Laço de Repetição

As estruturas de repetição são loops que utilizamos para executar tarefas repetidas em um bloco de código até que sua determinada condição sea satisfeita.

Por exemplo: imagine que você esteja desenvolvendo um programa que coletará o voto dos alunos para o cargo de presidente do grêmio. Cerca de 200 alunos votarão, com isso entendemos que 200 alunos farão a mesma ação. Em vez de criar um sistema, com 200 instruções para cada aluno, criamos uma instrução de repetição, e agregamos o voto a variável destinada ao candidato.

Os existem diferentes loops que podemos fazer usando For, While ou Do/While, e com eles conseguimos realizar diversas tarefas, como fazer cálculos.

1. For

A For é uma estrutura de repetição em que seu ciclo será executado por um tempo ou condição pré-determinada. Esta estrutura deve receber 3 parâmetros:

* 1. A expressão de inicialização;
  2. A condição de teste;
  3. O contador.

Veja a sintaxe da estrutura for abaixo:

for(valor inicial; condição de teste; contador){

//Bloco de código

}

Vamos entender o que os parâmetros significam:

Valor inicial: este é um campo reservado para variáveis do tipo inteiro. O valor inicial deve ser um número para iniciar a contagem que, por sua vez, pode incrementar ou decrementar.

Condição de teste (limite): esta condição é escrita com operadores relacionais que definem os valores a serem atingidos pela variável que foi iniciada. A condição de teste realizará o teste enquanto as condições forem verdadeiras.

Contador: aqui indicamos como a contagem será realizada, indicando ao programa se o valor será acrescido (++) ou decrementado (--).

Agora que sabemos como é a estrutura for, vamos praticar um pouco, mas antes, veja o fluxograma abaixo para entender o que será realizado.

NÃO

SIM

FIM

INCREMENTA OU DECREMENTA A VARIÁVEL

BLOCO DE CÓDIGO EXECUTADO

CONDIÇÃO

O CONTADOR É INICIADO

INÍCIO

Ao iniciar o programa, a variável contadora é iniciada com seu valor pré-definido. Em seguida o programa testará se a condição seja verdadeira ou falsa; enquanto for verdadeira, o bloco de código será executado, mas se for falsa, o programa será encerrado.

Agora vamos passar o que foi definido no fluxograma para o “papel”. Veja abaixo um exemplo:

public static void main(String[] args) {

// Conatndo com Incremento

for(int a = 0; a <= 10; ++a) {

System.out.println("A variável A contou: " + a);

}

// Comntando com Decremento

for(int b = 10; b >= 0; --b) {

System.out.println("A variável B contou: " + b);

}

}

É possível combinar o que aprendemos em aulas passadas. No fluxograma abaixo veremos como podemos definir o que o programa fará para coletar diferentes dados:

INICIO

CONTADOR = 1

EXIBIR NOME

A < 4 ?

LEIA O NOME

DIGITE UM NOME:

SIM

CONTADOR ++

NÃO

FIM

Agora veja no exemplo abaixo como podemos usar um Scanner para coletar dados via console:

public static void main(String[] args) {

String nome;

int a;

Scanner leia = new Scanner(System.in);

for (a = 1; a < 4; a++) {

System.out.println("\nDigite o " + a + "º nome: ");

nome = leia.nextLine();

System.out.println("O " + a + "º nome é: " + nome);

}

}

No código acima, utilizamos o método Scanner para coletar o nome dos usuários. Quando o loop se repete, o programa solicitará um novo valor para a variável nome, que permanecerá com este valor até que ele seja sobrescrito novamente.

A variável a terá dois papeis:

Primeiro: servirá como contador, onde o programa comparará o valor da variável com o valor da condição.

Segundo: servirá como ranking, onde ele atribui a posição do nome de acordo com sua ordem de escrita. Ele pega o número da variável a cada rodada que o looping realiza.

2.0 While

A estrutura while é similar a estrutura for, com algumas diferenças. Quando utilizamos a estrutura for, nós sabemos a quantidade de vezes que o loop se repetirá. Nós utilizamos o while quando não sabemos a quantidade de vezes que o código será repetido.

Veja abaixo a sintaxe dessa estrutura:

while (condição){

//Bloco de código

}

A estrutura while permite que o código seja executado com base em uma determinada condição. Lembre-se que while recebe como parâmetro apenas a condição, e não um conjunto de parâmetros, conforme observamos anteriormente na estrutura for.

