
Não tenha
medo das
mudanças...
Coisas boas
se vão...
Para que
melhores
possam
vir!

Desenvolvimento de Aplicações Desktop

Estruturas de Controle (Repetição)

Professor: Charles Leite

Repetição de Instruções

Na resolução de problemas em programação, frequentemente, precisamos repetir uma mesma sequência de instruções várias vezes

Na maioria dos casos, não sabemos de antemão quantas vezes a sequência de instruções será repetida

Repetição de Instruções

A linguagem de programação deve fornecer uma estrutura (comandos) que permita a execução repetida da mesma sequência de instruções para:

- Evitar esforço do programador
- Permitir que o programador não tenha que saber quantas vezes uma sequência de instruções será executada

Estruturas de Repetição

- Permitem repetir diversas vezes uma instrução ou sequência de instruções
  - Cada repetição é chamada de ITERAÇÃO
- São geralmente conhecidas como *loops* (laços)
- Da mesma forma que comandos condicionais, são controladas por expressões booleanas
- Java oferece três tipos de estruturas (comandos) de repetição
  - O laço **`for`**
  - O laço **`while`**
  - O laço **`do-while`**

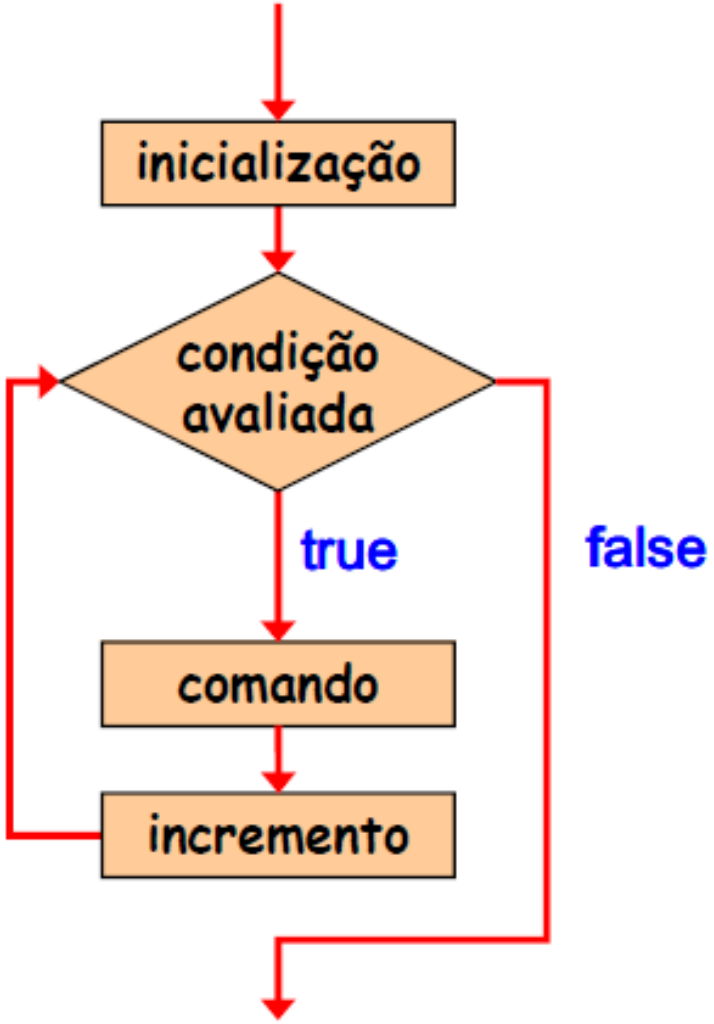
O Comando `for`

```
for(inicialização;condição;incremento)
  corpo
```

- `inicialização` é uma expressão que denota a variável de controle do laço, fornecendo o seu valor inicial
  - É executada somente uma vez
- `condição` booleana, representando uma condição que determina se o laço deve ou não continuar executando
  - É avaliada antes do corpo do laço
- `incremento` modifica o valor da variável de controle para que a condição de continuação do laço, por fim, se torne falsa
  - É executado após cada iteração do laço

O Comando `for`

- Fluxo de controle:



O Comando `for`

- Exemplo 1:

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

Variável valor é inicializada com 0

Comando será realizado enquanto valor for menor ou igual a 10

A cada iteração, valor é incrementado em 1

O Comando `for`

- Exemplo 2:

```
int soma = 0;
int n = leia.nextInt();
for(int valor = 0; valor <=n; valor++){
        soma = soma + valor;
}
System.out.println("A soma é: " + soma);
```

Variável valor é declarada dentro da inicialização e só acessível dentro do for

Se n for menor que 0, não será executado o corpo do for

O Comando `for`

- Exemplo 3:

```
int soma = 0;
int n = leia.nextInt();
for(int valor = n; valor >=0; valor--){
        soma = soma + valor;
        S.println("A soma parcial é: " + soma);
}
S.println("A soma total é: " + soma);
```

