

# Java 13

## Links :-

1. <http://openjdk.java.net/projects/jdk/13/>
2. <https://cr.openjdk.java.net/~iris/se/13/latestSpec/apidiffs/overview-summary.html>  
(API differences)
3. <https://www.oracle.com/java/technologies/javase/13-relnote-issues.html#NewFeature>
- 4.

## Switch Expression :-

1. Only change is to use "yield" to return value instead of using "break" to return values from switch statements.
2. Break is deprecated and no longer in use to return a value from switch expression.

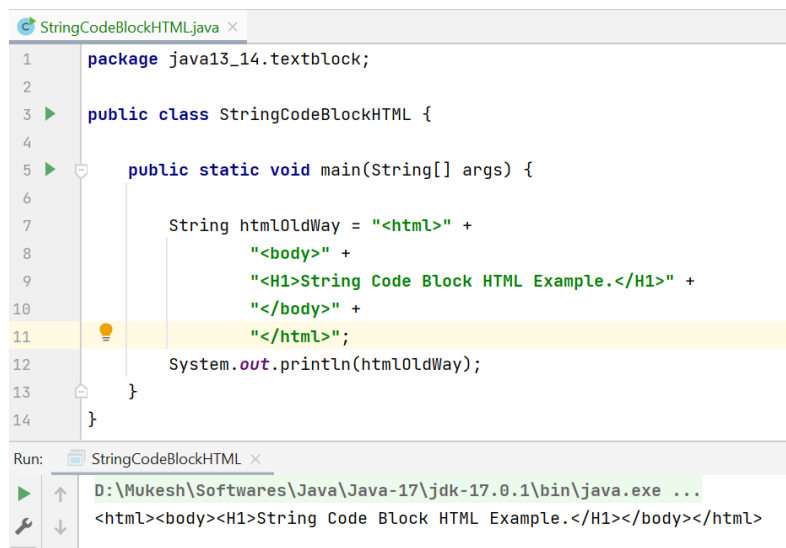
## ZGC to return Uncommitted Unused memory :-

1. Similar to G1 GC functionality to release uncommitted memory.

## Text Blocks (String Content Block):-

1. Provided support for other languages by giving their engines, Like JS, html etc.
2. This feature is a preview feature and still under maintenance.
3. Some time we need to write some other code of some other languages like HTML.

All HTML code is part of a string. For ex:-



```
StringCodeBlockHTML.java x
1 package java13_14.textblock;
2
3 public class StringCodeBlockHTML {
4
5     public static void main(String[] args) {
6
7         String htmlOldWay = "<html>" +
8             "<body>" +
9             "<H1>String Code Block HTML Example.</H1>" +
10            "</body>" +
11            "</html>";
12        System.out.println(htmlOldWay);
13    }
14 }
```

Run: StringCodeBlockHTML x

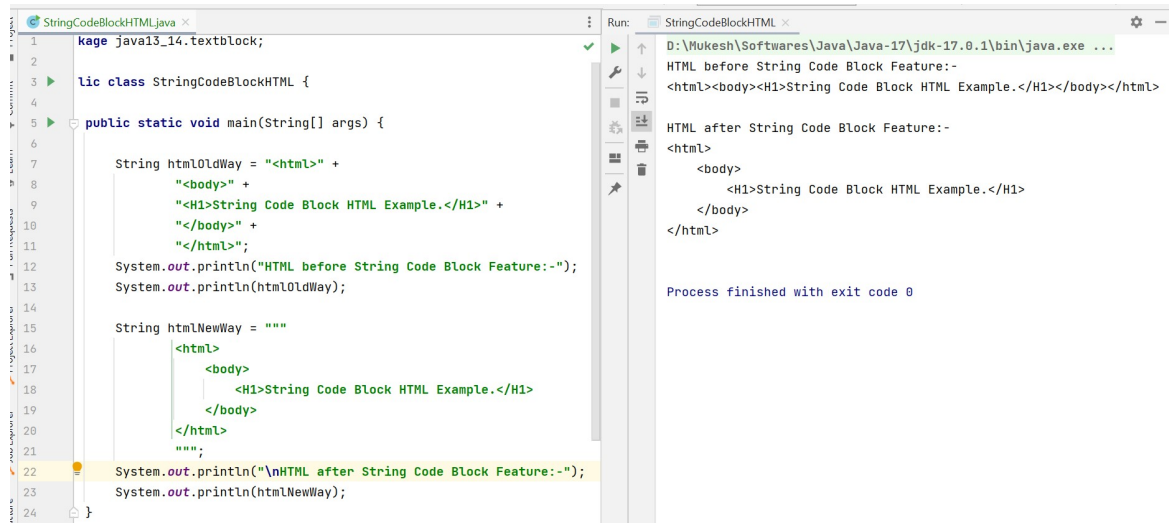
D:\Mukesh\Softwares\Java\Java-17\jdk-17.0.1\bin\java.exe ...

