Programação Estruturada

Aula 05 - Comandos de iteração

Material Didático do Instituto Metrópole Digital - IMD
Versão 5.0 - Todos os Direitos reservados

Apresentação

Na aula anterior, você aprendeu a utilizar os comandos de seleção, lembra? O comando if testa uma expressão boleana. Já o comando switch verifica expressões como números, letras, etc. Esses dois comandos são utilizados para decidir qual bloco de código deverá ser executado, certo?

Entretanto, existem situações nas quais você precisará repetir a execução de um mesmo bloco de código. Por exemplo, para calcular a nota de todos os alunos de uma sala, você precisará executar o código de cálculo de média repetidamente, uma vez para cada aluno da sala. Nesta aula, vamos apresentar os comandos utilizados em tais situações, os chamados de comandos de iteração.

Faça uma boa leitura!

Vídeo 01 - Apresentação

Objetivos

Ao final desta aula, você será capaz de:

- Descrever o funcionamento dos comandos de iteração.
- Saber aplicar corretamente comandos de iteração, visando executar repetidas vezes determinados trechos de código.

1. Comandos de Iteração

No módulo Lógica de Programação e Algoritmos você já estudou sobre comandos de iteração, lembra? Agora, você vai estudar cada um deles aplicados na linguagem Java, sendo seus nomes *for, while* e *do-while*. E sempre que você tiver dúvida, releia, compare, releia aulas anteriores deste módulo ou de módulos anteriores, se precisar. E tire também suas duvidas na plataforma digital!

Comecemos então pelo termo iteração. O termo “iteração” significa “repetição”. Os comandos que você aprenderá nesta aula são utilizados em situações em que você precisa que o computador execute certos trechos de um programa de forma repetida — ou seja, várias vezes, até que uma determinada condição seja alcançada. Alguns autores chamam essas estruturas de “laços de repetição” ou, em inglês, de *loops*.

Você perceberá que cada um dos comandos de iteração disponíveis na linguagem Java são de maneira geral equivalentes, mas cada um deles pode ser utilizado mais facilmente em situações específicas. Então, fique atento para saber fazer a escolha de quando utilizar cada um deles.

2. O Laço for

O primeiro comando de iteração que você irá aprender é o laço **for**. Imagine se você precisasse imprimir os números de 1 a 10, como você faria? Uma possibilidade é escrever o seguinte código:

```
public class ProgramaFor {
    public static void main(String[] args) {
        System.out.println("1");
        System.out.println("2");
        System.out.println("3");
        System.out.println("4");
        System.out.println("5");
        System.out.println("6");
        System.out.println("7");
        System.out.println("8");
        System.out.println("9");
        System.out.println("10");
    }
}
```

Existem, porém, dois problemas nessa solução. A primeira é que há uma duplicação de código onde comandos similares são usados para imprimir na tela várias vezes, mudando apenas o valor passado como parâmetro. O segundo problema é que esse código tem um comportamento fixo, ou seja, não é possível escolher a quantidade de vezes que o comando de imprimir na tela será executado depois que o código for compilado e estiver em execução. Deve haver uma forma de receber do usuário um número positivo N e imprimir de 0 (zero) até esse número N.

Para evitar duplicação de código e para permitir que seja configurável, em tempo de execução do programa, a quantidade de vezes na qual o comando vai ser executado, usamos os chamados comandos de iteração (laço). O primeiro comando que veremos aqui é o comando **for**. Veja como funciona o mesmo código agora implementado com esse comando de iteração:

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

Note que o programa ficou bem mais enxuto em termos de número de linhas de código, certo? Vamos agora entender como funciona esse código mostrado. Para isso, observemos primeiro o formato geral do laço de iteração **for** em Java, o qual é o mesmo encontrado em diversas outras linguagens de programação:

```
for (inicialização; condição; incremento) {
   ... // lista de comando;
}
```

Note que ele começa com uma seção chamada de inicialização. Essa seção de código é geralmente utilizada para declarar uma variável que será utilizada como variável de controle de execução do laço. No exemplo mostrado do ProgramaFor, temos o seguinte código de inicialização:

```
int i = 1;
```

Observe que essa variável está sendo declarada e inicializada ao mesmo tempo. Depois da inicialização da variável, vem a seção de condição. Nessa condição, você deve colocar uma expressão que resulte no valor verdadeiro, enquanto for necessário executar o laço, e falso caso esteja na hora de pará-lo. No exemplo do ProgramaFor, temos a seguinte condição:

```
i <= 10
```

