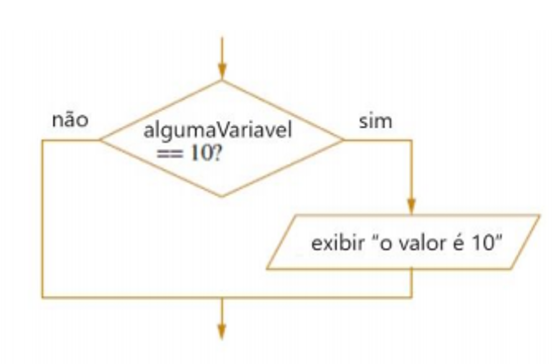
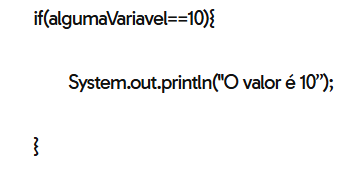
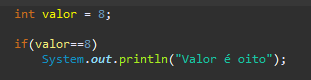
**IF:**

Se você precisar alterar a execução do programa com base em uma **determinada condição**, poderá usar as instruções “if”.



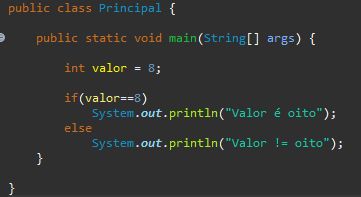


É o **famoso se** , no Java a é utilizado da seguinte maneira:

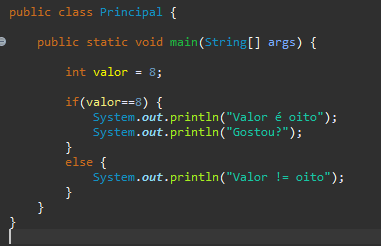


**Else:**

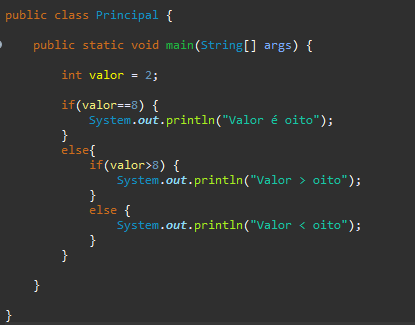
Caso o if for falso ele faça outra coisa. É utilizada assim:



Mas se eu tiver mais de uma instrução, vamos perceber que ela não vai executar! Para solucionar isso é preciso colocar o **if entre chaves**:



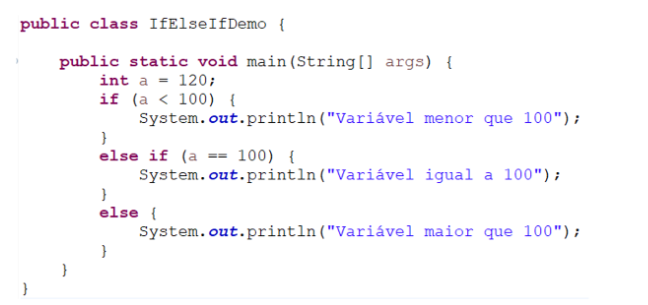
**Podemos usar vários if e vários else**:



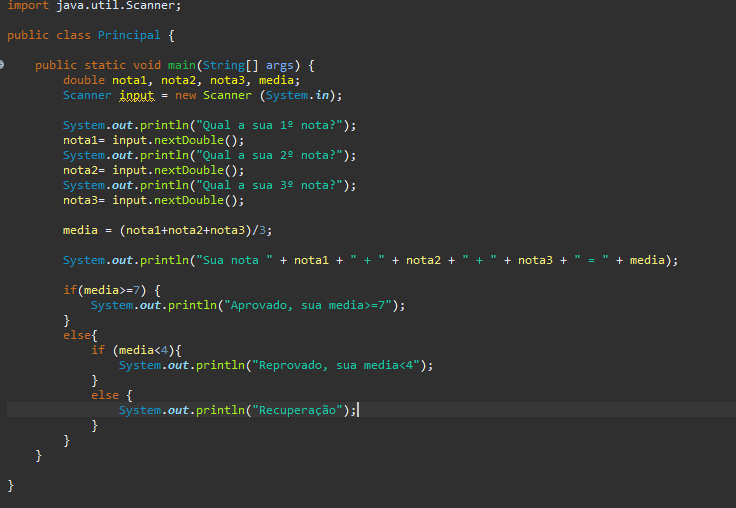
Mas para isso também temos o:

**Elseif:**

Podemos ter uma situação em que precisamos verificar o valor várias vezes para encontrar a condição exata de correspondência. (**então se, senão, se**)

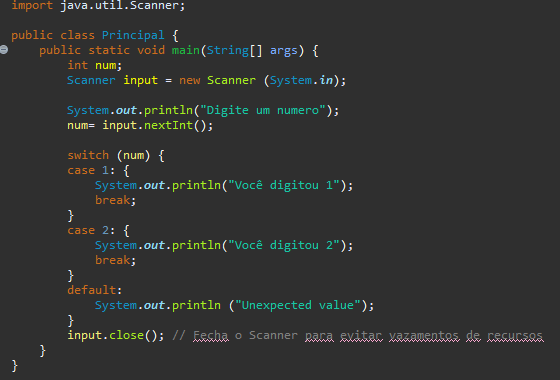


**Exercício:**



**Switch:**

Uma linguagem de programação usa instruções de controle para fazer com que o fluxo de execução avance e se ramifique com base nas alterações no estado de um programa. Java suporta duas instruções de controle de fluxo: if e switch. Estas instruções permitem controlar o fluxo da execução do seu programa com base nas condições conhecidas apenas durante o tempo de execução.



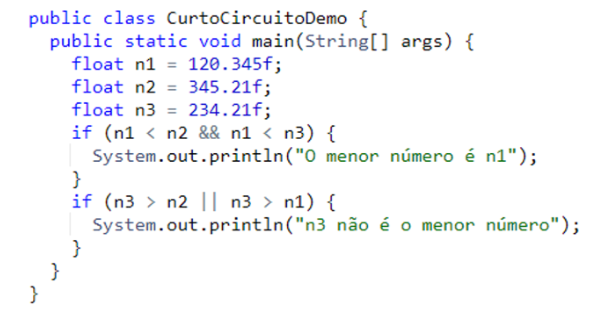
1. **A opção só pode verificar a igualdade**. Isso significa que os outros operadores relacionais, como maior que, são inutilizados em um caso.
2. As constantes de caso são avaliadas de cima para baixo, e a primeira constante de caso que corresponde à expressão do comutador é o ponto de entrada de execução. Se nenhuma instrução de interrupção for usada, todo o caso após o ponto de entrada será executado.
3. **Nenhuma constante de dois casos no mesmo comutador pode ter valores idênticos**. Obviamente, uma declaração de chave e uma chave externa anexa podem ter constantes de maiúsculas e minúsculas em comum.
4. O caso padrão pode estar localizado no final, meio ou parte superior. Geralmente, o padrão aparece no final de todos os casos.

* **Operadores Lógicos de curto circuito:**

**&&** que é o E

**||** que é o OU

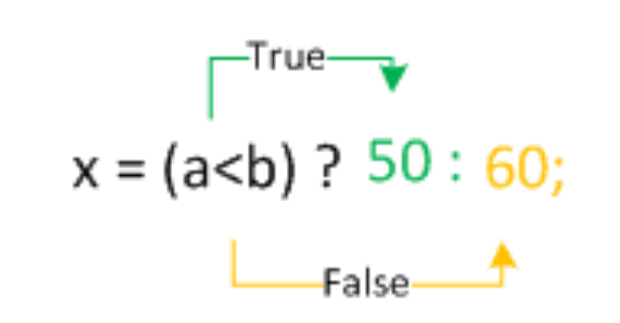
Os operadores && e || avaliam apenas valores booleanos.