<html><body><H1>String Code Block HTML Example.</H1></body></html>

4. As we can see in the attached screenshot:-

- Code readability/quality is very low because of that level string concatenation.
- Result is also not easily readable and take extra effort to understand it because it is in one line.
- We can manage the this by placing `\n` and manual indentation. But it will reduce the code readability.

5. But Text Block is a feature to overcome by this problem. It provided triple quotes (Like Kotlin) to handle all such use cases. Lets have an example:-



```
StringCodeBlockHTML.java
1 package java13_14.textblock;
2
3 public class StringCodeBlockHTML {
4
5     public static void main(String[] args) {
6
7         String htmlOldWay = "<html>" +
8             "<body>" +
9             "<H1>String Code Block HTML Example.</H1>" +
10            "</body>" +
11            "</html>";
12
13        System.out.println("HTML before String Code Block Feature:-");
14        System.out.println(htmlOldWay);
15
16        String htmlNewWay = """
17            <html>
18            <body>
19            <H1>String Code Block HTML Example.</H1>
20            </body>
21            </html>
22            """;
23
24        System.out.println("\nHTML after String Code Block Feature:-");
25        System.out.println(htmlNewWay);
26    }
27 }
```

Run: StringCodeBlockHTML

D:\Mukesh\Softwares\Java\Java-17\jdk-17.0.1\bin\java.exe ...

HTML before String Code Block Feature:-

```
<html><body><H1>String Code Block HTML Example.</H1></body></html>
```

HTML after String Code Block Feature:-

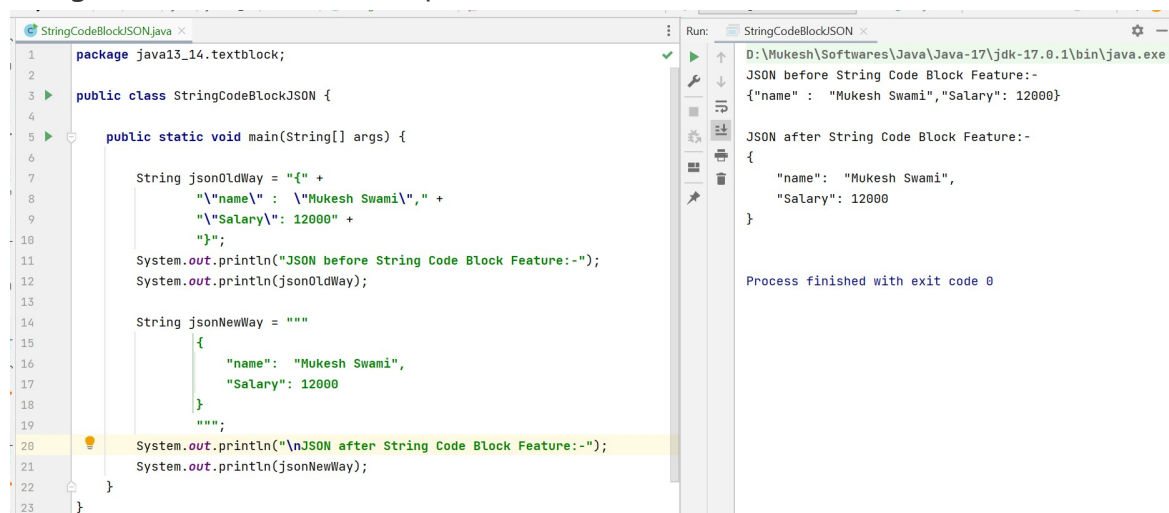
```
<html>
<body>
  <H1>String Code Block HTML Example.</H1>
</body>
</html>
```

Process finished with exit code 0

6. By using this feature:-

- We do not need to manually concatenate the Strings.
- It automatically manages the spaces and indentation.
- Output will remain same as we provides the input.
- It improves the readability.

