Backslash (Escape) Characters in Java 21

In Java, a backslash \ is used to create **escape sequences** — special character combinations that represent characters you can't easily type directly in source code.

Common Escape Characters:

Sequence	Name	Description:
\t	Tab	Inserts a horizontal tab
\n	Newline	Moves to the next line
\r	Carriage Return	Moves cursor to start of line
\\\\	Backslash	Inserts a literal backslash
\\"	Double Quote	Inserts " inside a String literal
\\'	Single Quote	Inserts ' inside a char literal
\0	Null	Null character (char value 0)

Examples:

\t — Tab

```
System.out.println("Name:\tAlice");
System.out.println("Age:\t30");
// Output:
// Name:   Alice
// Age:    30
```

\n — Newline

```
System.out.println("Line one\nLine two\nLine three");
// Output:
// Line one
// Line two
// Line three
```

\\ — Literal Backslash

```
System.out.println("C:\\Users\\Alice\\Documents");
// Output:
// C:\Users\Alice\Documents
```

\" — Double Quote inside a String

```
System.out.println("She said \"Hello!\"");
// Output:
// She said "Hello!"
```

Combining them:

```
System.out.println("Column1\tColumn2\tColumn3\nA\tB\tC");
// Output:
// Column1   Column2   Column3
// A         B         C
```

Java 21 Bonus: Text Blocks

If you find yourself using a lot of \n and \", Java 13+ introduced **text blocks** (finalized in Java 15), which are still very relevant in Java 21. They let you write multi-line strings without escape clutter:

```
// Old way
String json = "{\n\t\"name\": \"Alice\",\n\t\"age\": 30\n}";

// Text block way (much cleaner!)
String json = """
      {
         "name": "Alice",
         "age": 30
      }
      """;
```

Key Things to Remember:

- Escape sequences work inside regular String literals ("...") and char literals ('...').
- A lone \ with an unrecognized character after it will cause a **compile error** — always double-check your sequences.
- Text blocks are great when you have lots of \n and \" cluttering your code.