Estruturas de Repetição ou Iteração

Prof. Ítalo Assis

Ajude a melhorar este material =]

Encontrou um erro? Tem uma sugestão?

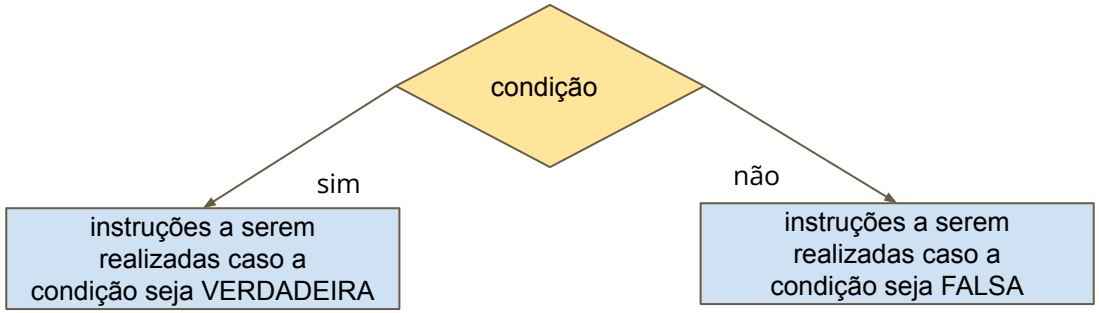
Envie e-mail para italo.assis@ufersa.edu.br

Agenda

- Introdução
- Contadores
- Laço *while*
- Laço do-*while*
- Laço *for*
  - *for each*
- Comandos *break* e *continue*

Introdução

- As estruturas condicionais permitem que o fluxo de execução de um método ou programa seja alterado dependendo de uma condição executando trechos de código e deixando de executar outros.
- Não é possível, com estas instruções, repetir parte do código que foi executado anteriormente ou iterar.



Introdução

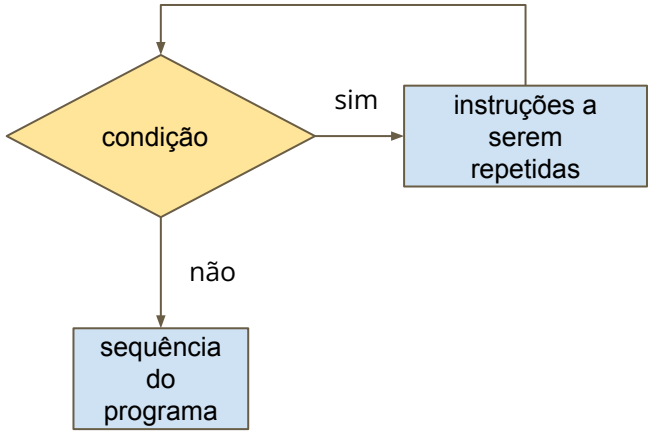
- Em muitas aplicações, há a necessidade de repetir, iterar ou contar:
  - o fatorial de um número é calculado multiplicando-se os valores *de* um *até* o valor especificado;
  - uma sequência de caracteres em uma String representando parte do DNA é comparada com outra se tomando os caracteres *um por um*, *do* primeiro *até* o último;
  - uma impressão de relatórios sobre bolsistas pode ser feita abrindo-se um arquivo, lendo os dados dos bolsistas *um por um*, *enquanto* existirem informações no arquivo;
  - uma autenticação de uso de programas por meio de senha pode ser feita pedindo-se ao usuário que entre a senha *até que* a senha correta seja entrada *ou* o número de tentativas esgotado.

Introdução

- A capacidade de repetição de trechos de programas ou de métodos é uma das características de linguagens de programação que tornam os computadores mais úteis:
  - dada uma tarefa ou rotina, esta pode ser repetida inúmeras vezes de forma automática.
- A repetição por si própria não é muito útil, a não ser que diferentes dados sejam processados a cada iteração da repetição.
  - Considere a tarefa de somar muitos valores ou procurar palavras em um texto longo ou pesquisar vários registros de um banco de dados - tarefas tediosas se feitas por um ser humano, mas que podem ser facilmente implementadas e repetidas por um computador.

