



## Estrutura de Repetição

---

Lógica de Programação I

## Estrutura de Repetição - *for* e *while*

Java possui duas estruturas de controle de repetições: *for* e *while* (*do-while* é uma variação desta última). Enquanto a primeira executa uma quantidade pré-definida conhecida de repetições, a outra define repetições baseadas em uma condição booleana.

## Estrutura de Repetição - *for* e *while*

While/Do While

A estrutura `while` executa continuamente um bloco de código baseado na avaliação de uma expressão booleana (verdadeiro/falso). Sua sintaxe-base é:

```
while (expressão) {  
    // statement(s)  
}
```

## Estrutura de Repetição - *for* e *while*

While/Do While

Assim como fornecemos uma variável booleana para a estrutura condicional *if*, aqui também podemos fornecer uma única variável ou uma expressão completa.

Considere o trecho a seguir,

```
class WhileDemo {  
    public static void main(String[] args){  
        int count = 1;  
        while (count < 11) {  
            System.out.println("Count is: " +  
count);  
            count++;  
        }  
    }  
}
```



## Estrutura de Repetição - *for* e *while*

While/Do While

Mas, o que acontece quando não temos um valor inicial? Esse é o caso especial da variação **do-while**.

Enquanto no caso padrão da estrutura **while** a primeira execução já está sujeita a avaliação da expressão booleana (ou seja, o bloco pode ser executado 0..\* repetições). Temos que para a variação **do-while** o bloco será executado no mínimo 1 vez, visto que a avaliação da expressão de repetição será apenas ao final do bloco.

# Estrutura de Repetição - *for* e *while*

While/Do While

```
class DoWhileDemo {  
    public static void main(String[] args){  
        int count = 1;  
        do {  
            System.out.println("Count is: " +  
count);  
            count++;  
        } while (count < 11);  
    }  
}
```

## Estrutura de Repetição - *for* e *while*

### Laço de Repetição **FOR**

A estrutura de repetição **for** tem a intenção de percorrer um conjunto definido de valores. O laço **for** executa a repetição de um bloco de código repetidas vezes até satisfazer uma determinada condição. A sintaxe básica é

```
for (inicialização; terminação; incremento) {  
    instrução()  
}
```

## Estrutura de Repetição - *for* e *while*

Laço de Repetição **FOR**

Com base nos exemplos vistos na seção anterior (*while*), podemos re-escrever um programa que escreva em console todos os números inteiros entre 1-10.

```
class ForDemo {  
    public static void main(String[] args){  
        for(int i=1; i<11; i++){  
            System.out.println("Count is: " + i);  
        }  
    }  
}
```



## Estrutura de Repetição - *for* e *while*

Laço de Repetição **FOR**

Os laços **for** são especialmente úteis para percorrer *arrays*.  
Considere o seguinte exemplo

```
class ForArrayDemo {  
    public static void main(String[] args){  
        int[] numbers = {1,2,3,4,5,6,7,8,9,10};  
        for(int i=0; i < numbers.length; i++){  
            System.out.println("Array value is: " + numbers[i]);  
        }  
    }  
}
```

## Sintaxe especial do **for** para arrays

Java também oferece uma variação do laço **for**, chamada **foreach**. Embora a sintaxe seja diferente, continuamos usando a palavra-chave **for**. Veja a seguir

```
class EnhancedForDemo {  
    public static void main(String[] args){  
        int[] numbers = {1,2,3,4,5,6,7,8,9,10};  
        for (int item : numbers) {  
            System.out.println("Count is: " + item);  
        }  
    }  
}
```

## Sintaxe especial do **for** para arrays

Neste trecho não temos uma variável de controle. O próprio compilador Java entende que estamos percorrendo um array, e associa cada valor contido no array à variável informada. Ou seja, o código anterior é equivalente a

```
class EnhancedForDemo {  
    public static void main(String[] args){  
        int[] numbers = {1,2,3,4,5,6,7,8,9,10};  
        for (int i=0; i < numbers.length; i++) {  
            int item = numbers[i]; // here is the trick!  
            System.out.println("Count is: " + item);  
        }  
    }  
}
```

# Exercícios



## Exercício de código

### **Exercício 1: do-while - Adivinhe o Número**

Escreva um programa que gera aleatoriamente um número entre 1 e 10. O usuário deve adivinhar o número. O programa deve informar se o número inserido é muito alto, muito baixo ou correto. O jogo continua até que o usuário adivinhe corretamente.

Utilize a biblioteca do java *Random*.

Exemplo:

```
random.nextInt(10) + 1;
```

## Exercício de código

### Exercício 2: for - Tabuada

Escreva um programa que solicita ao usuário para inserir um número e imprime a tabuada desse número de 1 a 10.

Exemplo de saída:

```
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
```

### **Exercício 3: while - Números Primos**

Escreva um programa que solicita ao usuário para inserir um número e verifica se esse número é primo ou não.

Obrigada