Java 15: Text Block

Thursday, 6 November 2025 6:50 PM

Let's understand the problem:

Following are the 2 common use-cases for using multiline strings in Java code:

```
String query = "SELECT e.id, e.name\n"
        + "FROM employees e\n"
        + "WHERE e.salary > 6000\n"
        + "ORDER BY e.name ASC;";
```

```
String jsonString = "{\n" +
            "    \"name\": \"Shrayansh Jain\",\n" +
            "    \"country\": \"India\",\n" +
            "    \"profession\": \"Software Engineer\"\n" +
            "}";
```

Disadvantage of above multi-line Strings are:

- Difficult to read due to too many special characters:
  - -> \n : newline character
  - -> \\ : escape character
  - -> + : string concatenation

- Error prone $\longrightarrow$ String jsonString = "{\n" +

I intentionally removed the ','
but its very difficult to identify.

```
"    \"name\": \"Shrayansh Jain\"\n" +
"    \"country\": \"India\",\n" +
"    \"profession\": \"Software Engineer\"\n" +
"}";
```

- Hard to maintain, as small change required editing multiple line carefully.

- Not good for Copy Paste, as we can not directly code the SQL from our IDE to DB tool or JSON validator without cleanup.

Solution: Text Block

- It is Finalized in Java 15.
- It's a multi-line string literal used to make writing and reading multi-line text easier.

```
String text = """
        This is a multiline text block.
        Which is easy to read and write.
        """;
```

- It starts and ends with three double quotes **(""")**.

Now lets see the same 2 examples with Text Block:

```
String query = "SELECT e.id, e.name\n"
        + "FROM employees e\n"
        + "WHERE e.salary > 6000\n"
```

$\xrightarrow{\text{to}}$

```
String query = """
   SELECT e.id, e.name
   FROM employees e
   WHERE e.salary > 6000
   ORDER BY e.name ASC;
```

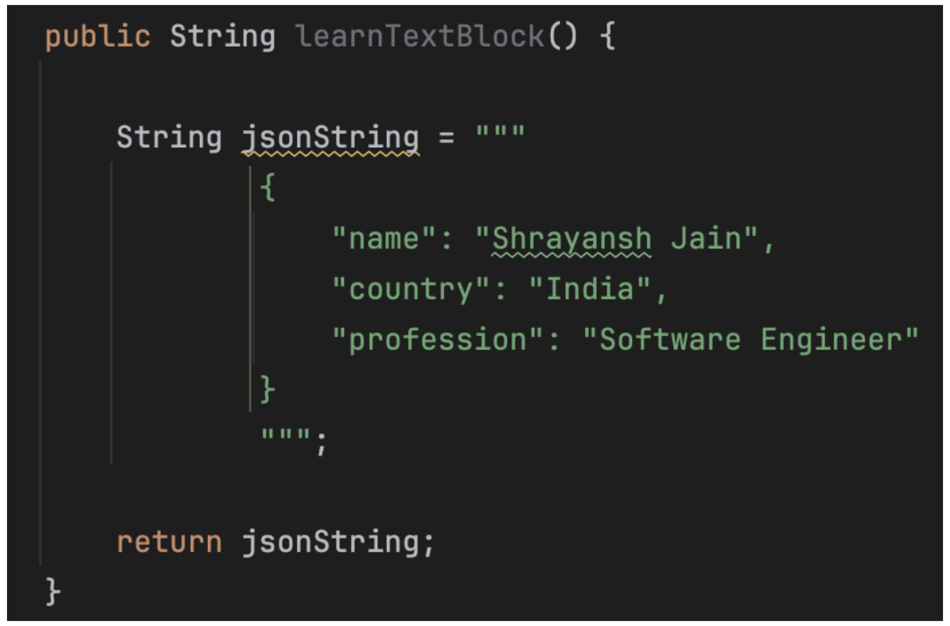
```
+ "ORDER BY e.name ASC;";                    """;
```

```
String jsonString = "{\n" +
             "   \"name\": \"Shrayansh Jain\",\n" +
             "   \"country\": \"India\",\n" +
             "   \"profession\": \"Software Engineer\"\n" +
             "}";
```