Vejamos como isso funciona. Ao executar o bloco de código do comando for (lista de comandos entre chaves), essa condição é testada. Se seu valor retornar *true*, o laço será executado mais uma vez para depois testar novamente a condição. Caso contrário, ele irá parar. Isso quer dizer que o laço mostrado continuará a ser executado até o momento no qual a variável *i* tiver valor maior ou igual a 10. Para que essa condição de parada do laço chegue, usualmente você irá usar a seção de incremento do laço for. Última do comando for, essa seção é utilizada para você alterar valores de variáveis, em particular, o valor da variável de controle. No caso do exemplo ProgramaFor, temos o seguinte na seção de incremento:

```
i++;
```

Esse comando é uma instrução válida em Java, em C e em outras linguagens que equivalem à seguinte instrução:

```
i = i + 1;
```

Dessa forma, a instrução i++ incrementa o valor de *i* em uma unidade. É um comando bem enxuto e bastante utilizado dentro da seção de incremento dos laços for. Uma característica adicional do laço for e de sua variável de controle é que se a

variável *i* já existisse, você poderia simplesmente inicializá-la, como no exemplo de código a seguir:

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

É possível fazer esse trabalho de inicialização também antes do laço, como no exemplo a seguir:

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

Note que o código de inicialização está vazio, visto que todo o trabalho de declaração e inicialização da variável de controle foi feito antes do laço. Entretanto, a forma mais recomendada é a de se declarar e de se inicializar a variável de controle dentro da própria instrução for, como mostrado no ProgramaFor e no código a seguir. Isso porque geralmente a variável *i* só é útil para o conteúdo dentro do laço for, sendo então interessante ela ter escopo local ao bloco do for:

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

No caso do exemplo mostrado, o valor da variável *i* é utilizado tanto para controle do laço como para realizar processamento (imprimir seu valor para o usuário). Porém, não é incomum termos variáveis utilizadas apenas para controle do laço, sem ter seu valor usado diretamente na execução de cada laço. Veja isso no exemplo a seguir:

```
public class ProgramaForAsteristico {
   public static void main(String[] args) {
      String texto = "*";
      for (int i = 1; i <= 10; i++) {
         System.out.println(texto);
         texto = texto + "*";
      }
   }
}
```

Esse programa declara uma outra variável, de nome texto, a qual é utilizada no processamento do laço. Você consegue perceber o que ela gera como resultado, ou seja, o que é impresso na tela? Fica como exercício para você fazer isso na sua mente ou no papel inicialmente, e depois implementar e executar o programa para validar sua resposta!

Veja agora um passo a passo do que se deve fazer ao usar o laço for:

1. Cria-se uma variável de controle, que poderá servir como uma espécie de contador, ou seja, para contar o número de vezes que o bloco de comandos será executado.
2. A variável recebe um valor inicial (chamamos a esse processo de "inicialização"). No caso de um contador, geralmente, esse valor inicial é 0 (zero) ou 1 (um).
3. Uma "condição" determinada (por exemplo, se o valor final do contador foi atingido) é testada e, se for verdadeira, o bloco de comandos que pertence ao laço é executado. É através dessa condição que é determinado o número de repetições do laço.
4. A variável de controle é incrementada ou decrementada.
5. O programa então voltará ao passo 3 do laço para testar novamente a condição. Se a condição for falsa (ou seja, o valor de término tenha sido atingido), o laço é interrompido. Caso contrário, o laço continuará com os passos 3 e 4, e assim novamente.

Ao usar o laço for, você deve ter o cuidado de inicializar corretamente as variáveis utilizadas. Veja o exemplo a seguir. Tente observar sem escrever e rodar o programa, só imaginando na sua mente ou papel, quantas vezes o código dentro do laço será executado.

```
public class ProgramaFor2 {
    public static void main(String[] args) {
        String texto = "*";
        for (int i = 10; i < 10; i++) {
            System.out.println(texto);
        texto = texto + "*";
        }
    }
}
```

No laço for, o teste condicional é executado no início da execução, ou seja, se logo no início do programa todas as condições forem falsas, o código dentro do laço não será executado nenhuma vez.

2.1. O Laço Infinito

Quando trabalhamos com laços, uma coisa que pode ocorrer são os laços infinitos. Observe o seguinte código e pense em quantas vezes ele será executado:

```
public class ProgramaForInfinito {
    public static void main(String[] args) {
        for (int i = 1; i < 10; i = i) {
            System.out.println("Olá");
        }
    }
}
```

O laço mostrado irá imprimir a mensagem “Olá” sem parar, pois há um erro na condição de incremento, que está i = i, ao invés de i = i + 1, necessário para incrementar a variável *i*. Em geral, laços infinitos são erros no programa, pois raras são as situações nas quais nós precisamos utilizá-los. De qualquer forma, caso você realmente precise de um laço, use a seguinte estrutura:

```
for ( ; ; ) {
    // comandos que vão rodar infinitamente
}
```

Note que não é especificado nenhum código nas seções do for (inicialização, condição e incremento). E lembre-se: quando executar um programa que entre em laço infinito, utilize um botão geralmente vermelho (ex: ) presente nos ambientes de desenvolvimento (NetBeans, Eclipse, etc.) para parar a execução do programa.