Introdução

- As estruturas de repetição são chamadas de **laços** ou **loops**.
- Em suas formas básicas, elas consideram uma **condição** que determina se o laço deve ser executado ou não, e executam um ou mais comandos enquanto a condição especificada for válida ou verdadeira.
- **Cuidado com laços infinitos!**



Introdução

- Uma tarefa inerente à execução de laços é a modificação de variáveis que controlam a execução dos laços (chamadas **variáveis de controle**).
- Estas variáveis podem refletir o **estado da execução de um laço**, por exemplo, representando um contador que contará quantas vezes a repetição será feita, ou verificando se uma condição já foi cumprida para que o laço termine.
- Um caso especial de variáveis de controle são **contadores**.

Contadores

- Variáveis que recebem um valor inicial e são modificadas a cada iteração de uma estrutura de repetição.
- Contadores podem ser variáveis de qualquer tipo numérico.
- Estas variáveis devem:
  - receber um valor inicial
  - ser alteradas a cada iteração do laço
  - ter seu valor verificado a cada iteração a fim de saber se um valor final foi alcançado.
- Por exemplo, para fazer uma estrutura que conte de um até dez, pode-se utilizar um contador que receberá o valor inicial de um, sendo acrescido de um em um até que o valor deste contador seja igual a dez, interrompendo então a repetição do laço.

Contadores

- Valores dos contadores são alterados através da atribuição do resultado de uma operação à variável que representa o contador.
- Geralmente estas operações envolvem o próprio contador
  - *linha = linha + 1*
- Java tem operadores especiais para a modificação de variáveis usando a própria variável

++	var++ ou ++var	var = var + 1
+=	var += val	var = var + val
--	var-- ou --var	var = var - 1
-=	var -= val	var = var - val
***=**	var *= val	var = var * val
/=	var /= val	var = var / val

Lembra do pré/pós incremento?

- Vamos executar esses códigos e observar seus resultados:
  - ```
    int a = 5;
    int b = 5 + a++;
    System.out.println("a = " + a  + " b = " + b);
    ```
    - a = 6 b = 10
  - ```
    int c = 5;
    int d = 5 + ++c;
    System.out.println("c = " + c  + " d = " + d);
    ```
    - c = 6 d = 11

Contadores

- A alteração de contadores pode levar a problemas quando os valores modificados são muito grandes ou pequenos para ser representados pelos seus tipos.
  - Exemplo:
    - Um valor do tipo *short* pode representar valores entre -32768 e 32767
    - Ao final do código *short cont = 32767; cont += 1;* a variável *cont* valerá -32768
    - Problema é conhecido como **overflow**
    - O compilador Java não informa este tipo de erro em potencial
- O overflow também acontece com valores de ponto flutuante, exceto que um tratamento mais correto é dado:
  - Ao final do código *float valor = 3.4e38f; valor *= 2;* a variável *valor* valerá *Infinity*, um valor especial que existe para os tipos *double* e *float*.

Contadores

- De forma similar ao overflow, erros potenciais de **underflow** podem ocorrer quando o tipo de ponto flutuante não é capaz de representar um número muito pequeno (próximo de zero)
- O compilador aproximará estes valores para zero e não indicará erros

Estruturas de repetição

- Repete um comando ou bloco de comandos enquanto uma condição for verdadeira
- A condição deve ser um valor booleano ou expressão cujo resultado seja booleano
- As diferentes sintaxes utilizadas para representar laços são intercambiáveis

Laço *while*

- O bloco ou comando associado ao laço será repetido enquanto o valor booleano avaliado pela instrução *while* a cada iteração for *true*.
- Se o argumento para a instrução *while* for inicialmente *false*, o comando ou bloco de comandos associado não será executado nem mesmo uma vez.

```
while (condição) instrucao_a_ser_repetida;
```

```
while (condição) {
  // instruções a serem
  // repetidas
}
```