$\xrightarrow{\text{to}}$

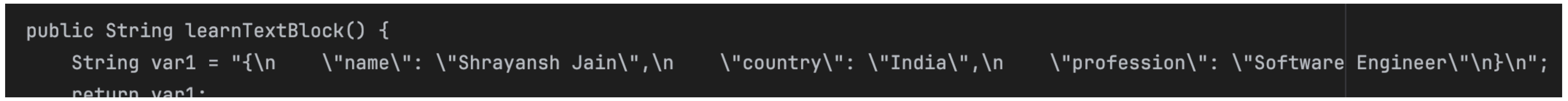
```
String jsonString = """
   {
      "name": "Shrayansh Jain",
      "country": "India",
      "profession": "Software Engineer"
   }
   """;
```

Internally, text block (""") is compiled into a regular String object only.



```
public String learnTextBlock() {

    String jsonString = """
            {
                "name": "Shrayansh Jain",
                "country": "India",
                "profession": "Software Engineer"
            }
            """;

    return jsonString;
}
```

.class file:



```
public String learnTextBlock() {
    String var1 = "{\n    \"name\": \"Shrayansh Jain\",\n    \"country\": \"India\",\n    \"profession\": \"Software Engineer\"\n}\n";
    return var1;
```

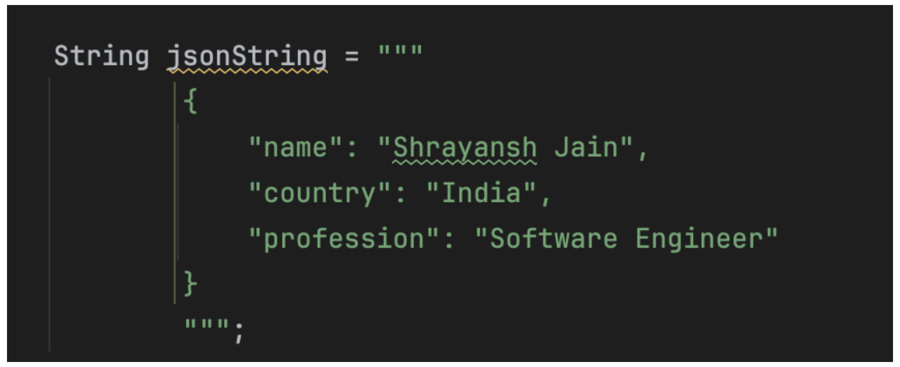
```
    return var1;
}
```

But during compilation it follows certain rules:

1. Opening delimiter:

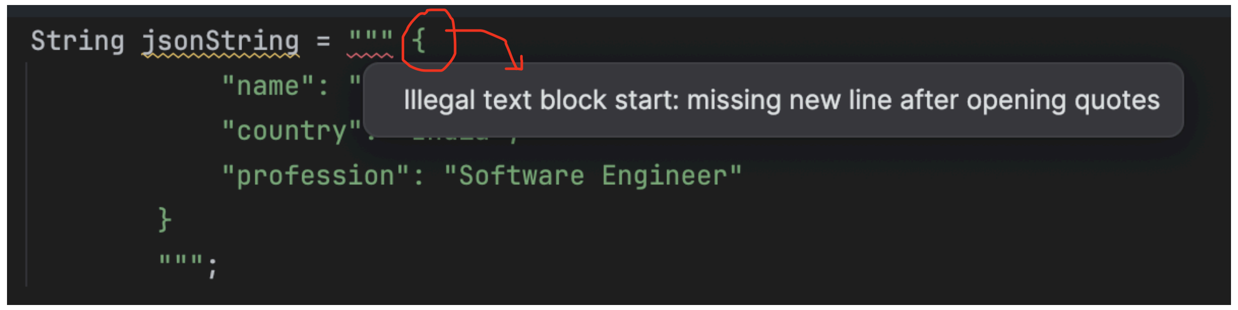
After Opening delimiter (""") text or content is not allowed on the same line.