7. String Code Block JSON example:-



```
StringCodeBlockJSON.java
1 package java13_14.textblock;
2
3 public class StringCodeBlockJSON {
4
5     public static void main(String[] args) {
6
7         String jsonOldWay = "{" +
8             "\"name\" : \"Mukesh Swami\"," +
9             "\"Salary\" : 12000" +
10            "}";
11
12        System.out.println("JSON before String Code Block Feature:-");
13        System.out.println(jsonOldWay);
14
15        String jsonNewWay = """
16            {
17            "name": "Mukesh Swami",
18            "Salary": 12000
19            }
20            """;
21
22        System.out.println("\nJSON after String Code Block Feature:-");
23        System.out.println(jsonNewWay);
24    }
25 }
```

Run: StringCodeBlockJSON

D:\Mukesh\Softwares\Java\Java-17\jdk-17.0.1\bin\java.exe

JSON before String Code Block Feature:-

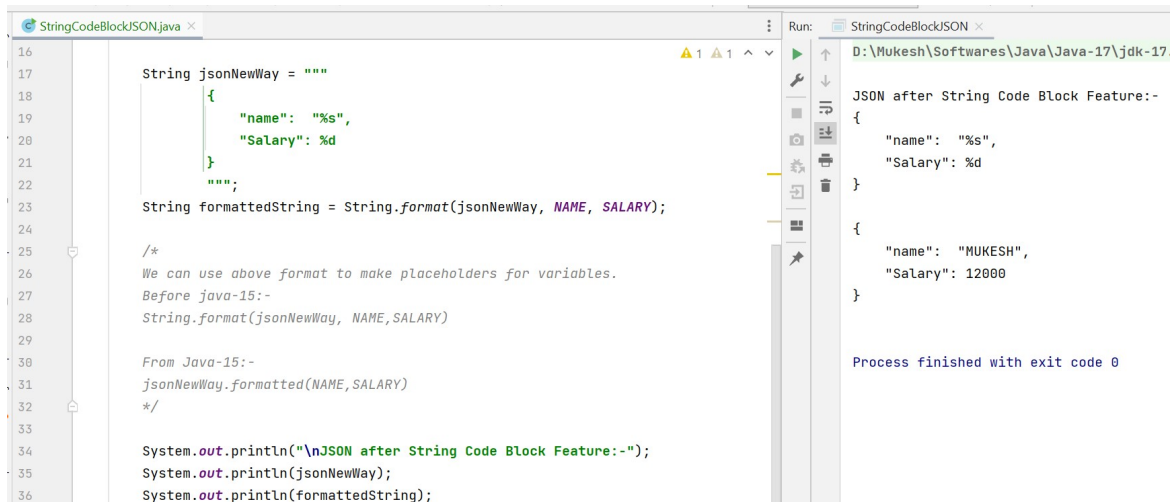
```
{"name" : "Mukesh Swami","Salary": 12000}
```

JSON after String Code Block Feature:-

```
{
  "name": "Mukesh Swami",
  "Salary": 12000
}
```

Process finished with exit code 0

8. There is a huge difference between both the output strings even though input is same and readability is a very big concern in old way.
9. We can add placeholders for variables as well:-



The screenshot shows an IDE with a Java file named `StringCodeBlockJSON.java`. The code defines a JSON string with placeholders for variables and demonstrates two ways to format it. The output window shows the result of the formatting, which is a readable JSON object.

```
String jsonNewWay = ""
{
    "name": "%s",
    "Salary": %d
}
"";

String formattedString = String.format(jsonNewWay, NAME, SALARY);

/*
We can use above format to make placeholders for variables.
Before java-15:-
String.format(jsonNewWay, NAME, SALARY)

From Java-15:-
jsonNewWay.formatted(NAME, SALARY)
*/

System.out.println("\nJSON after String Code Block Feature:-");
System.out.println(jsonNewWay);
System.out.println(formattedString);
```

Run: StringCodeBlockJSON

D:\Hukesh\Softwares\Java\Java-17\jdk-17

JSON after String Code Block Feature:-

```
{
  "name": "%s",
  "Salary": %d
}
```

Process finished with exit code 0

10. For now, we have to use `String.format()` method to fill the placeholders.
11. But from java-15, we can use `string.formatted()` method to do so. (Only applicable on Text Block not on normal String)