



Back to Basics: Strings in Java

Michael Krimgen



What Is a String?

Sequence of characters in a defined order.

| | | | | | | | | | | | | | | | | | | |
|---|---|--|---|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|
| N | O | | S | T | R | I | N | G | S | | A | T | T | A | C | H | E | D |
|---|---|--|---|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|

Motivation

The image is a collage of three main elements. On the left is a presentation slide with a teal background. It features a cartoon character resembling a teabag with a face, holding a small cup and saucer. The text on the slide reads "HISTORICAL ODDITIES & PERSISTENT ITCHES" in large white letters, followed by "MOSTLY SKIMMING THE SURFACE, RAMBLING AND AVOIDING THE JVM" in smaller white letters. At the bottom, there are three small text blocks: "AUGUST 2024" on the left, "James Gosling" in the center, and "JVMLS" on the right. The middle section of the collage shows a photograph of James Gosling, an older man with a white beard and glasses, wearing a brown t-shirt with a quote and a portrait of Steve Jobs. The right section of the collage features the "ORACLE" logo in gold letters.

AUGUST 2024

James Gosling

JVMLS

HISTORICAL ODDITIES
& PERSISTENT ITCHES

MOSTLY SKIMMING THE SURFACE, RAMBLING AND AVOIDING THE JVM

JVM Language Summit 2024

ORACLE



Name

Old English: line, thick thread, cord

edge of the upper-deck ports.—12. In *printing*, a piece-compositor's aggregate of the proofs of types set by him, pasted on a long strip of paper. The amount of work done is determined by the measurement of this string.—13.



Strings Are Special

```
1 var message = "Hello " + "Java".toUpperCase();
```

- Literals
- Can call method on literal
- Operator overload
- One of most used classes
- String pool



Immutability

```
1 void main() {  
2     String language = "Java";  
3     language.toUpperCase();  
4     System.out.println(language);  
5 }
```



Immutability

- ❑ Security
- ❑ Synchronization
- ❑ Class Loading
- ❑ String Pool
- ❑ Cashing the hash code

Immutability

```
1  public int hashCode() {  
2      int h = hash;  
3      if (h == 0 && !hashIsZero) {  
4          h = isLatin1() ? StringLatin1.hashCode(value)  
5                          : StringUTF16.hashCode(value);  
6          if (h == 0) {  
7              hashIsZero = true;  
8          } else {  
9              hash = h;  
10         }  
11     }  
12     return h;  
13 }
```

Java <= 6

```
1 public final class String
2     implements java.io.Serializable, Comparable<String>, CharSequence
3 {
4     /** The value is used for character storage. */
5     private final char value[];
6
7     /** The offset is the first index of the storage that is used. */
8     private final int offset;
9
10    /** The count is the number of characters in the String. */
11    private final int count;
12
13    /** Cache the hash code for the string */
14    private int hash; // Default to 0
15
16    //...
17 }
```



Java 7/8

```
1 public final class String
2     implements java.io.Serializable, Comparable<String>, CharSequence {
3     /** The value is used for character storage. */
4     private final char value[];
5
6     /** Cache the hash code for the string */
7     private int hash; // Default to 0
8
9     // ...
10 }
```



Java >9

```
1 public final class String // implements ...
2 {
3     @Stable
4     private final byte[] value;
5
6     private final byte coder;
7
8     private int hash;
9
10    private boolean hashIsZero;
11
12    static final boolean COMPACT_STRINGS;
13
14    // ...
15 }
```

Implementation

```
1 public int length() {  
2     return value.length >> coder();  
3 }
```

```
1 public char charAt(int index) {  
2     if (isLatin1()) {  
3         return StringLatin1.charAt(value, index);  
4     } else {  
5         return StringUTF16.charAt(value, index);  
6     }  
7 }
```