✓



```
String jsonString = """
        {
            "name": "Shrayansh Jain",
            "country": "India",
            "profession": "Software Engineer"
        }
        """;
```

✗



```
String jsonString = """ {
            "name": "
            "country": "India",
            "profession": "Software Engineer"
        }
        """;
```

Illegal text block start: missing new line after opening quotes

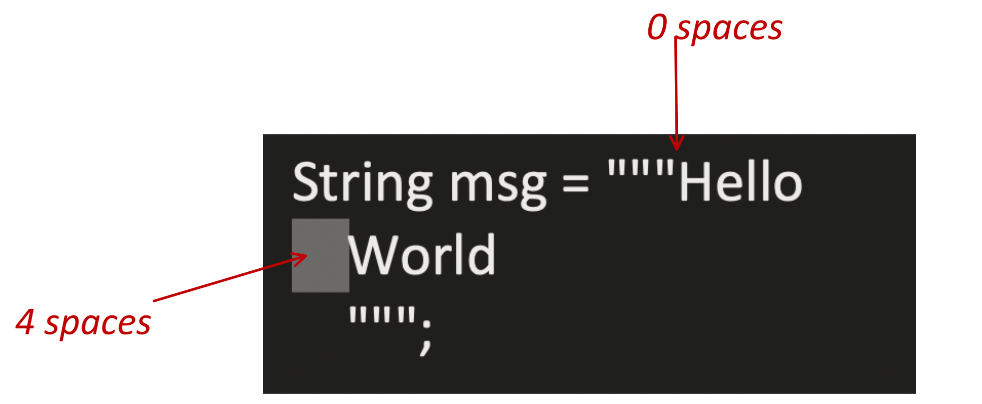
Why so?

Because, compiler tries to remove the indentation (number of spaces before the text).

This rule makes the complier job easy, it can clearly determine how many white spaces need to be remove.

Example:

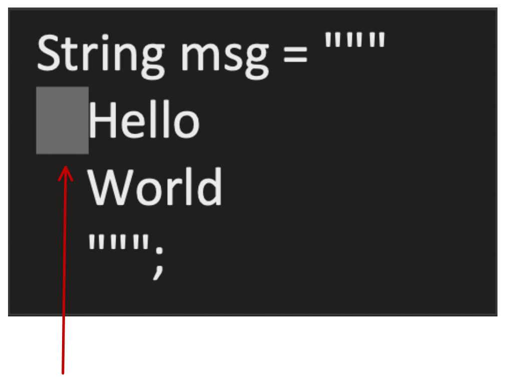
Without the rule:



```
String msg = """Hello
    World
    """;
```

Compiler is confused now, how many front spaces I need to remove, 0 or 4.

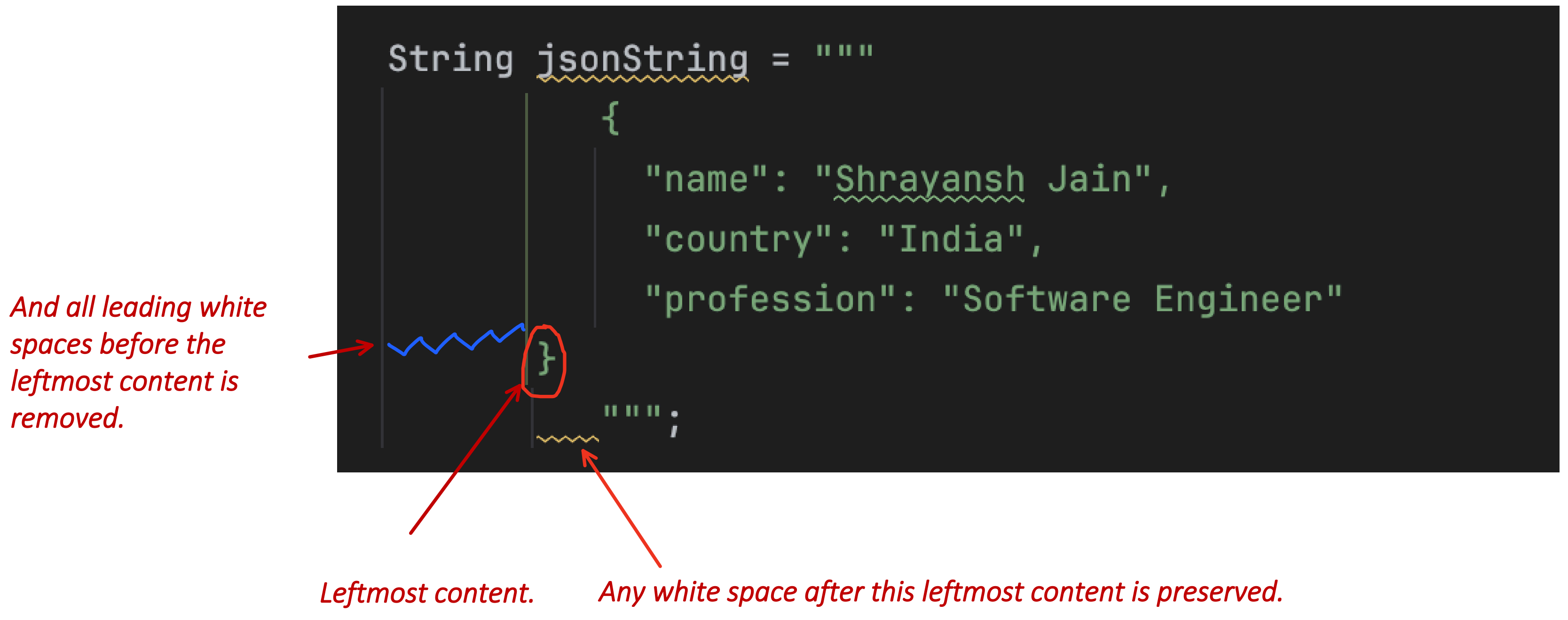
With the rule:



```
String msg = """
    Hello
    World
    """;
```

