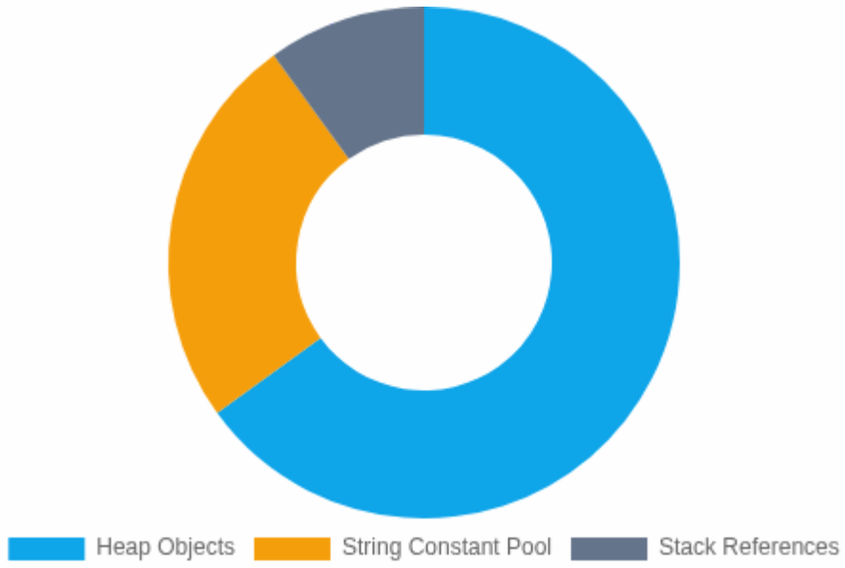
String Mastery

Visualizing the JVM String Constant Pool, Memory Efficiency, and Performance Secrets.

The Memory Split

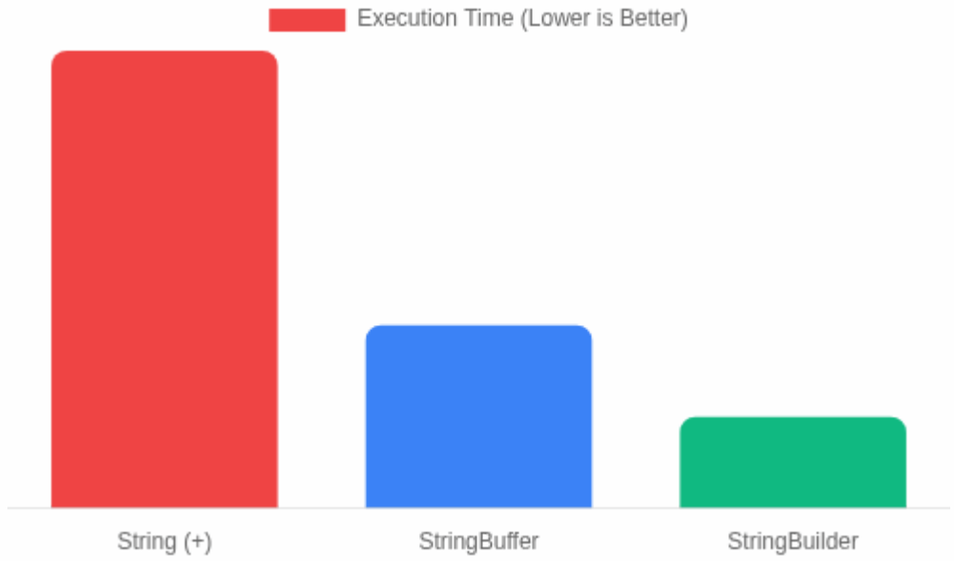
Visualizing where Strings actually live. Note that since Java 7, the **SCP** is a specialized region inside the **Heap**.




```
String s1 = "Java"; // SCP
String s2 = new String("Java"); // Heap
boolean check = (s1 == s2); // false
```

Concatenation Speed

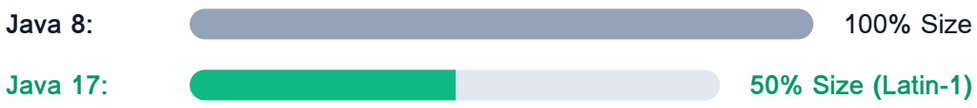
Comparing the efficiency of appending 10,000+ strings. **StringBuilder** wins for CPU, while **String** costs O(n²).



STRING: SLOW | BUILDER: FAST | BUFFER: SAFE

Compact Strings Evolution

Before Java 9, every String used 2 bytes per character (UTF-16). **Compact Strings** identify Latin-1 characters and drop storage to 1 byte.




```
// Internal Change
private final byte[] value;
private final byte coder; // LATIN1 or UTF16
```

The `intern()` Reference Rule

- In Java 7+, if a string is not in the pool, `intern()` stores the **reference** to the heap object.
- This makes `s.intern() == s` possible for dynamic strings.

```
String s = new
StringBuilder("Ja").append("va").toString();
boolean firstTime = (s.intern() == s); // true
```