Estrutura de controle

Comando condicional if-else.

O comando condicional `if-else` em Java é usado para tomar decisões em um programa com base em uma condição específica. Ele permite que o programa execute diferentes blocos de código com base no resultado de uma expressão booleana.

A sintaxe básica do comando `if-else` em Java é a seguinte:

```
if (condição) {
    // código a ser executado se a condição for verdadeira
} else {
    // código a ser executado se a condição for falsa
}
```

A condição é uma expressão booleana que retorna `true` ou `false`. Se a condição for verdadeira, o bloco de código dentro do `if` é executado. Caso contrário, o bloco de código dentro do `else` é executado.

Aqui estão alguns exemplos de como usar o comando `if-else` em Java:

```
int idade = 18;

if (idade >= 18) {
    System.out.println("Pode votar.");
} else {
    System.out.println("Não pode votar.");
}
```

Nesse exemplo, o programa verifica se a variável `idade` é maior ou igual a 18. Se a condição for verdadeira, o programa imprime "Pode votar". Caso contrário, o programa imprime "Não pode votar".

```
int numero = 5;

if (numero % 2 == 0) {
    System.out.println("O número é par.");
} else {
    System.out.println("O número é ímpar.");
}
```

Nesse exemplo, o programa verifica se a variável `numero` é divisível por 2. Se a condição for verdadeira, o programa imprime "O número é par". Caso contrário, o programa imprime "O número é ímpar".

Comando switch-case.

O comando switch-case em Java é usado para executar diferentes blocos de código com base no valor de uma variável específica. É uma alternativa ao uso de vários comandos if-else aninhados.

Aqui está a sintaxe básica do comando switch-case em Java:

```
switch (variável) {
    case valor1:
        // código a ser executado se a variável for igual a valor1
        break;
    case valor2:
        // código a ser executado se a variável for igual a valor2
        break;
    // ...
    default:
        // código a ser executado se a variável não for igual a nenhum dos
valores
}
```

A variável é a variável que o programa está verificando, enquanto cada caso é um valor específico que pode ser atribuído à variável. Se a variável for igual a um valor específico, o bloco de código dentro desse caso será executado. O comando `break` é usado para interromper a execução do bloco de código atual e sair do comando switch-case.

Aqui está um exemplo simples de como usar o comando switch-case em Java:

```
int diaDaSemana = 3;

switch (diaDaSemana) {
    case 1:
        System.out.println("Domingo");
        break;
    case 2:
        System.out.println("Segunda-feira");
        break;
    case 3:
        System.out.println("Terça-feira");
        break;
    case 4:
        System.out.println("Quarta-feira");
        break;
    case 5:
        System.out.println("Quinta-feira");
        break;
    case 6:
        System.out.println("Sexta-feira");
        break;
    case 7:
```

```
            System.out.println("Sábado");
            break;
        default:
            System.out.println("Dia da semana inválido");
    }
```

Nesse exemplo, o programa verifica o valor da variável `diaDaSemana` e executa o bloco de código correspondente ao caso que corresponde ao valor. Como `diaDaSemana` é igual a 3, o programa imprime "Terça-feira". Se o valor de `diaDaSemana` não corresponder a nenhum dos casos, o programa executará o bloco de código dentro do `default`.

Laços de repetição for, while e do-while.

for:

O laço de repetição for em Java é usado para executar um bloco de código repetidamente por um número específico de vezes. A sintaxe básica é a seguinte:

```
for (inicialização; condição; atualização) {
    // código a ser executado repetidamente
}
```

Onde:

- `inicialização`: define a variável de controle do laço e atribui um valor inicial a ela.
- `condição`: é uma expressão booleana que determina se o bloco de código deve ser executado novamente. O laço será executado enquanto a condição for verdadeira.
- `atualização`: é uma instrução que atualiza o valor da variável de controle após cada execução do bloco de código.

