Linguagem Java

Nesta apresentação, você aprenderá os fundamentos da linguagem de programação Java. Exploraremos as estruturas de decisão, que são essenciais para tomadas de decisão durante a construção de aplicações robustas e eficientes.

by Alexandre de Souza Jr.

Estruturas de decisão

1 Tomada de Decisões

As estruturas de decisão permitem que o programa tome decisões com base em condições específicas, guiando o fluxo do programa.

2 Operadores Relacionais

Esses operadores, como >, <, ==, !=, >=, <= são usados para comparar valores e retornar um resultado booleano.

3 Operadores Lógicos

Os operadores lógicos AND (&&), OR (||) e NOT (!) combinam múltiplas condições para criar expressões booleanas mais complexas.

4 Fluxo de Controle

As estruturas de decisão controlam o fluxo de execução do programa, permitindo que ele tome caminhos diferentes com base nas condições avaliadas.

Declarações if-else

1

Estrutura básica

As declarações if-else permitem que você execute diferentes blocos de código com base em uma condição. A estrutura básica é: if (condição) { código } else { código }.

2

Múltiplas condições

Você pode encadear múltiplas declarações if-else para lidar com várias condições, usando else if. Isso fornece um controle de fluxo mais granular em seu programa.

3

Boas práticas

É importante manter suas declarações if-else concisas e legíveis. Evite aninhar muitas condições e use variáveis significativas para deixar o código mais autoexplicativo.

Operadores relacionais e lógicos

Igualdade

Usar o operador de igualdade (==) para verificar se dois valores são iguais.

Desigualdade

Usar o operador de desigualdade (!=) para verificar se dois valores são diferentes.

Maior que

Usar o operador maior que (>) para verificar se um valor é maior que outro.

Menor que

Usar o operador menor que (<) para verificar se um valor é menor que outro.

Estruturas de seleção aninhadas

1

Decisões encadeadas

Casos complexos exigem mais de uma condição.

2

Verificações múltiplas

Analisar diferentes cenários de forma organizada.

3

Controle aprofundado

Maior precisão na tomada de decisão.

As estruturas de seleção aninhadas permitem criar decisões complexas, com múltiplas condições. Isso possibilita um controle mais aprofundado do fluxo do programa, analisando diferentes cenários de forma organizada e tomando decisões mais precisas.

Operador ternário

O que é o operador ternário?

O operador ternário, também conhecido como operador condicional, é uma forma concisa de escrever declarações if-else em Java. Ele permite tomar uma decisão com base em uma condição e retornar um valor apropriado.

Sintaxe do operador ternário

A sintaxe do operador ternário é:

condição ? valor_se_verdadeiro : valor_se_falso.

Ele avalia a condição e retorna o valor à esquerda se for verdadeira, ou o valor à direita se for falsa.

Declarações switch-case

Múltiplas opções

Selecione entre várias alternativas.

Comparações exatas

Testa igualdade de valores.

Execução contínua

Executa código até encontrar um break.

As declarações **switch-case** em Java permitem que você selecione um bloco de código para executar com base em uma expressão. Elas são úteis quando você tem múltiplas opções a serem avaliadas, pois oferecem uma sintaxe mais concisa do que uma sequência de declarações **if-else**. Cada **case** é comparado com o valor da expressão **switch**, e o código do primeiro **case** que corresponde é executado.

Uso de break e default

Declaração break

A declaração **break** é usada para interromper imediatamente um loop ou uma estrutura de seleção, permitindo que o programa prossiga para a próxima instrução.

Declaração default

A cláusula **default** em uma estrutura **switch-case** é utilizada para especificar um bloco de código a ser executado quando nenhum dos casos correspondentes for encontrado.

Boas práticas

É recomendado usar **break** dentro de loops e **default** em estruturas **switch-case** para garantir o fluxo de execução correto do programa.

Boas práticas em estruturas de decisão

Ao utilizar estruturas de decisão em Java, é importante seguir algumas boas práticas para garantir um código mais eficiente, legível e fácil de manter. Evite condições complexas, prefira usar operadores lógicos e relacionais, e organize o fluxo de decisão de forma clara e intuitiva.

Além disso, documente bem o propósito de cada estrutura de decisão e considere casos especiais, como valores nulos ou limites extremos. Isso ajudará outros desenvolvedores a entenderem e manterem o seu código no futuro.

Exercícios práticos e aplicações

Exercícios de lógica

Resolva problemas lógicos envolvendo estruturas de decisão, como identificar o maior número entre três valores ou determinar se um ano é bissexto.

Projetos do mundo real

Desenvolva aplicações práticas que utilizem estruturas de decisão, como um sistema de cadastro de clientes com validação de dados ou um sistema de controle de acesso a uma sala.

Desafios de programação

Participe de competições de programação e resolva problemas complexos que envolvam o uso de estruturas de decisão, como algoritmos de busca e classificação.