Vídeo 02 - Comando For

Atividade 01

1. Implemente o ProgramaForAsteristico mostrado e indique qual a resposta produzida pelo programa.
2. Implemente o ProgramaFor2 mostrado anteriormente e indique qual a resposta produzida pelo programa. Comente em cima da quantidade de vezes em que o laço dentro do programa é executado.
3. Implemente e execute o ProgramaForInfinito. Relate o que acontece e como você para a execução do programa.
4. Implemente um programa que realiza um laço infinito e que, em cada laço, mostre a quantidade de vezes que ele já foi executado.

3. O Laço while

O próximo comando de iteração que você irá aprender é o laço **while**, que tem a seguinte forma:

```
1 while (condição) comando;
```

Assim como o comando for, o comando while é utilizado para repetir a execução de um bloco de comandos. Eles podem ser utilizados no lugar do comando for, como mostrado no exemplo a seguir:

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

O ProgramaWhile é utilizado para imprimir os números de 1 a 10, exatamente como o ProgramaFor anteriormente mostrado faz. Vejamos como funciona esse comando. Primeiro, a execução do while depende de uma condição, que no caso do exemplo é i <= 10. Essa condição deve ser uma expressão boleana e seu valor é utilizado para determinar se o laço deve (valor true) ou não (valor false) ser executado. Se o valor da expressão for falso, o laço não é executado, mas sim a próxima linha de código do programa.

Apesar do while e do for poderem ser utilizados de forma alternativa, cada um é mais apropriado para determinadas situações. Por exemplo, qual implementação você achou mais interessante, a do ProgramaFor (mostrado novamente abaixo) ou do ProgramaWhile?

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

Em geral, as pessoas preferem a implementação do ProgramaFor. Isso porque laços implementados com o comando for são interessantes quando existe uma contagem. Em geral, sabe-se quantas vezes o laço será executado. Já o while é bastante interessante quando não se sabe inicialmente quantas vezes o laço será executado. Imagine a situação onde deva-se sortear um número entre 1 e 10 e perguntar qual é esse número para o usuário até que ele acerte. Isso é feito de forma relativamente natural usando-se o comando while:

```
import java.util.Scanner;
public class ProgramaWhileSorteio {
    public static void main(String[] args) {
        long numero = Math.round(Math.random() * 10);
        long chute;
        Scanner leitor = new Scanner(System.in);
        System.out.println("Digite um número entre 1 e 10");
        chute = leitor.nextLong();
        while (numero != chute) {
            System.out.println("Digite um número entre 1 e 10");
            chute = leitor.nextLong();
        }
        System.out.println("Você acertou!");
    }
}
```

Observe no código que o sorteio é realizado na seguinte linha de código:

```
long numero = Math.round(Math.random() * 10);
```

A instrução Math.random() retorna um valor numérico aleatório de precisão decimal (double) entre 0 e 1. Quando multiplicamos por 10, esse valor passa a ser entre 0 e 10. Esse novo valor é então passado para a função Math.round(), a qual arredonda para o número inteiro mais próximo.

Em seguida, é solicitado ao usuário que digite um número entre 1 e 10, usando-se o leitor (Scanner) já discutido anteriormente. O laço então compara o valor lido do usuário com o valor sorteado. Enquanto esses valores não forem iguais (numero != chute), o laço será executado solicitando novamente um novo número. Quando o usuário acertar, o laço irá terminar e a mensagem “Você acertou!” será impressa na tela.

Se você tentar implementar esse programa usando um laço for, o seu uso não será nada natural.

Vídeo 03 - Comando While

Atividade 02

1. Implemente e execute o ProgramaFor e o ProgramaWhileSorteio.
2. Implemente um programa chamado ProgramaForSorteio que é a implementação do ProgramaWhileSorteio usando-se for ao invés de while.