Encoding

```
1 void main() {  
2     var strings = List.of(  
3         "String",  
4         "我喜欢茶",  
5         "abc",  
6         "用人❤️咖啡"  
7     );  
8  
9     strings.forEach(s -> println(s + " -> " + s.length()));  
10 }
```

Encoding

1 String -> 6

2 我喜欢茶 -> 4

3 abc -> 6

4 🧑‍🤝‍🧑 -> 4

Encoding

```
1 var strings = List.of(  
2     "String",  
3     "我喜欢茶",  
4     "abc",  
5     "👱‍♀️❤️☕"  
6 );  
7  
8 strings.forEach(s ->  
9     println(s + " -> " + s.codePointCount(0, s.length())))  
10 );
```

J a V a

Encoding

1 String -> 6

2 我喜欢茶 -> 4

3 abc -> 3

4 🧑‍🤝‍🧑 -> 3



J a V a

Encoding

```
1 static void printReverse(String string) {  
2     for(int i = string.length() - 1; i ≥ 0; i--) {  
3         System.out.print(string.charAt(i));  
4     }  
5     System.out.print("\n");  
6 }
```



Encoding

- 1 gnirts
- 2 茶欢喜我
- 3 ?ba?
- 4 ☕ ❤ ??

Encoding

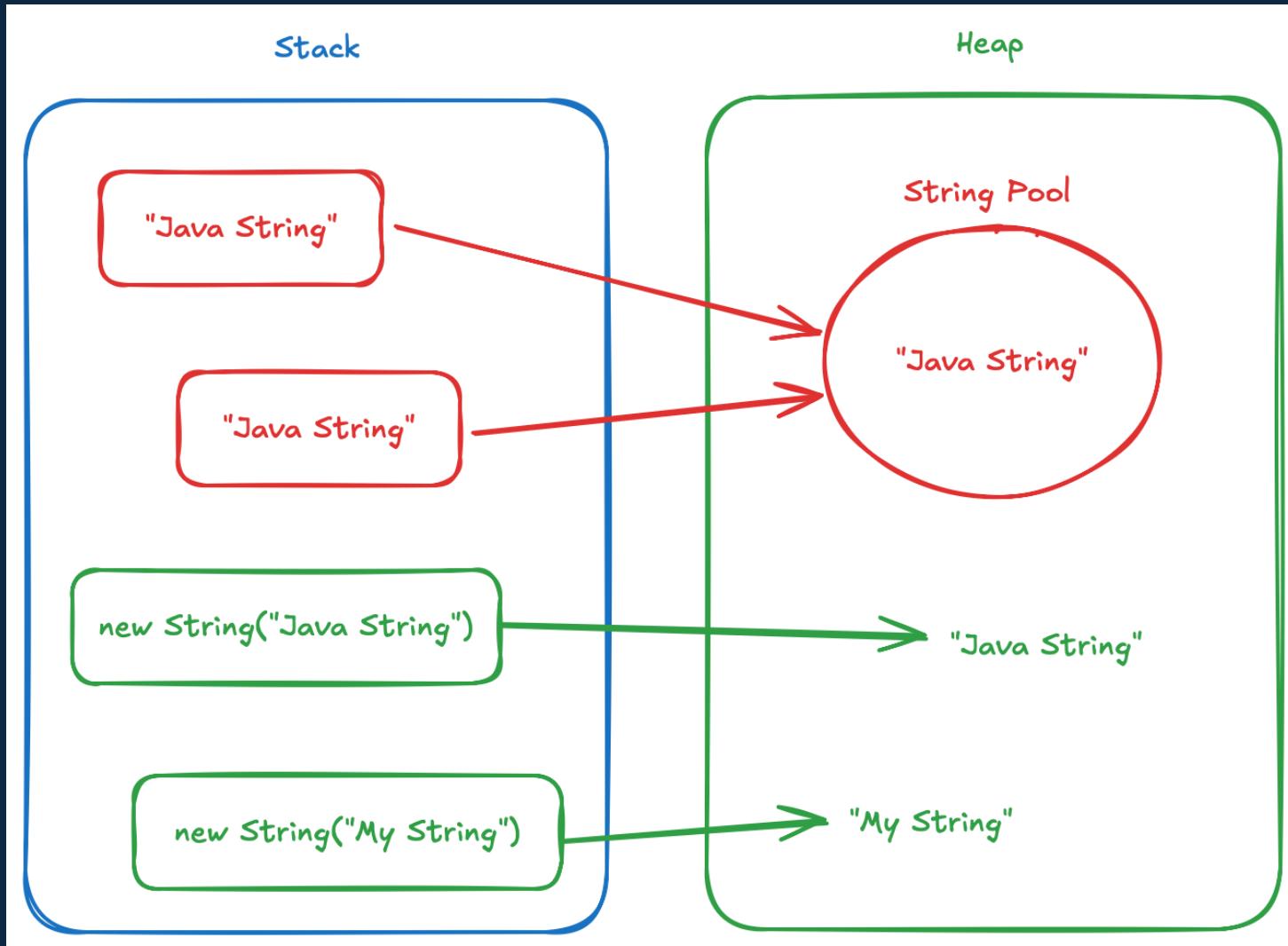
| | | | |
|--------------|-------------------|-------------------|------------------------|
| Emoji | | | |
| Codepunten | U+1F9D7 | U+200D | U+2640 U+FEOF |
| Unicode-naam | Klimmende Persoon | Zero Width Joiner | Vrouwelijkheidssymbool |



