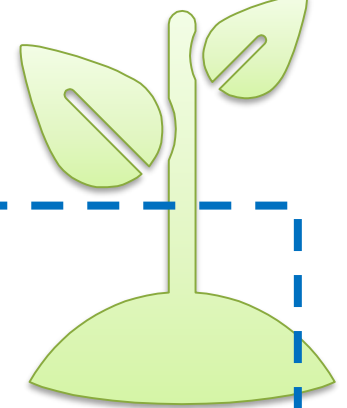


## Instrução continue

Enquanto a instrução `break` é utilizada para encerrar um laço, a instrução **`continue`** serve para iniciar uma **nova repetição** em que todas as instruções tenham sido executadas. Em laços `while` e `do-while`, uma instrução `continue` desvia o fluxo de execução para a condição. Em um laço `for`, ela desvia o fluxo de execução para a iteração e, em seguida, a condição é lida novamente.



# Instrução break

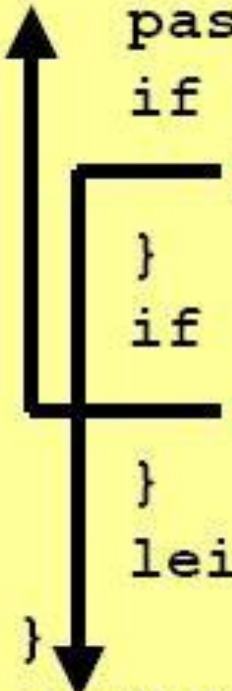


```
continue;
```

```
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+" ");  
            //Serão impressos os números ímpares entre 1  
e 10  
        }  
    }  
}
```

# Exemplo

```
while (!terminado) {  
  passePagina();  
  if (alguemChamou == true) {  
    break;           // caia fora deste loop  
  }  
  if (paginaDePropaganda == true) {  
    continue;       // pule esta iteração  
  }  
  leia();  
}  
restoDoPrograma();
```

