A lexme is a sequence of characters constituting a fundamental unit of a programming language somewhat like words in natural language used to build more complex and significant grammatical constructs.

A Unicode escape, like \u0065, is an alternative name or representation for a character using the Unicode standard’s codepoint in hexadecimal digits. It is not a lexme, to Java it is just another character out of which lexmes are build.
Like words and punctuation in an English composition, every little character of a Java program is part of one of seven things. These things are called tokens or lexems. The seven lexical tokens in Java:

1. **white space**
2. **comments** and Java Doc comments
3. **punctuation** aka separators or delimiters
4. **identifiers** (Unicode letters)
5. **Java 18 keywords and Contextual Keywords**
6. **Java 18 literals**
7. **operators**
White space is defined as the ASCII space character, horizontal tab character, form feed character, and line terminator characters.

A line terminator is one of:

- the ASCII LF character, also known as "newline"
- the ASCII CR character, also known as "return"
- the ASCII CR character followed by the ASCII LF character

“As a special concession for compatibility with certain operating systems, the ASCII SUB character (\u001a, or control-Z) is ignored if it is the last character” in the input.

Don’t use tab, form feed, or sub.
- “end of line” comments: // ... line-terminator. Both single-line and trailing.
- block: /* ... */ May include line-terminators, but not nested block comments.
  - javadoc comments: /** ... */

Java supports literate programming by having tools the make the source code executable and extract the documentation as structured HTML files.

**Definition**

The practice of literate programming combines source code and documentation into a single source file.
Identifiers: letters (of unicode), digits, _, $. The $ is intended for use in computer generated Java code or to access names in legacy code, not but for use in ordinary programming.

- Identifier.java

String
i3
MAX_VALUE
// Hello.java -- using Unicode ch\u0041racters

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

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

    public static int größtergermentsamerTeiler (int x, int y) {
        return (0);  \u002F/ This is an odd comment
    }
}
Java 19; 51 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 `strictfp` is obsolete and should be used anymore. The keyword `_` (underscore) is reserved for possible future use in parameter declarations.
Java 19; 16 Contextual Keywords

exports module non-sealed open opens permits provides record
requires sealed to transitive uses var with yield
1 taking words in their usual or most basic sense without metaphor or allegory. “dreadful in its literal sense, full of dread”

2 (of a translation) representing the exact words of the original text. “a literal translation from the Spanish”

3 (in a programming language) tokens or lexemes representing directly a specific value
The Java tutorial at Oracle is a good reference.
1. literals of type int, long
2. literals of type float, double
3. literals of type character (character escapes)
4. literals of type boolean (true and false)
5. literals of type String, text block ""
6. null
7. Java 18 Class literals
Text Blocks

• Block.java, an example of text blocks
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. These literals are represented without loss in standard hardware (unlike decimal literals).

\[0x1.8p1\]

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 \(0x1.ffffffffffffffffp1023\) 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 more readable than the raw-bit encoding. In addition, "printf" facility including the %a format for hexadecimal floating-point.
Three literals are composed of just letters: true, false, and null. These are not keywords, but they are not legal identifiers either. They are literals.

```java
int true = 20; // syntax error: "not a statement"
int false = 30; // syntax error: "not a statement"
float null = 23.6f; // syntax error: "not a statement"
int record = 10; // contextual keywords can be used as ide
Delimiters/Separators/Punctuation

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

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