<https://emojipedia.org/nl/klimmende-vrouw#technical>

<https://www.outsideonline.com/outdoor-adventure/climbing/introducing-first-female-climbing-emoji/>

String Pool



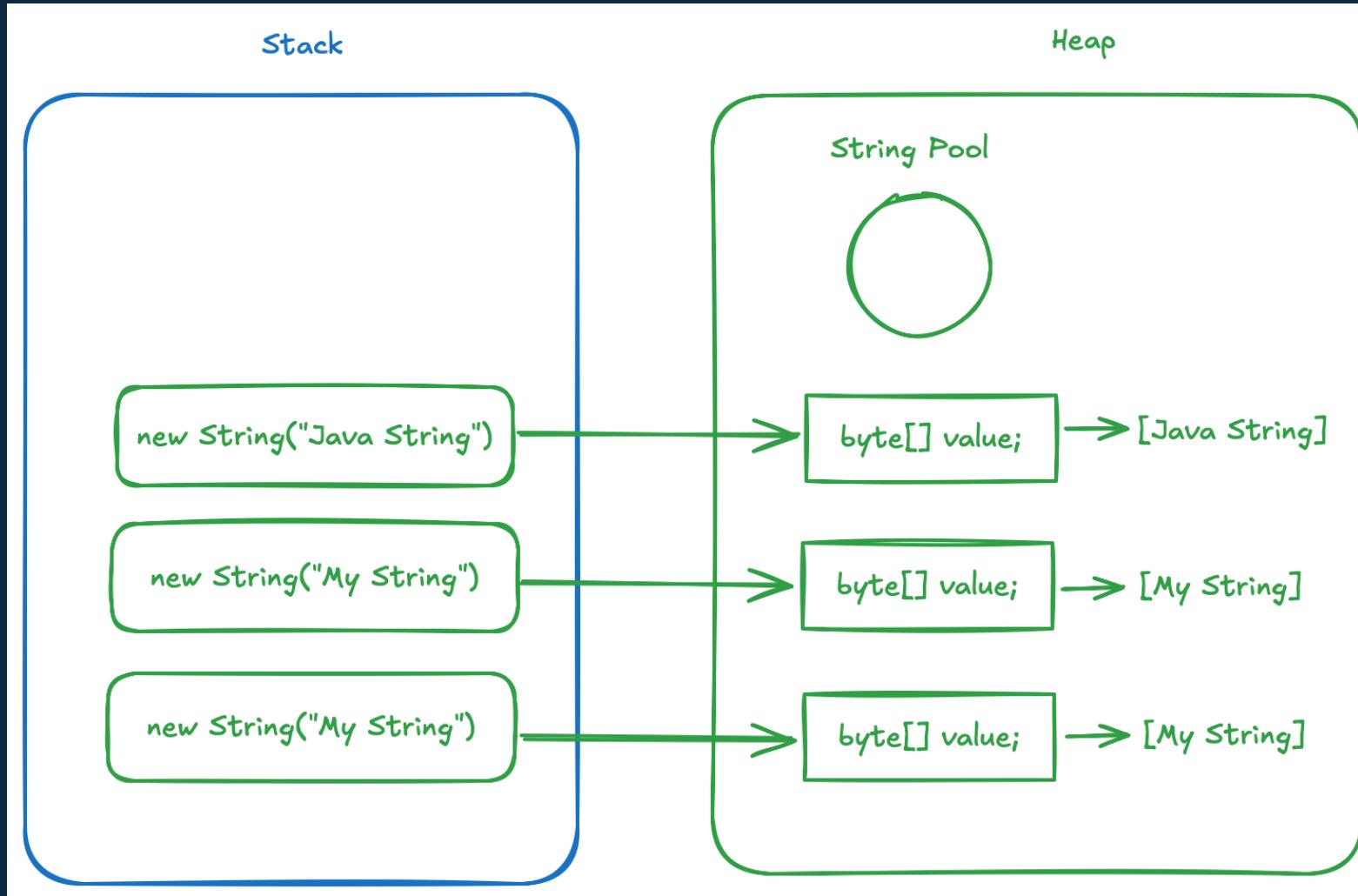


