Java 15 Language Specification
Java 18 Language Specification
The seven lexical tokens in Java:

- white space
- comments and Java Doc comments
- punctuation aka separators or delimiters
- identifiers (Unicode letters)
- Java 18 keywords and Contextual Keywords
- Java 18 literals
- operators
• “end of line” comments: // ... line-terminator
• block: /* ... */
  These may include line-terminators, but not nested block comments.
  • javadoc comments: /** ... */
// Hello.java -- using Unicode ch\u0041racters

// \u002F = /    \u0041 = A
// \u0029 = )    \u0061 = a
// \u002E = .    \u0065 = e

cl\u0061ss H\u0065llo {  
    public static void main (String \u0041rgs[]\u0029 {  
        Syst\u0065m.out\u002Eprintln ("¡Hol\u0061 mundo!");  
    }  

    public static int größtergernsamerTeiler (int x, int y) {  
        return (0);  \u002F/ This is an odd comment  
    }  
}
Java 15; 50 Keywords

The keywords `const` and `goto` are reserved, even though they are not currently used. This may allow a Java compiler to produce better error messages if these C++ keywords incorrectly appear in programs. The keyword `_` (underscore) is reserved for possible future use in parameter declarations.
The Java tutorial at Oracle is a good reference.

data types tutorial
1 literals of type int, long
2 literals of type float, double
3 literals of type character
4 literals of type String, text block
5 null
6 Java 18 Class literals
Hexadecimal Floating-Point Literals

[Do literals belong under the topic of expressions or here under data? Under data!]

Hexadecimal floating-point literals originated in C99 and were later included in a revision of the IEEE 754 floating-point standard. sign, significand, and exponent fields defining a finite floating-point value; sign0xsignificandpexponent. This syntax allows the literal

\[0x1.8p1\]

to be used to represent the value \(3; 1.8_{16} \times 2^1 = 1.5_{10} \times 2 = 3\). More usefully, the maximum value of can be written as \(0x1.fffffffffffffp1023\) and the minimum value of \(2^{-1074}\) can be written as \(0x1.0P-1074\) or \(0x0.0000000000001P-1022\), which maps easily to the various fields of the floating-point representation and is much more scrutable than the raw-bit encoding.

In addition, "printf" facility including the %a format for hexadecimal floating-point.
Separators

() {} [] ; , . . . @ ::
Operators

=  >  <  !  ~  ?  :  ->
==  >=  <=  !=  &&  ||  ++  --
+  -  *  /  &  |  ^  %  <<  >>  >>>
+=  -=  *=  /=  &=  |=  ^=  %=  <<=  >>=  >>>=
```java
class Main {
public static void main(
final String [] args){
    // Do not make useless comments!
    System.out.println (null=="Hello world"+"!"+42);
}}
```