**Escopo – Palavras reservadas - Documentação**

**Objetivo do curso**

Identificar a localização mais conveniente para a escrita de algoritmos necessário para o nosso programa.

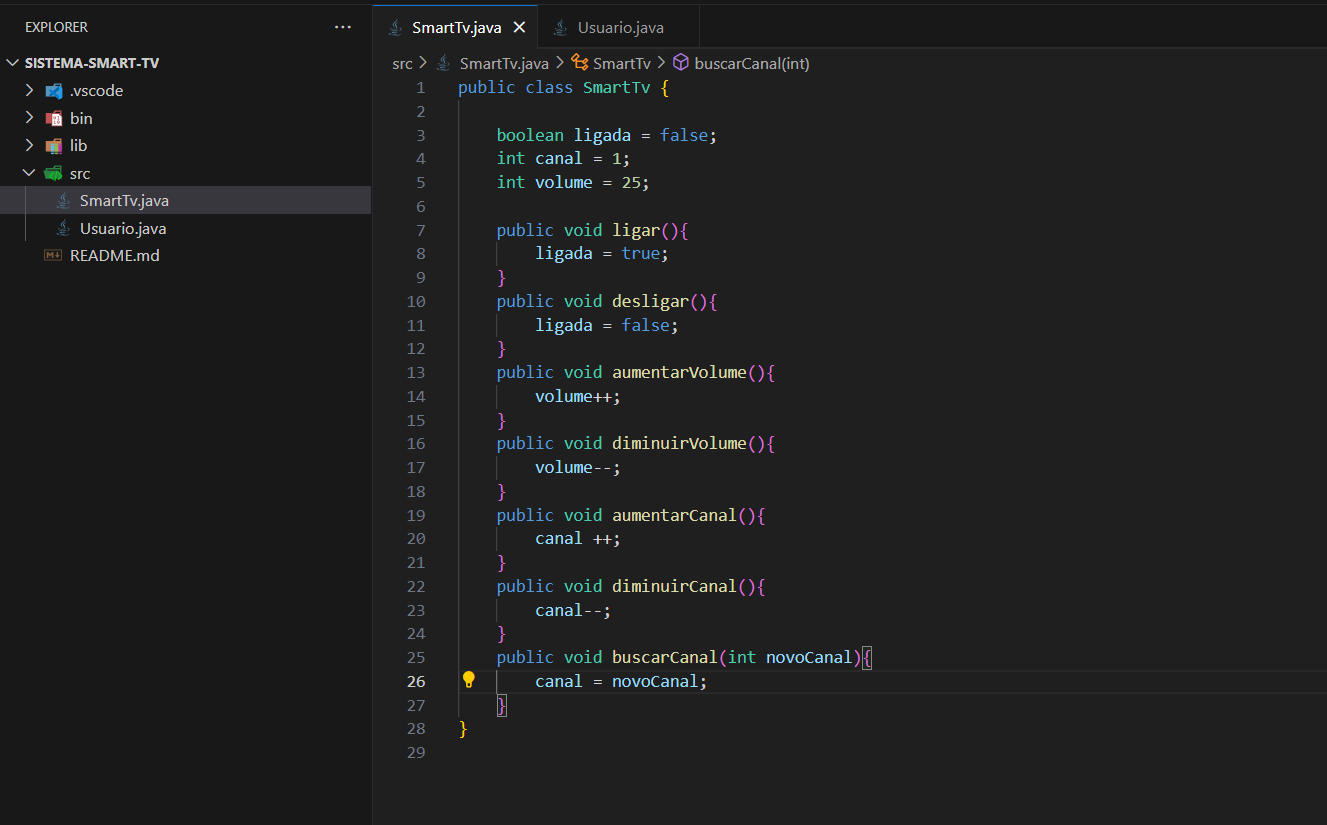
**Assuntos**

* Escopo de classe
* Escopo de método
* Escopo de fluxo

**Escopo**

* É o lugar, nível ou local onde uma variável pode ser acessada.
* Escopo de variáveis vai de acordo com o bloco onde ela foi declarada.
* Variável pode ser acessada apenas no seu bloco de execução ao qual pertence, portanto não pode ser lida ou manipulada por rotinas e códigos que estão fora do seu bloco de declaração, ou seja, fora do **escopo de variável.**
* Em uma Classe, podemos visualizar a diferença de escopos. Atributos (variáveis e constates) são declarados no corpo (escopo) principal da classe, sendo, portanto, acessíveis por todos os métodos.
* Dentro de um método, o escopo de uma variável está limitado apenas ao corpo desse método, ou seja, dentro das chaves que limitam o método

É fundamental na elaboração de algoritmos simples ou complexos, determinar a localização do código. Sem um domínio sobre o escopo de códigos, seu projeto tende a conter falhas estruturais comprometendo a proposta principal da aplicação.