Sensitive Data

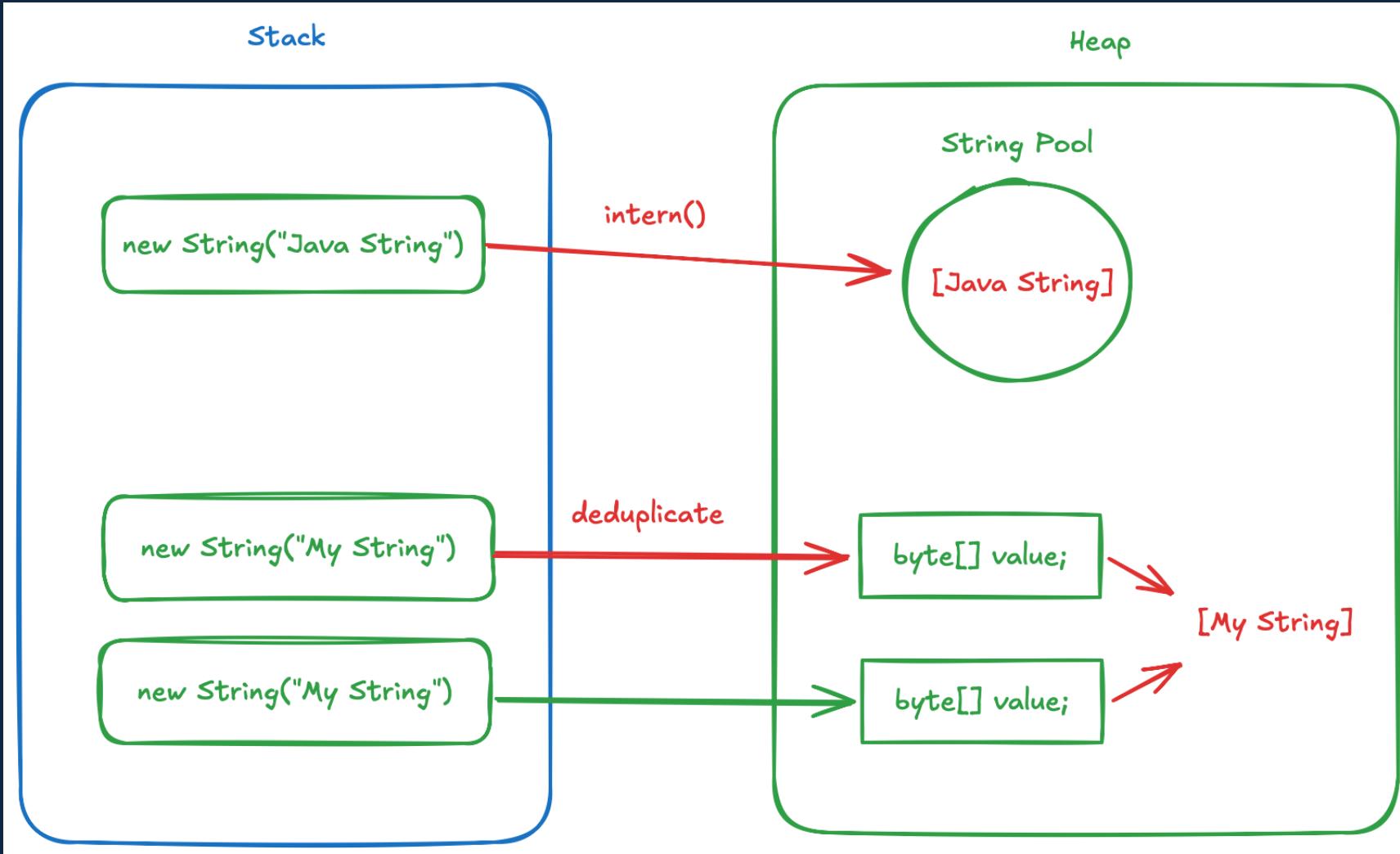
```
1 String password = "password";
```

```
1 char[] password = {'p', 'a', 's', 's', 'w', 'o', 'r', 'd'};  
2 Arrays.fill(password, '0');
```