Exemplo

- Escreva um programa em Java que imprima a velocidade em metros por segundo, milhas por horas e pés por segundo correspondentes às velocidades em quilômetros por hora, de zero a cinquenta, de meio em meio quilômetro por hora.
- A conversão das unidades de velocidade segue a lista abaixo.
  - 1 quilômetro por hora = 0.2778 metros por segundo
  - 1 quilômetro por hora = 0.6214 milhas por hora
  - 1 quilômetro por hora = 0.9113 pés por segundo

Exemplo

- Crie um programa para receber do usuário os dados de um cartão e verificar se os dados são válidos.
  - Os dados que o programa deve receber são nome, número, código e validade (mês e ano).
  - A validação consiste em verificar se a validade é maior que a data atual.
  - Caso não seja, deve-se solicitar novamente ao usuário a inserção da validade até que sejam informados dados válidos.
  - O programa deve exibir uma mensagem caso os dados sejam validados.

Laço *do-while*

- Java oferece outro tipo de laço que é executado enquanto uma condição for verdadeira, mas garante que o bloco associado ao laço será executado ao menos uma vez.
- A condição é avaliada ao final do laço.

```
do {
  // instruções a serem
  // repetidas
} while (condição);
```

Exemplo

- No exemplo anterior, troque as estruturas *while* por *do-while*

Exemplo

- Escreva um programa que implementa um jogo simples de adivinhação de números.
  - O número a ser adivinhado (alvo) deve ser definido no código do programa
  - O usuário deve fazer tentativas para acertar o número
  - A cada tentativa do usuário, o programa deve dizer se o alvo é maior ou menor que o número atual
  - O programa deve encerrar após cinco tentativas ou no caso do usuário acertar o valor

Laço *for*

- O Java tem uma estrutura especializada para a implementação de repetição controlada por contadores, que agrupa a inicialização, modificação e comparação da variável de controle em uma única instrução.

```
for (inicialização; verificação_de_condições; atualização) instrucao_a_ser_repetida;
```

```
for (inicialização; verificação_de_condições; atualização) {
  // instruções a serem
  // repetidas
}
```

Laço *for*

- Quando variáveis de controle são usadas somente para controlar o contador de um laço for, é comum declarar as variáveis dentro da expressão *inicialização*

  for (int i=0; i<10; i++)
- Nesse caso, o escopo da variável fica sendo o laço
- A variável não será mais definida e não poderá ser usada assim que o laço for encerrado

Exemplo

- Escreva um programa que receba um número *n* e calcule e exiba o fatorial de cada número de zero a *n*.

Comandos *break* e *continue*

- A instrução *break*, quando executada em um *while*, *for*, *do-while* ou *switch*, ocasiona a saída imediata desta instrução.
    - A execução continua com a primeira instrução depois da instrução de controle.
- Já a instrução *continue*, quando executada em um *while*, *for* ou *do-while*, pula as instruções restantes no corpo do laço e prossegue com a próxima iteração.

Exemplo

- Execute o código a seguir:

```
for (int i = 1; i <= 10; i++) {
                if (i == 3) continue;
                    if (i == 8) break;
                        System.out.println(i);
        }
```

- Como os comandos *break* e *continue* afetam a contagem dos números?

Laço *for each*

- O Java possui uma sintaxe especial que pode ser utilizada em alguns casos para acessar elementos de um vetor
- Veremos esse laço adiante no curso

Os códigos relacionados a esta aula estão disponíveis em

https://github.com/italoaug/Programacao-Orientada-a-Objetos/tree/main/codigos/repeticao

Referências

SANTOS, R. **Introdução à programação orientada a objetos usando JAVA**. 2. ed. Rio de Janeiro: Campus, 2013. 336p.

DEITEL, Paul; DEITEL, Harvey. **Java: como programar**. 10. ed. São Paulo: Pearson Education do Brasil, 2017.