Valor agora é decrementado

O Comando `for`

- Exemplo 4:

```
int soma = 0;
int valor;
int n = leia.nextInt();
for(valor = 0; valor <=n; valor = valor + 2){
        soma = soma + valor;
        S.println("A soma parcial é: " + soma);
}
S.println("A soma total é: " + soma);
```

Valor agora é incrementado em 2

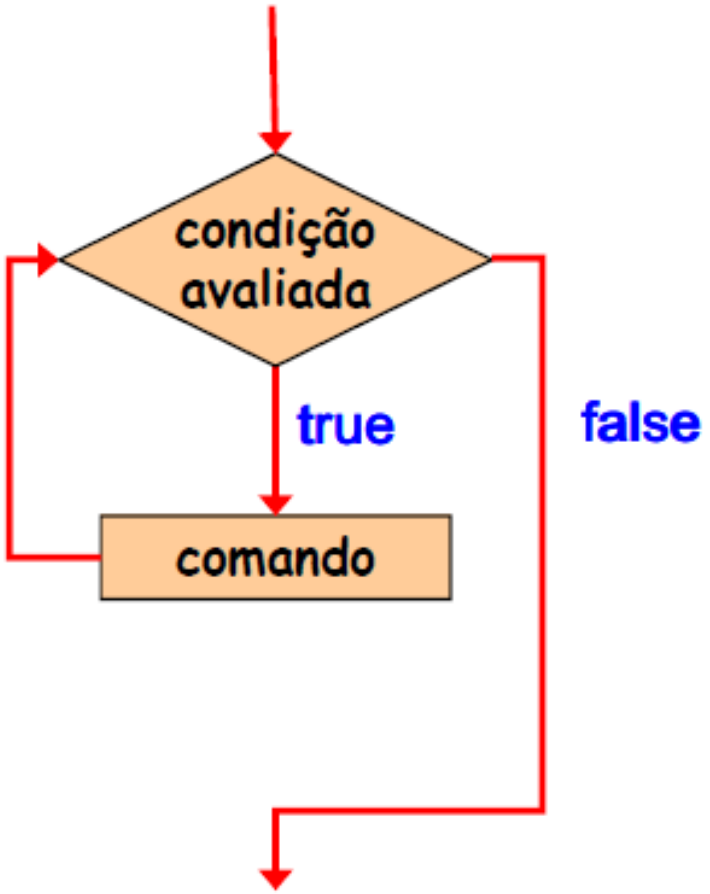
O Comando **while**

```
while (expressãoBooleana)
  corpo
```

- Executa corpo várias vezes até que a avaliação da expressão booleana retorne **false**
- A expressão booleana é avaliada após cada iteração do laço (execução do corpo)
- O corpo não é executado nenhuma vez caso a primeira avaliação da expressão booleana retorne **false**

O Comando `while`

- Fluxo de controle:



O Comando `while`

- Exemplo 1:

```
int soma = 0;
int valor = 0;
int n;
while (valor <= n){
        soma = soma + valor;
        valor++;
}
```

Inicialização da variável de controle é feita fora do laço while

Se n for negativo, não se executa o corpo do while

Incremento da variável de controle é feita no corpo do laço while

O Comando **while**

- Exemplo 1:

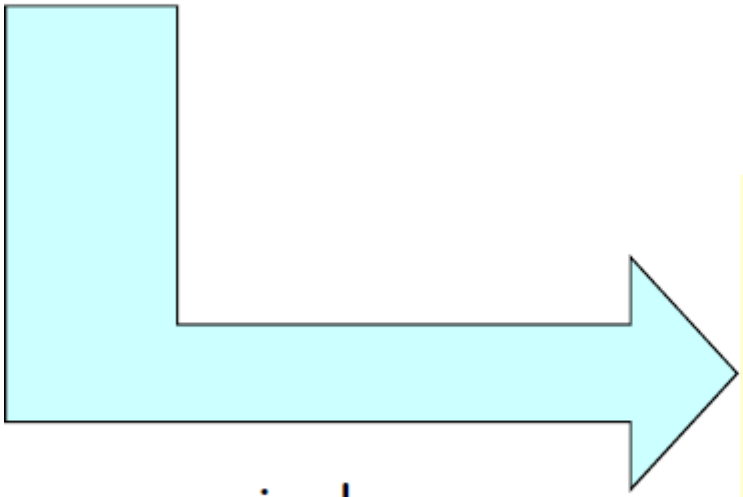
```
int soma = 0;
int valor = 0;
int n;
while (valor <= n){
        soma = soma + valor;

}
```