Compiler can easily determine the number of white spaces it need to remove before the text.

2. Indentation (Leading Whitespaces):
   All white spaces before the leftmost text or content is removed.



```
String jsonString = """
        {
          "name": "Shrayansh Jain",
          "country": "India",
          "profession": "Software Engineer"
      }
        """;
```

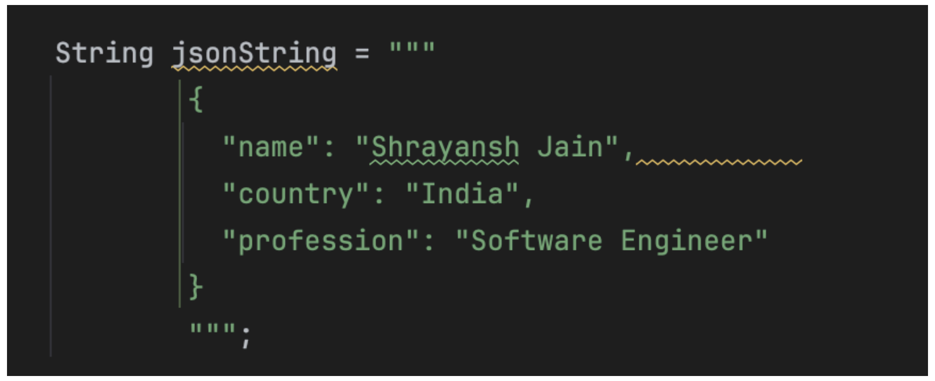
And all leading white spaces before the leftmost content is removed.

Leftmost content.

Any white space after this leftmost content is preserved.

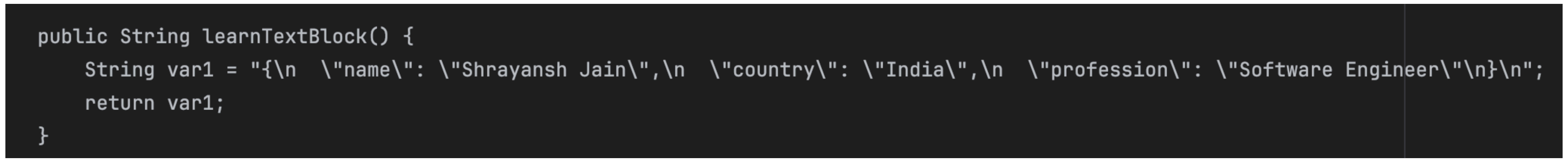
3. Trailing Whitespace:
   By default all trailing white spaces are removed, but if we want it then we have to add '\s' at the end.

