Lógica de Programação

Unidade 15 – Laços de Repetição (for)

QI ESCOLAS E FACULDADES
Curso Técnico em Informática

SUMÁRIO

INTRODUÇÃO

Nesta unidade iremos trabalhar com mais um laço de repetição, o laço **for**.

INSTRUÇÃO FOR

A instrução ***for*** é um laço de repetição assim como o *while* e *do while*, porém ao contrário deles ele possui uma estrutura um pouco diferente. Também executa suas instruções enquanto a condição estabelecida for verdadeira.

Sintaxe da estrutura for

```
for(comando inicial; condição; comando de loop){
      <instruções>;
}
```

Vamos analisar a sintaxe da estrutura for. Observe que na primeira linha, dentro dos parênteses, ao invés de termos apenas a condição como acontecia com o *while* e *do while*, temos agora três comandos separados por ponto e vírgula.

Comando inicial

Este comando executa apenas uma vez, quando o código entra no *loop*. Aqui podemos declarar uma variável, por exemplo, como é muito comum.

Condição

O mesmo tipo de condição que montamos em *while* e *do while*. Se ela for verdadeira, as instruções serão executadas. Se for falsa, o *loop* encerra.

Comando de loop

Este comando é executado sempre que o *for* completa uma volta, ou seja, na primeira execução ele pula este comando.

Exemplo da utilização do for

Vamos desenvolver um programa que mostre a tabuada do 1 ao 10 de um número digitado pelo usuário. Para isso, trabalharemos com a classe Tabuada:

Tabuada
-valor:int
+get/set +gerarTabuada(multiplicador:int):String

Código da classe Tabuada

```
public class Tabuada{
    private int valor;

    public int getValor(){
        return this.valor;
    }
    public void setValor(int valor){
        this.valor = valor;
    }
    public String gerarTabuada(int multiplicador){
        return this.valor + " x " + multiplicador + " = " + this.valor*multiplicador;
    }
}
```

Observe que o método *gerarTabuada* recebe o multiplicador por argumento e retorna o cálculo no formato: "numero x multiplicador = resultado".

Imagine por um momento que não existam comandos de *loop*. Se tivéssemos que fazer um programa que mostra a tabuada de um número do 1 ao 10, teríamos que chamar o método *gerarTabuada* 10 vezes, cada vez utilizando um multiplicador diferente. Observe o exemplo:

```
import java.util.Scanner;
public class Main1{
    public static void main(String args[]){
        Scanner ler = new Scanner(System.in);

        Tabuada t1 = new Tabuada();
        System.out.print("Informe um número inteiro para ver sua tabuada:");
        t1.setValor(ler.nextInt());
        int multiplicador = 1;
        //mostrando a tabuada de um número
        System.out.println(t1.gerarTabuada(multiplicador));
        multiplicador++;
        System.out.println(t1.gerarTabuada(multiplicador));
        multiplicador++;
        System.out.println(t1.gerarTabuada(multiplicador));
        multiplicador++;
        System.out.println(t1.gerarTabuada(multiplicador));
        multiplicador++;
        System.out.println(t1.gerarTabuada(multiplicador));
        multiplicador++;
        System.out.println(t1.gerarTabuada(multiplicador));
        multiplicador++;
        System.out.println(t1.gerarTabuada(multiplicador));
        multiplicador++;
        System.out.println(t1.gerarTabuada(multiplicador));
        multiplicador++;
        System.out.println(t1.gerarTabuada(multiplicador));
        multiplicador++;
        System.out.println(t1.gerarTabuada(multiplicador));
        multiplicador++;
    }
}
```

Podemos notar que os comandos abaixo são repetidos 10 vezes:

```
System.out.println(t1.gerarTabuada(multiplicador));
multiplicador++;
```

Para não precisarmos escrever os mesmos comandos tantas vezes, podemos colocá-los dentro de uma estrutura de repetição, como o *for*, por exemplo, e fazer com que o *loop* execute o comando quantas vezes forem necessárias. Observe como ficou a classe *Main* utilizando o *for*:

```
1  import java.util.Scanner;
2  public class Main2{
3      public static void main(String args[]){
4          Scanner ler = new Scanner(System.in);
5
6          Tabuada t1 = new Tabuada();
7          System.out.print("Informe um número inteiro para ver sua tabuada:");
8          t1.setValor(ler.nextInt());
9
10         //mostrando a tabuada de um número
11         for(int multiplicador=1; multiplicador<=10; multiplicador++){
12             System.out.println(t1.gerarTabuada(multiplicador));
13         }
14     }
15 }
```

Note que o código ficou bem mais curto! Vamos analisar como o *for* funciona:

Após o usuário digitar o valor que deseja, o código passa para o *for*. Na linha 11, ele encontra três comandos:

```
11.     for(int multiplicador=1; multiplicador<=10; multiplicador++){
```

O for nunca executa os três comandos na mesma execução!

Vamos imaginar que o usuário tenha digitado 5. Primeiramente ele cria a variável "multiplicador" e atribui a ela o valor 1, e em seguida testa a condição: multiplicador<=10 → 1<=10 → verdadeiro. Assim, ele ignora o último comando (multiplicador++) e executa o comando interno "System.out.println(t1.gerarTabuada(multiplicador))".

t1[valor = 5**]**	**multiplicador**
	1

No terminal: **5 x 1 = 5**

Ao chegar na linha 13 (chaves) ele retorna à linha 11(for) e primeiramente executa o último comando (multiplicador++), alterando o valor da variável:

t1[valor = 5**]**	**multiplicador**
	~~1~~
	2

Após aumentar o valor do multiplicador, ele testa novamente a condição (multiplicador<=10 → 2<=10 → verdadeiro) e executa novamente o comando interno, exibindo no terminal: **5 x 2 = 10.**

Ao chegar novamente na linha 13, das chaves, ele retorna à linha 11(for), executa o comando "multiplicador++" e testa novamente a condição. Enquanto a condição for verdadeira, ele permanece no *loop*.

O resultado final no terminal, caso o usuário informasse 5, seria:

```
Informe um número inteiro para ver sua tabuada:5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```