4. O Laço do-while

Não sei se você notou, mas, no ProgramaWhileSorteio, existe uma repetição do seguinte trecho de código:

```
System.out.println("Digite um número entre 1 e 10");
chute = leitor.nextLong();
```

Essa duplicação deve-se ao fato de que é preciso ter um valor inicial para ser testado na primeira vez em que o laço é executado. Para evitar essa duplicação, você poderia fazer algo como:

```
import java.util.Scanner;
public class ProgramaWhileSorteio2 {
    public static void main(String[] args) {
        long numero = Math.round(Math.random() * 10);
        Scanner leitor = new Scanner(System.in);
        long chute = -1;
        while (numero != chute) {
            System.out.println("Digite um número entre 1 e 10");
            chute = leitor.nextLong();
        }
        System.out.println("Você acertou!");
    }
}
```

Note que agora o valor -1, que é um valor que nunca será sorteado, está sendo usado como valor inicial da variável. Isso garante que o laço seja executado pelo menos uma vez. Nessas situações, em que sabemos que o bloco de código do laço precisa ser executado pelo menos uma vez, nós fazemos uso do comando **do-while**. Veja agora esse mesmo programa com o do-while:

```
import java.util.Scanner;
public class ProgramaDoWhileSorteio {
    public static void main(String[] args) {
        long numero = Math.round(Math.random() * 10);
        long chute;
        Scanner leitor = new Scanner(System.in);
        do {
            System.out.println("Digite um número entre 1 e 10");
            chute = leitor.nextLong();
        } while (numero != chute);
        System.out.println("Você acertou!");
    }
}
```

Note que no do-while a condição vai para o final da estrutura. O funcionamento desse código é o seguinte: o bloco do código do laço é executado a primeira vez e depois é verificada a condição, que no caso do exemplo é numero != chute. Se essa condição retornar *true*, o laço será executado novamente. Caso a expressão seja falsa, o laço termina. Perceba que ProgramaDoWhileSorteio é uma implementação bem mais natural que a do ProgramaWhileSorteio ou do ProgramaWhileSorteio2, apesar de todas essas implementações serem equivalentes.

No laço do-while, o código dentro do laço sempre será executado pelo menos uma vez.

Um uso interessante da estrutura **do-while** é na validação de valores lidos do usuário. Considere o caso de querer que o usuário digite um número inteiro entre 1 e 10. Você pode escrever o seguinte bloco de código para garantir que o programa só dará continuidade após o usuário digitar um número que atende a restrição:

```
do {
    System.out.println("Digite um número entre 1 e 10");
    valor = leitor.nextLong();
} while (valor < 1 || valor > 10);
```

Note que o laço acima só irá terminar quando o usuário digitar um número entre 1 e 10. Qualquer número menor do que 1 ou maior do que 10 irá fazer com que o laço seja executado novamente, ou seja, que o número seja solicitado novamente ao usuário.

Vídeo 04 - Comando do-while

Atividade 03

1. Implemente e execute os programas ProgramaWhileSorteio2 e ProgramaDoWhileSorteio.
2. Crie um programa que leia um número e que imprima o seu valor ao quadrado. Caso o quadrado desse número seja igual a 0 (zero), o programa deve sair.

Conclusão

Bem, chegamos ao final desta aula. Lembre que o comando de laço for é o mais adequado quando o programador quer determinar previamente o número de vezes que o trecho será executado. Já o comando while,executa uma iteração enquanto uma determinada condição for verdadeira, testando essa condição no início de sua execução.

Por fim, o comando do-while é semelhante ao while, porém testa a condição no final de sua execução, garantindo, dessa forma, pelo menos uma execução do bloco do código do laço.

Resumo

Nesta aula, você aprendeu a utilizar os comandos de iteração: *for*, *while* e *do-while*. Você viu que o comando for irá realizar um laço de repetição por quantas vezes for determinado ou até que certa condição seja encontrada. Além disso, você aprendeu a utilizar o comando *while*, que é um comando de iteração, onde o programa irá repetir o laço até a condição ser verdadeira. Sendo falsa, encerra o laço. E, por fim, você estudou a variação *do-while* que, ao contrário dos anteriores, testa a condição apenas ao final do laço, executando o laço pelo menos uma vez.

Autoavaliação

1. Escreva um programa que leia 10 números e que escreva na tela apenas os que forem ímpares.
2. Escreva um programa que solicite um número entre 1 e 4. Caso o número digitado seja diferente, mostre a mensagem "Número inválido". Caso contrário, escreva o número na tela.
3. Escreva um programa que gere a saída : 0, 2, 4, 6, 8, 10, 12, 14.

Referências

ASCENCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi de. **Fundamentos da programação de computadores**: algoritmos, Pascal, C/C++ e Java. São Paulo: Editora Pearson, 2008.

THE JAVA tutorials. Disponível em: <http://download.oracle.com/javase/tutorial/>. Acesso em: 6 dez. 2011.