Java Developer

Java Core

Sumário

Objetivos deste Módulo

Ao final deste módulo, objetiva-se que o aluno adquira:

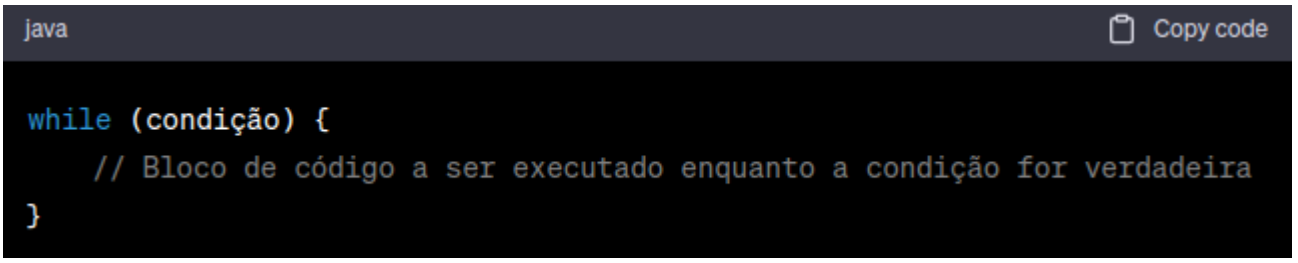
Conhecimentos:

- Estrutura de repetição while, do-while
- Tipos de incremento e decremento
- Geração de números randômicos

1. While

O loop **while** é uma estrutura de controle que permite executar um bloco de código repetidamente enquanto uma condição especificada for verdadeira. Ele verifica a condição antes de cada iteração do loop. Se a condição for verdadeira, o bloco de código dentro do loop é executado. Caso contrário, se a condição for falsa desde o início, o bloco de código dentro do loop não será executado nenhuma vez.

Aqui está a sintaxe básica do loop **while** em Java:



Aqui está um exemplo simples de como você pode usar o loop while em Java para imprimir os números de 1 a 5:

```
public class ExemploWhile {

    public static void main(String[] args) {

        int contador = 1;

        while (contador <= 5) {
            System.out.println(contador);
            contador++;
        }
    }

}
```

Neste exemplo, o loop **while** continua executando enquanto o valor da variável contador for menor ou igual a 5. A cada iteração, o número é impresso e o contador é incrementado em 1. O loop para quando o contador atinge o valor 6.

Lembre-se de que é importante garantir que a condição do loop **while** seja eventualmente falsa, caso contrário, você terá um loop infinito, o que travará o programa. Certifique-se de que algo no bloco de código dentro do loop esteja alterando a condição para que ela eventualmente se torne falsa.

Em resumo, o loop **while** é uma construção útil para executar repetidamente um bloco de código enquanto uma condição específica for verdadeira em Java.

Aqui estão algumas coisas a ter em mente ao usar o loop **while**:

Inicialização da Variável de Controle: Antes de entrar no loop while, é importante inicializar a variável de controle que será usada na condição do loop. Isso garante que o loop tenha um ponto de partida.

Atualização da Variável de Controle: Dentro do bloco de código do loop, certifique-se de atualizar a

variável de controle para que a condição eventualmente se torne falsa. Caso contrário, você terá um loop infinito.

Condição do Loop: A condição do loop deve ser uma expressão que pode ser avaliada como verdadeira ou falsa. A expressão geralmente envolve a variável de controle do loop.

Cuidado com Loops Infinitos: Se a condição do loop nunca se tornar falsa, você terá um loop infinito, o que pode travar seu programa. Certifique-se de que há uma maneira de sair do loop.

Controle de Fluxo: Lembre-se de que o fluxo do programa é controlado pela condição do loop. Se a condição for falsa desde o início, o bloco de código dentro do loop não será executado.

Pensando em Problemas: Ao usar um loop while, pense em como você pode criar uma condição que representa o término natural das repetições. Isso ajudará você a projetar loops eficientes e evitar loops infinitos.

Aqui está outro exemplo para ilustrar o uso do loop while. Vamos escrever um programa que encontra a soma de todos os números inteiros de 1 a 100:

```
public class SomaNumeros {

    public static void main(String[] args) {

        int numero = 1;
        int soma = 0;

        while (numero <= 100) {
            soma += numero;
            numero++;
        }

        System.out.println("A soma dos números de 1 a 100 é: " + soma);
    }

}
```

Neste exemplo, a variável numero é usada como a variável de controle do loop. A cada iteração, o valor de numero é adicionado à variável soma e depois incrementado. O loop continua até que numero seja menor ou igual a 100.