Without '/s':



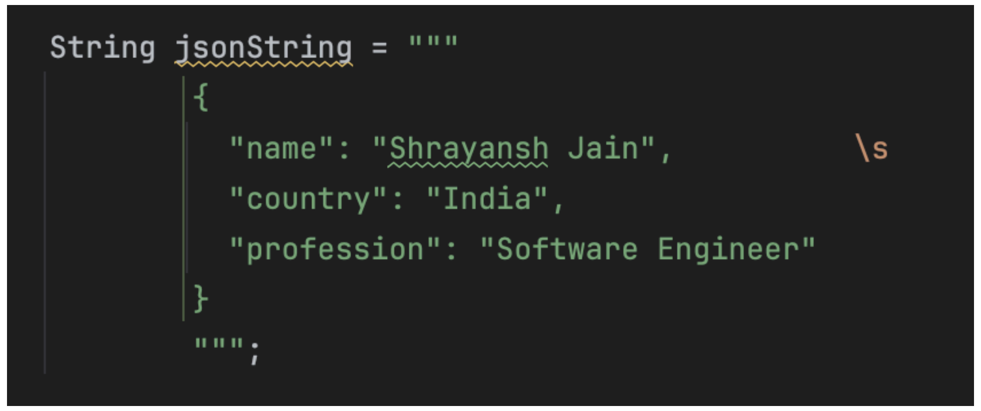
```
String jsonString = """
        {
          "name": "Shrayansh Jain",
          "country": "India",
          "profession": "Software Engineer"
        }
        """;
```

Trailing space is removed:



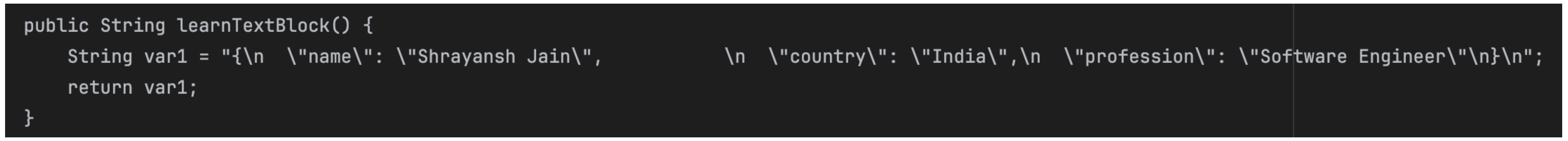
```
public String learnTextBlock() {
    String var1 = "{\n  \"name\": \"Shrayansh Jain\",\n  \"country\": \"India\",\n  \"profession\": \"Software Engineer\"\n}\n";
    return var1;
}
```

With '\s':



```
String jsonString = """
        {
          "name": "Shrayansh Jain",          \s
          "country": "India",
          "profession": "Software Engineer"
        }
        """;
```

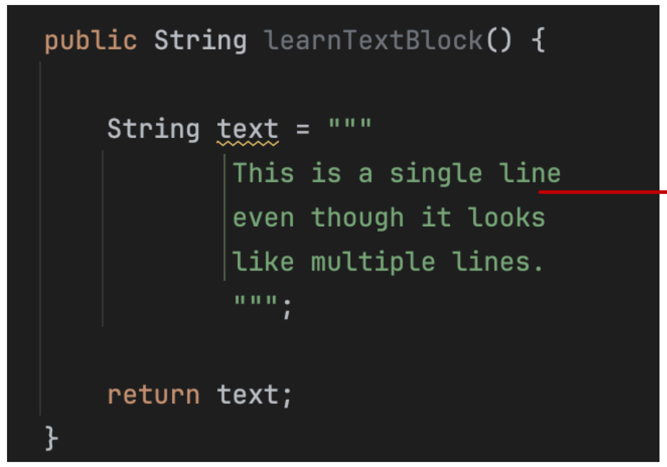
Trailing space is preserved:



```
public String learnTextBlock() {
    String var1 = "{\n  \"name\": \"Shrayansh Jain\",           \n  \"country\": \"India\",\n  \"profession\": \"Software Engineer\"\n}\n";
    return var1;
}
```

4. Continuation character:

- By default in text block, for each new line, a new line char(\n) is added in the compiled string.
- If we want the continuation, we need to use Continuation char (\) at the end of a line.