Podemos ver na imagem acima que as variáveis

1. boolean ligada
2. int canal
3. int volume

Estão declaradas no escopo da classe, tornando-a visível para todos os métodos. Já a variável **novoCanal** é um parâmetro (variável de assinatura) do método que pode ser vista somente dentro desse escopo.

**Palavras reservadas**

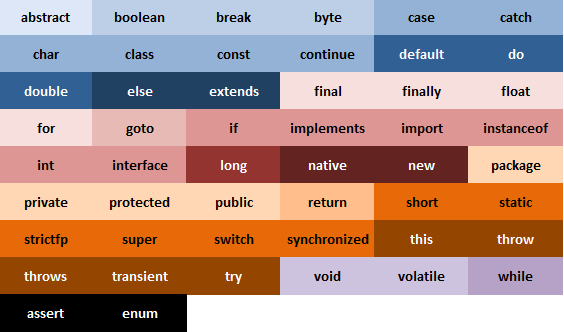
**Objetivo do curso**

Apresentar as 52 palavras reservadas organizadas por classificação de usabilidade considerando as regras da linguagem.

**Assuntos**

* Tipos primitivos
* Classificações
* Escopo de uso
* Palavras “opostas”

Todas as 52 palavras reservadas são escritas com letras minúsculas e geralmente as IDEs identificam elas com uma cor especial.



**Controle de pacotes**

**import.:**  importa pacotes ou classes para dentro do código

**packages:** especifica a que pacote todas as classes de um arquivo pertencem

**Modificadores de acesso**

**public:** acesso de qualquer classe

**private:** acesso apenas dentro da classe

**protected:** acesso por classe no mesmo pacote e subclasses

**Primitivos**

**boolean:** um valor indicando verdadeiro ou falso

**byte:** um inteiro de 8 bits (signed)

**char:** Character Unicode (16-bit unsigned)

**double:**  um número de ponto flutuante de 64 bits (signed)

**float:** um número de ponto flutuante de 32 bits (signed)

**int:** um inteiro de 32 bits (signed)

**long:** um inteiro de 64 bits (signed)

**short:** um inteiro de 32 bits (signed)

**void:** indica que o método não tem retorno de valor

**Modificadores de classes, variáveis ou métodos**

**class:** especifica uma classe

**extends:** indica a superclasse

**abstract:** classe que não pode ser instanciada ou método que precisa ser implementado por uma subclasse não abstrata

**final:** impossibilita que uma classe seja estendida, que um método seja sobrescrito ou que uma variável seja reiniciada

**implemets:** indica as interfaces que uma classe irá implementar

**interface:** especifica uma interface

**native:** indica que um método está escrito em linguagem dependente de plataforma, como o C

**new:** instancia um novo objeto, chamando seu construtor

**static:** faz um método ou variável pertencer à class ao invés de às instâncias

**strictfp:** usado em frente a um método ou classe para indicar que os números de ponto flutuante seguirão as regras de ponto flutuante em todas as expressões

**synchronized:** indica que em método só pode ser acessado por um thread de cada vez

**transiente:** impede a serialização de campos

**volatile:**  indica que uma variável pode ser alterada durante o uso de threads

**Controle de fluxo dentro de um bloco de código**

**break:** sai do bloco de código em que ele está

**case:** executa um bloco de código dependendo do teste do switch

**continue:**  pula a execução do código que viria após essa linha e vai para a próxima passagem do loop

<https://github.com/digitalinnovationone/trilha-java-basico>

**Documentação**

É imprescindível que falemos deste assunto, pois a linguagem JAVA traz para seus usuários várias formas de documentação que facilitam a interpretação do código no futuro.

Texto

Descrição gerada automaticamente

**JavaDoc**

API para gerar documentação em programas Java, a partir de código-fonte, resultando em uma expressão HTML.

**Executando o Javadoc**

1. Dentro do diretório entre no terminal
2. No terminal escreva e execute: Javadoc -encoding UTF-8 – docencoding ISSO-8859-1 -d ../docs src/\*.java

Onde **UTF-8** é para patrão português Brasil **-d** é de diretório **../**  para caminho de diretório **\*** para todos os arquivos.

1. Será gerado uma pasta dentro do seu diretório chamada docs onde estará a sua documentação.