Vamos entender melhor com um exemplo. Abaixo estará um código, onde trabalharemos com a estrutura while até que a condição seja verdadeira:

package Repeticao;

import java.util.Scanner;

public class estruturaWhile {

public static void main(String[] args) {

Scanner leia = new Scanner(System.in);

int numero = -1;

while (numero != 10) {

System.out.println("\n\nDigite um número: ");

numero = leia.nextInt();

if (numero == 10) {

System.out.println("\n\nVocê Acertou!!!"

+ "\nO número é " + numero);

}

else {

System.out.println("\n\nVocê errou..."

+ "\nO número não é " + numero);

}

}

}

Foi criado um Scanner com o nome leia para coletar do usuário um número; e em seguida, iniciamos um while e atribuímos o valor do Scanner leia na variável numero.

No while, enquanto a variável numero não for igual a condição (10), ele repetirá o processo de coleta dos dados. Caso o valor seja semelhante ao valor da condição (10), o loop se encerrará.

Trabalhamos com if/else para decidir qual mensagem será exibida caso o usuário acerte o número ou erre em sua tentativa. Na mensagem exibida no console, foi concatenada a ela o valor a variável numero, onde o usuário pode verificar o número que ele escolheu. Simples né!

Vamos fazer outro exemplo para praticar:

package Repeticao;

import java.util.Scanner;

public class estruturaWhile {

public static void main(String[] args) {

int numero, resultado, contador = 1;

Scanner leia = new Scanner(System.in);

while(contador < 4) {

System.out.println(

"\n\nDigite o " + contador + "º número: "

);

numero = leia.nextInt();

resultado = numero \* 3;

System.out.println(numero + " x 3 = " + resultado);

System.out.println(

"-----------------------------------------");

contador++;

}

System.out.println("\t\t O Programa foi encerrado");

}

}

Foram criadas três variáveis do tipo inteiro:

contador = 1: para agregar valor após o loop e impedir que o loop funcione sem um determinado número de rodadas.

numero: armazena o número selecionado pelo usuário, atribuído no método Scanner leia.

resultado: armazenará o resultado da operação entre numero x 3.

Foi criado um Scanner leia para coletar os dados do usuário. Depois iniciamos um while com a seguinte condição: contador < 4. Enquanto o contador for menor do que 4, o loop funcionará, coletando o número e multiplicando por 3.

Ao final de cada rodada do loop, há um contador++ que agrega valor ao contador no final de cada loop. No sim, quando o contador sai da condição < 4, o while é encerrado e a mensagem indicando o fim do programa é exibida.

3.0 DO/WHILE

O DO/WHILE é uma variação do while, onde o loop executará o bloco de código uma vez antes de verificar se a condição é verdadeira, e depois repetirá o loop enquanto a condição for verdadeira.

Veja a sintaxe abaixo:

do {

//Bloco de código a ser executado

} while(condição);

Para aprendermos melhor, vamos fazer um exemplo abaixo:

int i = 0;

do {

System.out.println(i);

i++;

}while(i < 5);

No exemplo que usamos, o loop será executado pelo menos uma vez, mesmo que a condição seja falsa, porque o bloco de código é executado antes da condição ser testada.

Por exemplo, quando mudamos o código while para “while(i < -1);”, o programa executará apenas um valor e não continuará o loop, porque a condição não é mais verdadeira.

Veremos o último exemplo, para fixarmos em nossa mente a estrutura DO/WHILE em funcionamento:

int contagem = 10;

System.out.println( "########################"

+ "# Contagem regressiva! #"

+ "########################");

do {

System.out.println(contagem);

contagem--;

}while (contagem > 0);

O código acima fará a contagem regressiva, decrementando um número superior para o valor de 0. Lembre-se que, se o contador estiver configurado igual “contagem++", o valor d variável nunca será o mesmo valor da condição imposta no while.

Conclusão

Na aula de hoje, observamos como fazer o Laço de Repetição não é complicado, mas exige-se um pouco de lógica do desenvolvedor para realizar ações simples.

Aprendemos também sobre a estrutura for e while, e como podemos utilizar if/else entro dessas estruturas. Também observamos como a estrutura do/while pode executar tarefas enquanto a condição imposta se manter verdadeira.

Espero que tenha gostado desta aula, apenas não se esqueça de fazer os exercícios para fixar na mente o que você aprendeu hoje.

Referências

Laços de Repetição.

https://github.com/Leon4rdoalves/CookBook-Java/blob/main/07.md

Estruturas condicionais e estrutura de repetição em Java.

https://www.treinaweb.com.br/blog/estruturas-condicionais-e-estruturas-de-repeticao-em-java#google\_vignette

O ciclo do/while.

https://www.w3schools.com/java/java\_while\_loop\_do.asp