String Deduplication



String Deduplication





String Deduplication

- ❑ First released in Java 8u20 for the G1 garbage collector
- ❑ Java 18 for ZGC, SerialGC, and ParallelGC
- ❑ Disabled by default, can be explicitly enabled:
`XX:+UseStringDeduplication`



ChatGPT

Tell me something about strings in Java that most people do not know.



ChatGPT

One lesser-known aspect of strings in Java is that **strings are immutable**, but there's a specific performance consideration that most people don't realize: **String concatenation using the + operator creates new string objects repeatedly.**



ChatGPT

While this may seem intuitive at first, it leads to inefficiencies because Java creates a new String object every time you concatenate. For example, doing `str = str + "Hello"` repeatedly in a loop creates a new string every time, and this can result in memory overhead and slower performance.

Here's the trick most people don't know: Use `StringBuilder` or `StringBuffer` for efficient string concatenation in loops or when building large strings.

J a V a

String Concatenation



```
void main() {
    println("OpenValue"
        + (char) 10084
        + Object.class.getName()
        .split("\\.")[0]);
}
```

Java 8 Version

```
1 public class OpenValue {  
2     public static void main(String[ ] args) {  
3         System.out.println(  
4             "OpenValue" +  
5             (char) 10084 +  
6             Object.class.getName( ).split("\\\\.")[0]  
7         );  
8     }  
9 }
```

