

# Palavras reservadas



Palavras reservadas, são identificadores de uma linguagem que já possuem uma finalidade específica, portanto, não podem ser utilizados para nomear variáveis, classes, métodos ou atributos.

A linguagem Java possui 52 palavras reservadas. Todas essas palavras são classificadas em grupos e escritas com letra minúscula, sendo identificadas com uma cor especial pela maioria das IDE's. Abaixo temos a lista de palavras agrupadas por suas finalidades.

## Palavras Reservadas em Java

<code>abstract</code>	<code>continue</code>	<code>goto</code>	<code>package</code>	<code>synchronized</code>
<code>assert</code>	<code>default</code>	<code>if</code>	<code>private</code>	<code>this</code>
<code>boolean</code>	<code>do</code>	<code>implements</code>	<code>protected</code>	<code>throw</code>
<code>break</code>	<code>double</code>	<code>import</code>	<code>public</code>	<code>throws</code>
<code>byte</code>	<code>else</code>	<code>instanceof</code>	<code>return</code>	<code>transient</code>
<code>case</code>	<code>extends</code>	<code>int</code>	<code>short</code>	<code>try</code>
<code>catch</code>	<code>final</code>	<code>interface</code>	<code>static</code>	<code>void</code>
<code>char</code>	<code>finally</code>	<code>long</code>	<code>strictfp</code>	<code>volatile</code>
<code>class</code>	<code>float</code>	<code>native</code>	<code>super</code>	<code>while</code>
<code>const</code>	<code>for</code>	<code>new</code>	<code>switch</code>	

### Controle de pacotes

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

**package:** 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 classes no mesmo pacote e subclasses.

### Primitivos

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

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

**char:** um 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

---

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

**class:** especifica uma classe;

**extends:** indica a superclasse que a subclasse está estendendo;

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

**implements:** indica as interfaces que uma classe irá implementar;

**interface:** especifica uma interface;

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

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

**static:** faz um método ou variável pertencer à classe 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 um método só pode ser acessado por uma thread de cada vez

**transient:** impede a serialização de campos;

## 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;

**default:** executa esse bloco de código caso nenhum dos teste de switch-case seja verdadeiro;

**do:** executa um bloco de código uma vez, e então realiza um teste em conjunto com o while para determinar se o bloco deverá ser executado novamente;

**else:** executa um bloco de código alternativo caso o teste "if" seja falso;

**for:** usado para realizar um loop condicional de um bloco de código;

**if:** usado para realizar um teste lógico de verdadeiro ou falso;

**instanceof:** determina se um objeto é uma instância de determinada classe, superclasse ou interface;

**return:** retorna um método sem executar qualquer código, que venha depois desta linha (também pode retornar uma variável);

**switch:** indica a variável a ser comparada nas expressões case;

**while:** executa um bloco de código repetidamente enquanto a condição for verdadeira.

## Tratamento de erros

---

**assert:** testa uma expressão condicional, para verificar uma suposição do programador;

**catch:** declara o bloco de código usado para tratar uma exceção;

**finally:** bloco de código, após um try-catch, que é executado independentemente do fluxo de programa seguido ao lidar com uma exceção;

**throw:** usado para passar uma exceção para o método que o chamou;

**throws:** indica que um método pode passar uma exceção para o método que o chamou;

**try:** bloco de código que tentará ser executado, mas que pode causar uma exceção.

## Variáveis de referência

---

**super:** refere-se a superclasse imediata;

**this:** refere-se a instância atual do objeto.

## Palavras reservadas não utilizadas

**const:** Não utilize para declarar constantes; use `public static final`;  
**goto:** não implementada na linguagem Java, por ser considerada prejudicial.

## Literais reservados

De acordo com a Java Language Specification, **null**, **true** e **false** são tecnicamente chamados de valores literais, e não keywords. Se você tentar criar algum identificador com estes valores, você também terá um erro de compilação.

## Escopo de uso

Uso	Palavras	Observação
Arquivo	<code>package</code> , <code>import</code> , <code>static</code> .	
Classe	<code>public</code> ou <code>protected</code> ou <code>private</code> + <code>final</code> ou <code>abstract</code> + <code>extends</code> ou <code>implements</code> .	<b>private</b> (em caso de classe interna), <b>final</b> ou <b>abstract</b>
Método	<code>public</code> ou <code>protected</code> ou <code>private</code> + <code>static</code> ou <code>final</code> ou <code>abstract</code> + <code>void</code> e <code>return</code> .	<b>void</b> em caso de não ter retorno ou <b>return</b> se houver
Atributo	<code>public</code> ou <code>protected</code> ou <code>private</code> + <code>static</code> ou <code>final</code> + tipo primitivo.	****

## Palavras "opostas"

Assim como nas classificações gramaticais da língua portuguesa, existem algumas palavras que são completamente opostas (antônimas) na linguagem Java conforme tabela abaixo:

Palavra	Palavra	Explicação
<code>package</code>	<code>import</code>	Enquanto <b>package</b> determina o diretório real da classe, o <b>import</b> informa de onde será importada a classe. Isso porque, podemos ter classes de mesmo nome.
<code>extends</code>	<code>implements</code>	enquanto <b>extends</b> determina que uma classe estende outra classe, <b>implements</b> determina que uma classe implementa uma interface, porém nunca o contrário.

final	abstract	enquanto <b>final</b> determina fim de alteração de valor ou lógica comportamental, <b>abstract</b> em métodos, exige que sub-classes precisarão definir comportamento e um método abstrato. NOTA: Se uma classe contém um único método abstrato, toda classe precisa ser.
throws	throw	Esta é uma das situações mais complicadas, de compreensão destas duas palavras. Enquanto a <b>throws</b> determina que um método pode lançar uma exceção, <b>throw</b> é a implementação que dispara a exceção**. Vamos conhecer mais sobre este conceito no assunto

## Referência

As 52 palavras reservadas do Java



Sintaxe - Previous

Escopo

Next - Sintaxe

Documentação



Last modified 1mo ago