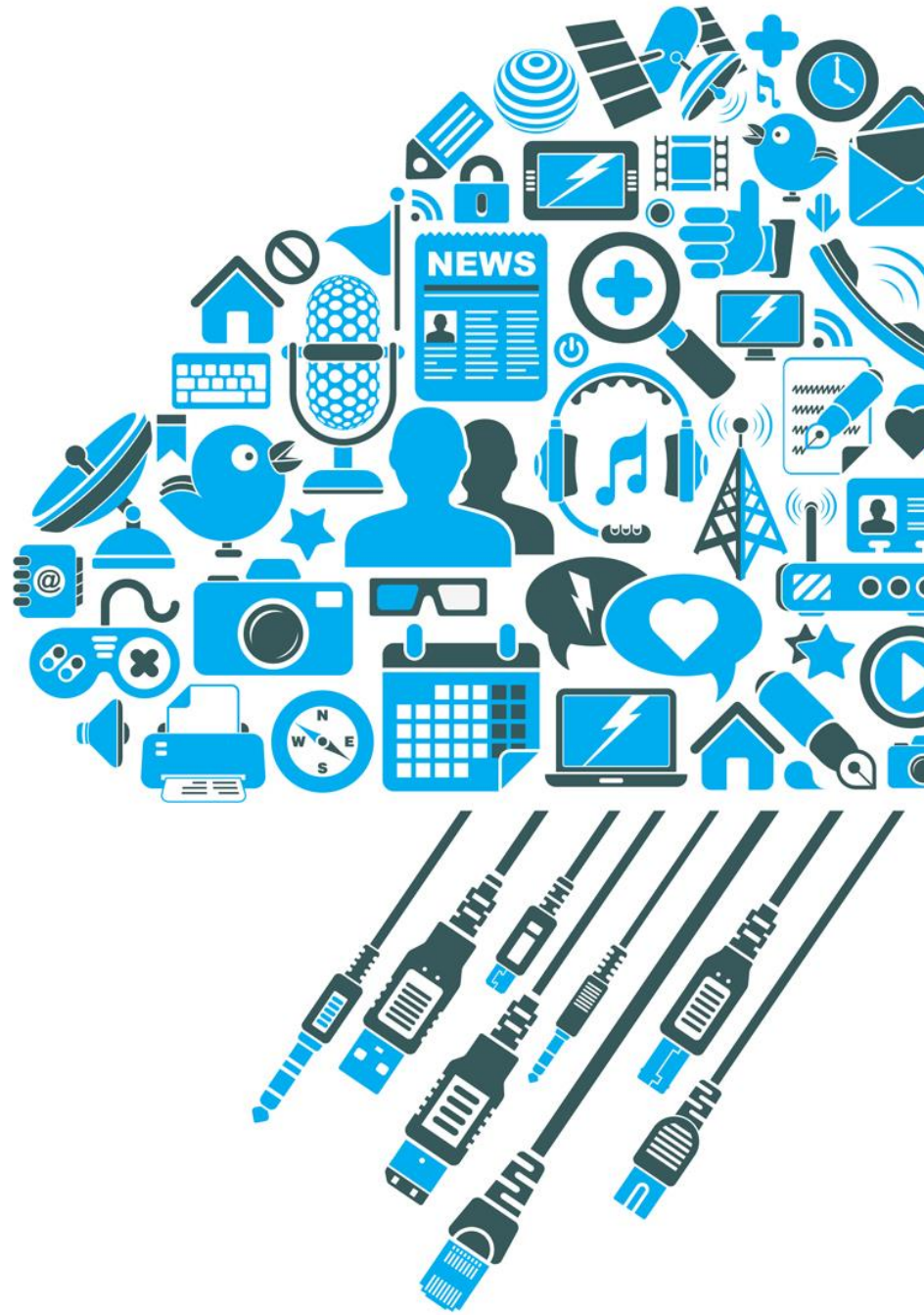


The diagram shows the execution flow of the code. A vertical arrow on the left indicates the loop's progression. A horizontal arrow from the `break;` statement points to the top of the loop, indicating an exit. A horizontal arrow from the `continue;` statement points back to the start of the loop body, indicating a skip to the next iteration.

## Rótulos (break e continue)

E se tivermos com um laço dentro de outro e quisermos quebrar o laço mais

externo? O break ou continue agem no laço mais interno. Existe uma maneira: rotular nossos laços. Observe a imagem a seguir:

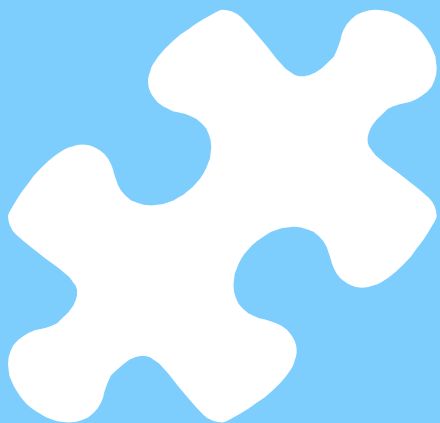




# Exemplo

```
revista: while (!terminado) {  
    for (int i = 10; i < 100; i += 10) {  
        passePagina();  
        if (textoChato) {  
            break revista;  
        }  
    }  
    maisInstrucoes();  
}  
restoDoPrograma();
```

*break sem rótulo  
quebraria aqui!*



# DESAFIO

E aí, vamos praticar?

# Número Primo

Na matemática, um Número Primo é aquele que pode ser dividido somente por 1 (um) e por ele mesmo. Por exemplo, o número 7 é primo, pois pode ser dividido apenas pelo número 1 e pelo número 7.

**Entrada:** A entrada contém vários casos de teste. A primeira linha da entrada contém um inteiro  $N$  ( $1 \leq N \leq 100$ ), indicando o número de casos de teste da entrada. Cada uma das  $N$  linhas seguintes contém um valor inteiro  $X$  ( $1 < X \leq 107$ ), que pode ser ou não, um número primo.

**Saída:** Para cada caso de teste de entrada, imprima a mensagem "X eh primo" ou "X nao eh primo", de acordo com a especificação fornecida.

**Entrada:**

3  
8  
51  
7

**Saída:**

8 nao eh primo  
51 nao eh primo  
7 eh primo

# Tempo de Jogo

Ler um valor N. Calcular e escrever seu respectivo fatorial. Fatorial de N =  $N * (N-1) * (N-2) * (N-3) * ... * 1$ .

**Entrada:** A entrada contém um valor inteiro N ( $0 < N < 13$ ).

**Saída:** A saída contém um valor inteiro, correspondente ao fatorial de N.

**Entrada:**

4

3

**Saída:**

24

6



# Obrigado!

## **Alguma pergunta?**

Você pode me contatar em:  
[ywassef@hotmail.com](mailto:ywassef@hotmail.com)