Se valor não é incrementado, este comando será executado infinitas vezes

Equivalência entre os Comandos **`for`** e **`while`**

```
for(inicialização;condição;incremento)
   corpo
```



equivale a ...

```
inicialização;
while(condição) {
   corpo;
   incremento;
}
```

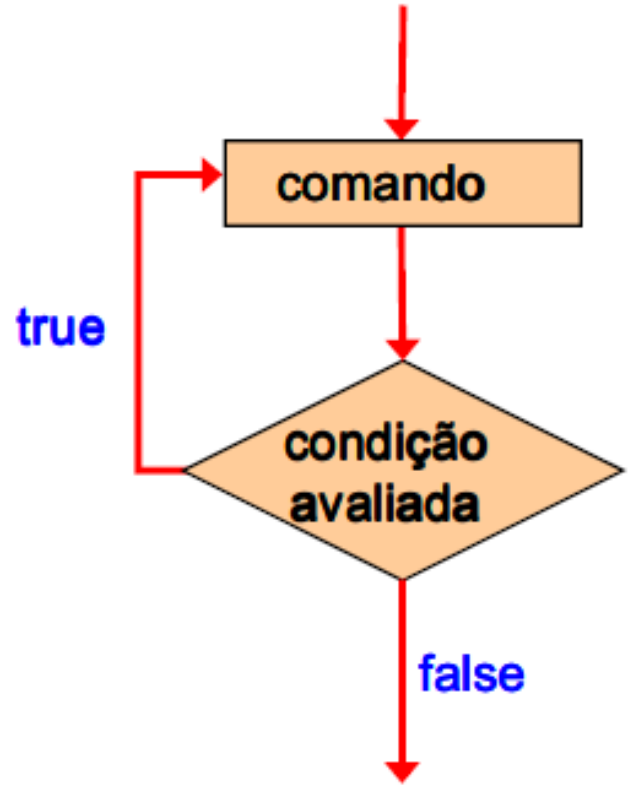
O Comando **do-while**

```
do {
   corpo
} while(expressaoBooleana)
```

- Executa o corpo, pelo menos uma vez, até que a avaliação da expressão booleana retorne **false**
- A expressão é avaliada, toda vez, após cada execução de corpo

O Comando `do-while`

- Fluxo de controle:



O Comando **do-while**

- Exemplo 1:

```
int soma = 0;
int valor = 0;
int n;
do {
        soma = soma + valor;
        valor++;
} while (valor <= n)
```

Inicialização da variável de controle é feita fora do laço while

Incremento da variável de controle é feita no corpo do laço while

Se n for negativo, o corpo do-while é executa pelo menos uma vez

Equivalência entre os Comandos **do-while** e **while**

```
do {
   corpo
} while(expressaoBooleana)
```

Equivalente a ...

```
corpo;
while(expressaoBooleana)
   corpo;
```

Laços Aninhados

Laços podem ser aninhados da mesma forma que comandos condicionais

- O corpo de um laço pode conter outro laço

Para cada iteração do laço externo, o laço interno é completamente executado

Laços Aninhados

- Exemplo 1:

```
int somafor = 0, somawhile = 0;
int n = leia.nextInt();
for(int valorfor = 0; valorfor <=n; valor++){
        int valorwhile = 0;
        while (valorwhile <= n){
        somawhile = somawhile + valorwhile;
        valorwhile++;
        }
        somafor = somafor + somawhile;
}
System.out.println("A soma é: " + somafor);
```

A cada iteração do for, o laço while é executado

Considerações sobre Laços

- Os três tipos de laços são funcionalmente equivalentes
  - Portanto, devem ser usados indiscriminadamente
- O laço **`for`** é geralmente usado quando se sabe de antemão o número de vezes que queremos repetir uma sequência de instruções
- Os laços **`for`** e **`while`** são executados 0 ou muitas vezes
- O laço **`do-while`** é executado 1 ou muitas vezes

O Comando **break**

```
break;
```

- Tem dois usos distintos:
  - Para forçar o término de um laço de repetição (**do-while**, **for** ou **while**)
    - Sequência de instruções é imediatamente finalizada
    - Próxima instrução após a estrutura de repetição é executada
  - Para forçar o término do comando **switch**, independente do **case** que estiver sendo executado

O Comando **break**

- Exemplo 1:

```
int soma = 0;
int valor;
int n = leia.nextInt();
for(valor = 0;   ; valor++){
        soma = soma + valor;
        if (valor == n){
                break;
        System.out.println("A soma parcial é: " + soma);
        }
}
System.out.println("A soma total é: " + soma);
```

Este comando não será executado quando valor==n

DÚVIDAS ...

Referências

DEITEL, Harvey. **Java Como Programar.** *Capítulo 2, páginas 81-108.* 8ª edição. São Paulo: Pearson Prentice Hall, 2010.