Aqui está um exemplo simples de como usar o laço de repetição for em Java:

```
for (int i = 0; i < 5; i++) {
    System.out.println("O valor de i é: " + i);
}
```

Nesse exemplo, o laço de repetição for é usado para imprimir os valores de `i` de 0 a 4. A variável de controle `i` é inicializada com 0, a condição é `i < 5` e a atualização é `i++`. O bloco de código dentro do laço é executado cinco vezes, uma vez para cada valor de `i`. A saída desse exemplo seria:

```
O valor de i é: 0
O valor de i é: 1
O valor de i é: 2
O valor de i é: 3
O valor de i é: 4
```

O laço de repetição for é útil quando você sabe quantas vezes o bloco de código deve ser executado.

while:

O laço de repetição while em Java é usado para executar um bloco de código enquanto uma condição específica for verdadeira. A sintaxe básica é a seguinte:

```
while (condição) {
    // código a ser executado repetidamente
}
```

Onde `condição` é uma expressão booleana que determina se o bloco de código dentro do laço deve ser executado.

Aqui está um exemplo simples de como usar o laço de repetição while em Java:

```
int i = 0;
while (i < 5) {
    System.out.println("O valor de i é: " + i);
    i++;
}
```

Nesse exemplo, o laço de repetição while é usado para imprimir os valores de `i` de 0 a 4. A variável `i` é inicializada com 0 e o bloco de código dentro do laço é executado enquanto `i` for menor que 5. A cada iteração, o valor de `i` é impresso e incrementado em 1. A saída desse exemplo seria:

```
O valor de i é: 0
O valor de i é: 1
O valor de i é: 2
O valor de i é: 3
O valor de i é: 4
```

O laço de repetição while é útil quando você não sabe quantas vezes o bloco de código deve ser executado, mas sabe que ele deve ser executado enquanto uma determinada condição for verdadeira. É importante garantir que a condição eventualmente se torne falsa para evitar loops infinitos.

Segue um outro exemplo de uso do while:

```
import java.util.Scanner;

public class ExemploWhile {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
```

```
        int numero;

        System.out.print("Digite um número positivo: ");
        numero = scanner.nextInt();

        while (numero <= 0) {
            System.out.print("O número digitado não é positivo. Digite
novamente: ");
            numero = scanner.nextInt();
        }

        System.out.println("O número digitado foi: " + numero);
    }
}
```

Nesse exemplo, o laço `while` é usado para verificar se o número digitado pelo usuário é positivo. Se o número não for positivo, o programa solicitará que o usuário digite novamente até que um número positivo seja inserido. O número de iterações pode variar em cada execução, dependendo do número de tentativas necessárias para que o usuário insira um número positivo.

Observe que, neste exemplo, **não é apropriado usar um laço for**, pois o **número de iterações não é conhecido antecipadamente** e não está relacionado ao tamanho de um intervalo definido de valores.
O laço while é a melhor escolha para lidar com esse tipo de situação em que a condição de repetição não depende diretamente do número de iterações.

for vs while:

Em resumo, o laço for é utilizado quando se sabe de antemão o número de iterações a serem executadas, enquanto o laço while é utilizado quando a condição de repetição não depende diretamente do número de iterações.

No laço for, a inicialização, condição e atualização são declaradas na mesma linha, enquanto no laço while, a condição é declarada na primeira linha e a atualização é realizada dentro do bloco do laço.

Em outras palavras, o laço for é mais adequado para percorrer um intervalo de valores definido ou quando o número de iterações é conhecido antecipadamente, enquanto o laço while é mais adequado para situações em que a condição de repetição é mais complexa e não depende diretamente do número de iterações.

do-while:

O laço do-while em Java é semelhante ao laço while, **mas a condição de repetição é verificada no final do bloco do laço, garantindo que o bloco do laço seja executado pelo**

menos uma vez, independentemente da condição de repetição. Em outras palavras, o bloco do laço é executado primeiro e depois a condição é verificada.

Aqui está um exemplo de uso do laço do-while em Java para solicitar ao usuário que insira um número positivo:

```java
import java.util.Scanner;

public class ExemploDoWhile {
    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("O número digitado foi: " + numero);
    }
}
```

Nesse exemplo, o laço `do-while` é usado para solicitar ao usuário que insira um número positivo.
O bloco do laço, que solicita ao usuário que insira um número, **é executado pelo menos uma vez**, independentemente da condição de repetição. Depois que o usuário inserir um número positivo, o bloco do laço será executado novamente somente se o usuário inserir um número não positivo.

O laço `do-while` é mais adequado para situações em que é necessário executar o bloco do laço pelo menos uma vez, independentemente da condição de repetição.

Thiago André Cardoso Silva
twitter/instagram: **@programador_who**