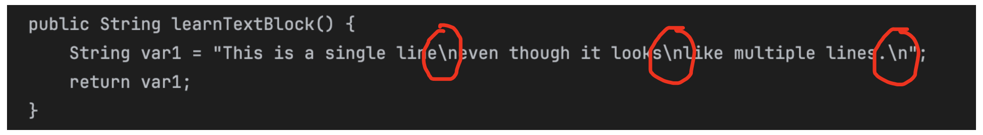


```
public String learnTextBlock() {

    String text = """
            This is a single line
            even though it looks
            like multiple lines.
            """;

    return text;
}
```

Used continuation char

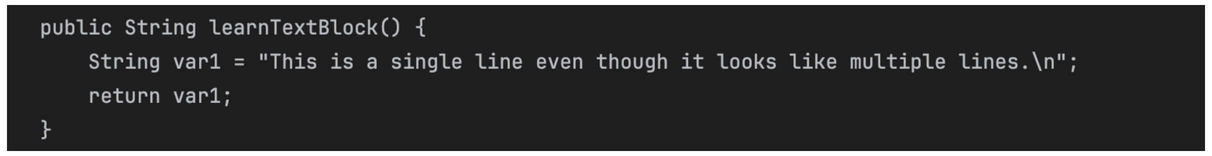
```
String text = """
   This is a single line \
   even though it looks \
   like multiple lines.
   """;
```

Complied:



```
public String learnTextBlock() {
    String var1 = "This is a single line\neven though it looks\nlike multiple lines.\n";
    return var1;
}
```

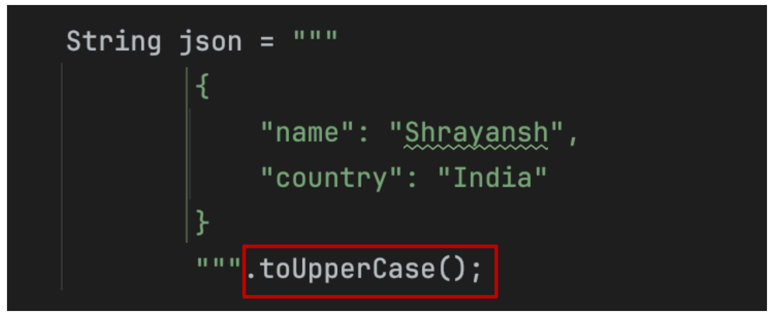
Complied:



```
public String learnTextBlock() {
    String var1 = "This is a single line even though it looks like multiple lines.\n";
    return var1;
}
```

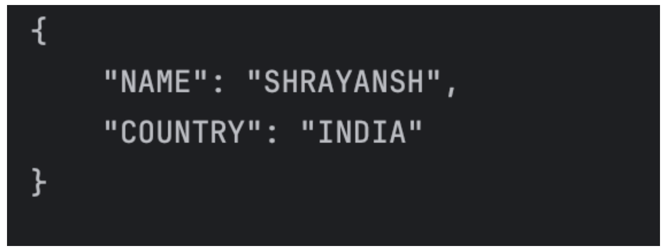
Methods we can apply on Text Blocks:

- As I mentioned earlier, it’s complied to a normal String only. So all methods applicable on String is also applicable on Text Blocks too.

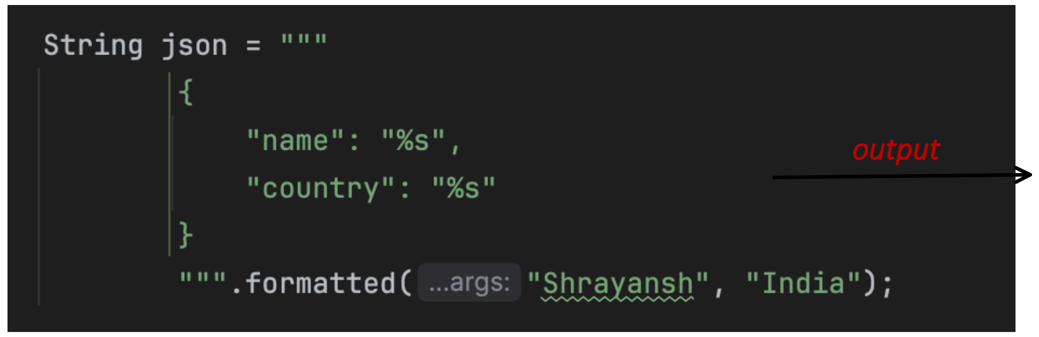


```
String json = """
        {
            "name": "Shrayansh",
            "country": "India"
        }
        """.toUpperCase();
```