2. Do While

O loop **do-while** é outra construção de repetição em Java que permite executar um bloco de código repetidamente enquanto uma condição especificada for verdadeira. A principal **diferença** entre o **do-while** e o **while convencional** é que no **do-while, o bloco de código é executado pelo menos uma vez**, independentemente da condição ser verdadeira ou falsa no início. Depois que o bloco de código é executado, a condição é verificada. Se a condição for verdadeira, o loop continuará executando, caso contrário, ele sairá.

Aqui está a sintaxe básica do loop do-while em Java:

```java
do {
    // Bloco de código a ser executado
} while (condição);
```

Aqui está um exemplo de um loop **do-while** que solicita ao usuário um número até que um número positivo seja inserido:

```
import java.util.Scanner;

public class NumeroPositivo {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        int numero;

        do {
            System.out.print("Digite um número positivo: ");
            numero = scanner.nextInt();
        } while (numero <= 0);

        System.out.println("Você digitou um número positivo: " + numero);

    }
}
```

Vamos ver um exemplo simples de como usar o loop **do-while** para imprimir os números de 1 a 5:

```
public class ExemploDoWhile {
    public static void main(String[] args) {

        int contador = 1;

        do {
            System.out.println(contador);
            contador++;
        } while (contador <= 5);
    }

}
```

Neste exemplo, mesmo que a condição **contador <= 5** seja falsa no início, o bloco de código dentro do do é executado pelo menos uma vez, imprimindo o número 1. Depois disso, a condição é verificada. Enquanto a condição ainda for verdadeira, o bloco de código continuará a ser executado até que o contador alcance o valor 6.

Outro exemplo:

```
import java.util.Scanner;

public class MenuExemplo {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        String escolha;

        do {
            System.out.println("1. Opção 1");
            System.out.println("2. Opção 2");
            System.out.println("3. Sair");
            System.out.print("Escolha uma opção: ");

            escolha = scanner.next();

            switch (escolha) {
                case "1":
                    System.out.println("Você escolheu a Opção 1.");
                    break;
                case "2":
                    System.out.println("Você escolheu a Opção 2.");
                    break;
                case "3":
                    System.out.println("Saindo do programa.");
                    break;
                default:
                    System.out.println("Escolha inválida. Tente novamente.");
            }
        } while (!escolha.equals("3"));

        scanner.close();

    }
}
```

Neste exemplo, o programa exibe um menu e pede ao usuário que escolha uma opção. O do-while garante que o menu seja exibido pelo menos uma vez, e o programa continua a mostrar o menu e a responder às escolhas do usuário até que o usuário escolha a opção "3" para sair.

Lembre-se de Fechar Recursos: Se você estiver usando recursos como Scanner, certifique-se de fechá-los quando não precisar mais deles, como demonstrado no exemplo acima (usando scanner.close()).

O uso adequado do loop **do-while** é quando você deseja que o bloco de código seja executado **pelo menos uma vez**, independentemente da condição. Por exemplo, ao solicitar entrada do usuário, você pode usar um loop do-while para garantir que o usuário forneça algum tipo de entrada antes de prosseguir.

Tenha em mente que, assim como outros loops, é importante garantir que algo dentro do bloco de código eventualmente altere a condição para que ela se torne falsa, evitando assim loops infinitos.

Em resumo, o loop do-while é uma estrutura de repetição que garante que o bloco de código seja executado pelo menos uma vez antes de verificar a condição. Isso pode ser útil em muitas situações em que você precisa de uma execução inicial garantida.

3. Tipos de incremento e decremento

Pré-incremento é uma operação que aumenta o valor de uma variável numérica (como um inteiro) em 1 antes de qualquer outra operação ser realizada na variável. Ele é representado pelo operador "++" colocado antes da variável. Aqui está um exemplo simples:

```
int numero = 5;
int resultado = ++numero;
```

Neste exemplo, o pré-incremento é usado na linha int resultado = ++numero;. O que acontece é o seguinte:

O valor da variável numero é aumentado em 1 antes de ser atribuído à variável resultado.

Portanto, após essa linha de código ser executada, numero será igual a 6 e resultado também será igual a 6.

O pré-incremento é uma operação muito útil quando você deseja aumentar o valor de uma variável antes de usá-la em uma expressão. Lembre-se de que ele difere do **pós-incremento** (que usa o operador "++" após a variável, como numero++), que primeiro usa o valor atual da variável em uma expressão e, em seguida, a aumenta em 1.