Java 8 Bytecode

```
1 public static main([Ljava/lang/String;)V
2   L0
3     LINENUMBER 4 L0
4     GETSTATIC java/lang/System.out : Ljava/io/PrintStream;
5     NEW java/lang/StringBuilder
6     DUP
7     INVOKESPECIAL java/lang/StringBuilder.<init> ()V
8     LDC "OpenValue\u2764"
9     INVOKEVIRTUAL java/lang/StringBuilder.append (Ljava/lang/String;)Ljava/lang/StringBuilder;
10    LDC Ljava/lang/Object;.class
11    L1
12    LINENUMBER 7 L1
13    INVOKEVIRTUAL java/lang/Class.getName ()Ljava/lang/String;
14    LDC "\\."
15    INVOKEVIRTUAL java/lang/String.split (Ljava/lang/String;)[Ljava/lang/String;
16    ICONST_0
17    AALOAD
18    INVOKEVIRTUAL java/lang/StringBuilder.append (Ljava/lang/String;)Ljava/lang/StringBuilder;
19    INVOKEVIRTUAL java/lang/StringBuilder.toString ()Ljava/lang/String;
```



Java 23 Bytecode (Java 9+)

```
1 main()V
2   L0
3     LINENUMBER 2 L0
4     LDC Ljava/lang/Object;.class
5   L1
6     LINENUMBER 5 L1
7     INVOKEVIRTUAL java/lang/Class.getName ()Ljava/lang/String;
8     LDC "\\."
9     INVOKEVIRTUAL java/lang/String.split (Ljava/lang/String;)[Ljava/lang/String;
10    ICONST_0
11    AALOAD
12    INVOKEDYNAMIC makeConcatWithConstants(Ljava/lang/String;)Ljava/lang/String; [
13      // handle kind 0x6 : INVOKESTATIC
14      java/lang/.invoke/StringConcatFactory.makeConcatWithConstants( // ..
15      // arguments:
16      "OpenValue\u2764\u0001"
17  ]
```



Java 23 - With Loop

```
1 void main() {  
2     String result = "";  
3     for(int i = 1; i < 1000; i++) {  
4         result += "x";  
5     }  
6     System.out.println(result);  
7 }
```

Java 23 Bytecode

```
1 L4
2 LINENUMBER 4 L4
3 ALOAD 1
4 INVOKEDYNAMIC makeConcatWithConstants(Ljava/lang/String;)Ljava/lang/String; [
5     // handle kind 0x6 : INVOKESTATIC
6     java/lang/invoke/StringConcatFactory.makeConcatWithConstants( // ..
7     // arguments:
8     "\u0001x"
9 ]
10 ASTORE 1
11 L5
12 LINENUMBER 3 L5
13 IINC 2 1
14 GOTO L2
```



Java 23+

- ❑ JEP 280: Indify String Concatenation
- ❑ JDK-8327247
- ❑ JDK 23:
 - Djava.lang.invoke.stringConcat.highArityThreshold=xx

String Concatenation Overview

```
1 void main() {  
2     String greeting = "Hello", name = "Java";  
3  
4     var withPlus          = greeting + " " + name;  
5     var withConcat         = greeting.concat(" ").concat(name);  
6     var withStringBuffer   = new StringBuffer(greeting).append(" ").append(name);  
7     var withStringBuilder  = new StringBuilder(greeting).append(" ").append(name);  
8     var withFormatted      = "%s %s".formatted(greeting, name);  
9     var withJoin           = String.join(" ", greeting, name);  
10    var withStringJoiner  = new StringJoiner(" ").add(greeting).add(name);  
11    var withStream          = Stream.of(greeting, name).collect(Collectors.joining(" "));  
12 }
```



String Updates - Java 8

```
1 // Java 8
2 String option = new Scanner(System.in).nextLine();
3
4 switch (option) {
5     case "1": println("Do the first possible thing"); break;
6     case "2": println("Do the second thing"); break;
7     default: println("Do something else");
8 }
```

String Updates - Java 11

```
1 // Java 11
2 String name = " Java ";
3
4 name = name.strip();
5 name = name.stripTrailing();
6 name = name.stripLeading();
7 name = name.repeat(3);
8 boolean test = name.isBlank();
```



String Updates - Java 12

```
1 // Java 12
2 String language = "Java";
3 language = language.indent(3);
4 String transformed = name.transform(s → s.strip().toUpperCase());
```