output

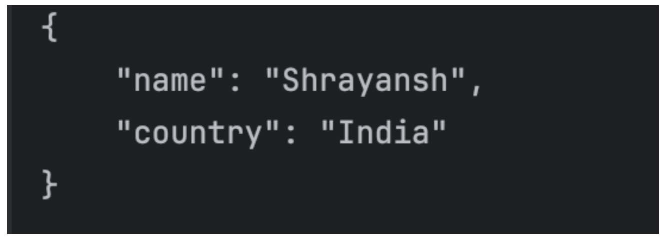


```
{
    "NAME": "SHRAYANSH",
    "COUNTRY": "INDIA"
}
```

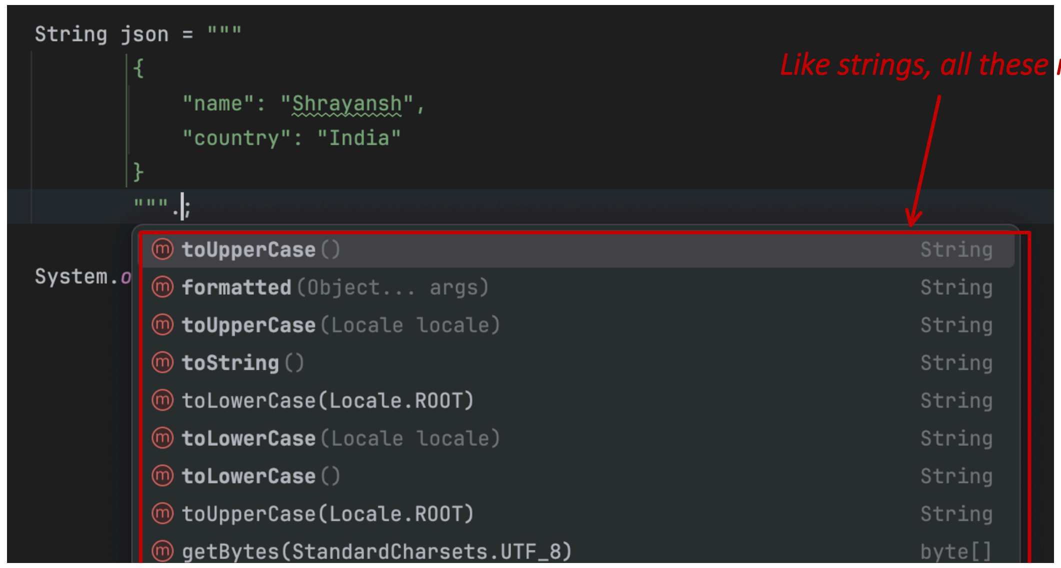


```
String json = """
        {
            "name": "%s",
            "country": "%s"
        }
        """.formatted( ...args: "Shrayansh", "India");
```

output



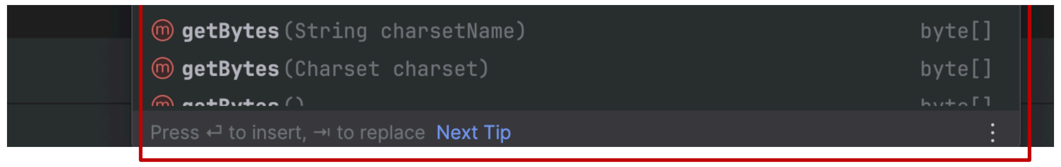
```
{
    "name": "Shrayansh",
    "country": "India"
}
```



```
String json = """
        {
            "name": "Shrayansh",
            "country": "India"
        }
        """.;

System.o
```

```
toUpperCase()                  String
formatted(Object... args)      String
toUpperCase(Locale locale)     String
toString()                     String
toLowerCase(Locale.ROOT)       String
toLowerCase(Locale locale)     String
toLowerCase()                  String
toUpperCase(Locale.ROOT)       String
getBytes(StandardCharsets.UTF_8) byte[]
```

Like strings, all these methods are available.



```
getBytes(String charsetName)        byte[]
getBytes(Charset charset)           byte[]
```