Pré-decremento é uma operação que diminui o valor de uma variável numérica (como um inteiro) em 1 antes de qualquer outra operação ser realizada na variável. Ele é representado pelo operador "--" colocado antes da variável. Aqui está um exemplo simples:

```
int numero = 5;
int resultado = --numero;
```

Neste exemplo, o pré-decremento é usado na linha int resultado = --numero;. O que acontece é o seguinte:

O valor da variável numero é reduzido em 1 antes de ser atribuído à variável resultado.

Portanto, após essa linha de código ser executada, numero será igual a 4 e resultado também será igual a 4.
O pré-decremento é útil quando você deseja diminuir o valor de uma variável antes de usá-la em uma expressão. Assim como o pré-incremento, ele difere do **pós-decremento** (que usa o operador "--" após a variável, como numero--), que primeiro usa o valor atual da variável em uma expressão e, em seguida, a diminui em 1.

Aqui está um exemplo de uso em um loop:

```
int contador = 5;
while (contador > 0) {
   System.out.println(contador);
   --contador; // Pré-decremento para diminuir o contador em 1
}
```

Qual a diferença de pré e pós incremento em java?

A diferença entre o **pré-incremento** e o **pós-incremento** em Java, ou seja, os operadores ++variável (pré-incremento) e variável++ (pós-incremento), está na ordem em que a variável é incrementada e em como o valor é avaliado em expressões. Aqui está a diferença entre eles:

Pré-incremento (++variável):

- Aumenta o valor da variável antes de qualquer outra operação ser realizada na variável.
- Retorna o valor incrementado.
- O valor atual da variável é usado na expressão em que o pré-incremento é aplicado.

Exemplo:

```
int a = 5;
int b = ++a;
// Agora, 'a' é igual a 6 e 'b' é igual a 6
```

Pós-incremento (variável++):

- Usa o valor atual da variável na expressão em que o pós-incremento é aplicado.
- Após a avaliação da expressão, a variável é incrementada em 1.
- Retorna o valor original (não incrementado).

Exemplo:

```
int x = 5;
int y = x++;
// Agora, 'x' é igual a 6, mas 'y' é igual a 5
```

4. Gerar números randômicos

Em Java, você pode gerar números aleatórios utilizando a classe java.util.**Random** ou a classe java.lang.**Math**. Aqui estão dois métodos populares para gerar números aleatórios:

```
import java.util.Random;

public class GerarNumerosRandomicos {
    public static void main(String[] args) {
        // Crie uma instância da classe Random
        Random random = new Random();

        // Gere um número inteiro aleatório entre 0 (inclusive) e 100 (exclusivo)
        int numeroInteiro = random.nextInt(100);

        // Gere um número decimal aleatório entre 0 (inclusive) e 1 (exclusivo)
        double numeroDecimal = random.nextDouble();

        System.out.println("Número inteiro aleatório: " + numeroInteiro);
        System.out.println("Número decimal aleatório: " + numeroDecimal);
    }
}
```

==========

```
public class GerarNumerosRandomicos {
    public static void main(String[] args) {

        // Gere um número decimal aleatório entre 0 (inclusive) e 1 (exclusivo)
        double numeroDecimal = Math.random();

        System.out.println("Número decimal aleatório: " + numeroDecimal);

    }
}
```

5. Exercícios

1) Crie um programa que preencha um array de inteiros com os números de 0 a 10 (exclusivo) e, em seguida, exiba esses números em ordem inversa usando um loop while.

2) Faça um programa que verifique se um número específico está presente em um array de inteiros usando um loop while.

3) Faça o exercício 2 com loop do-while

4) Crie um programa que multiplique todos os elementos de um array de números inteiros por 2 usando um loop while. Atualizar o novo valor no próprio array.

5) Faça um programa que leia números do usuário e calcule a média desses números. Pare a leitura quando o usuário inserir o número 0 e exiba a média usando um loop do-while.

6) Crie um programa que simule uma conta bancária. Peça ao usuário para inserir o saldo inicial e, em seguida, permita que o usuário faça depósitos e saques até que ele decida sair do programa. Usar while ou do-while

7) Crie um programa que peça ao usuário para adivinhar um número entre 1 e 100. Continue pedindo ao usuário para adivinhar até que ele acerte o número usando um loop while. Dê dicas se o número adivinhado estiver muito alto ou muito baixo. Não esqueça de gerar o número randômico previamente.