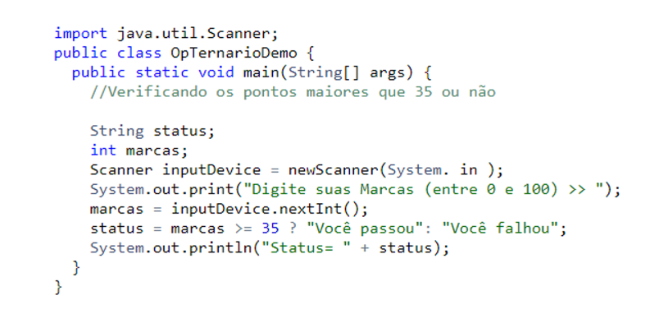
* **Operadores condicional ternário:**

O operador condicional é um operador ternário (**possui três operandos**) e é usado para avaliar expressões booleanas, como uma instrução if, exceto em vez de executar um bloco de código **se o teste for verdadeiro**, um operador condicional atribuirá um valor a uma variável.

Um operador condicional começa com uma operação booleana, seguida por dois valores possíveis para a variável à esquerda do operador de atribuição (=). O primeiro valor (aquele à esquerda dos dois pontos) é atribuído se o teste condicional (booleano) for verdadeiro e o segundo valor é atribuído se o teste condicional for falso. No exemplo abaixo, se a variável a for menor que b, o valor da variável x seria 50 ou se for maior, x = 60.



**Ele é X= (condição)? verdadeiro: falso**



* **Operador NOT:**

Também chamado de complemento bit a bit, o operador NOT unário, ~, inverte todos os bits do seu operando. Se aplicado no operando inteiro, ele reverterá todos os bits da mesma forma. Se aplicado ao literal booleano, ele será revertido.

int a = 23; // 23 é representado em binário como 10111

int b = ~ a; // isso reverte os bits 01000, que são 8 em decimal boolean x = true;

boolean y =! x; // Isso atribuirá valor falso a y, pois x é verdadeiro

* **Operador AND:**

O operador AND **“&”** produz 1 bit se ambos os operandos forem 1 caso contrário 0 bit. Da mesma forma, para operandos booleanos, resultará em true se ambos os operandos forem verdadeiros, caso contrário, o resultado será false.

int var1 = 23; // valor booleano seria 010111

int var2 = 33; // valor booleano seria 100001

int var3 = var1 & var2 // resulta no binário 000001 e no decimal 1 b1 boolean = true;

boolean b2 = false;

boolean b3 = b1 & b2; // b3 seria falso

* **Operador OR:**

O operador OR **“|”** produz um bit 0 se ambos os operandos forem 0, caso contrário, 1 bit. Da mesma forma, para operandos booleanos, resultará em false se ambos os operandos forem falsos ou o resultado será true.

int var1 = 23; // valor booleano seria 010111

int var2 = 33; // valor booleano seria 100001

int var3 = var1 | var2 // resulta no binário 110111 e no decimal 55 b1 boolean = true;

boolean b2 = false;

boolean b3 = b1 | b2; // b3 seria verdadeiro

* **Operador XOR (Ou exclusivo):**

O operador XOR “^” produz um bit 0 se ambos os operandos forem iguais (ambos 0 ou 1) caso contrário, 1 bit. Da mesma forma, para operandos booleanos, resultará em false se os dois operandos forem iguais (ambos são falsos ou ambos verdadeiros) ou o resultado será true.

int var1 = 23; // valor booleano seria 010111

int var2 = 33; // valor booleano seria 100001

int var3 = var1 ^ var2 // resulta no binário 110110 e no decimal 54 boolean b1 = true;

boolean b2 = false;

boolean b3 = b1 ^ b2; // b3 seria verdadeiro