String Updates - Java 15

```
1 // Java 15
2 String sql = """
3     SELECT l.name, l.creator, l.release, v.version
4     FROM language l
5     LEFT JOIN version v on v.l_id = l.id
6     WHERE l.name = 'java'
7     AND v.version > 5
8 """;
```



Future Development

- String Templates
- Project Valhalla

String Templates

```
1 void html() {  
2     var title = "String Templates in Java 21";  
3     var body = "A new cool feature";  
4     var html = """  
5         <html>  
6             <head><title>${title}</title></head>  
7             <body>${body}</body>  
8         </html>"";  
9 }
```

String Templates

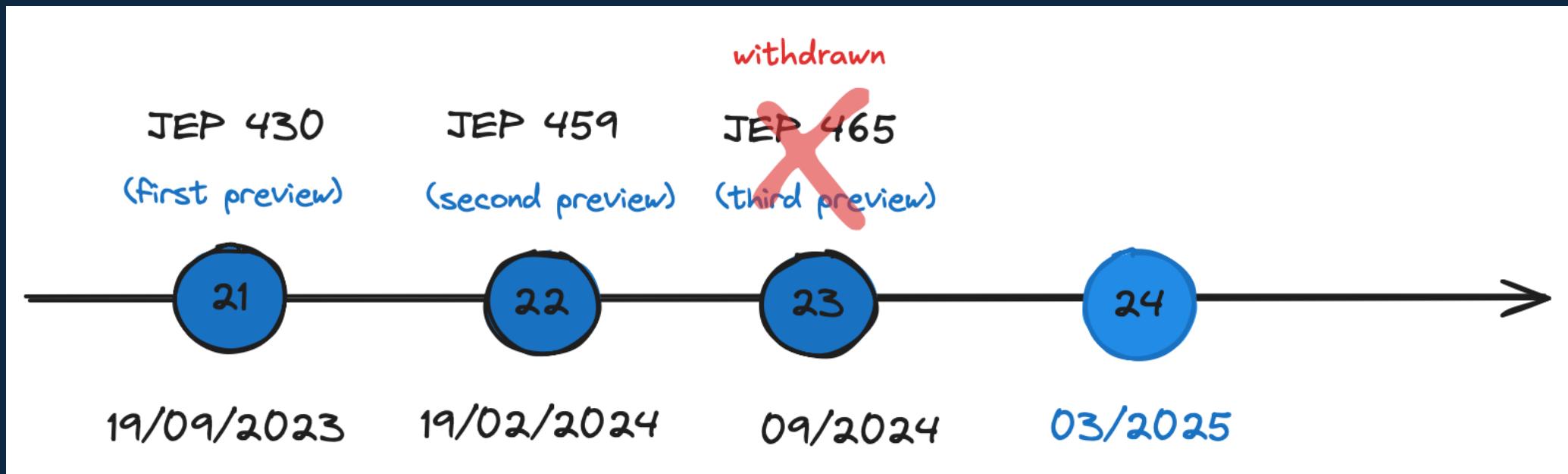
```
1 // Kotlin  
2 "$x plus $y equals ${x + y}"
```

```
1 # Python  
2 f"{x} plus {y} equals {x + y}"
```

String Templates

```
1 void hello(String name) {  
2     String message = STR."Hello\{name}";  
3     System.out.println(message);  
4 }
```

String Templates





String Templates

Unfortunately, most of the feedback we received on public channels was of the "someone didn't like the syntax" variety (...).

But these flaws had little to do with "someone didn't like the syntax"; they were much more fundamental to the model.

- Brian Goetz -



Project Valhalla

“In the short term, String is not on the list of
classes that will be migrated to a value type”



Project Valhalla

“This would be an expensive implementation change (...),
but would be possible, and might eventually be considered.”

- Brian Goetz -

J a V a

Slides & References

