Die Klasse String in Java

1)
Siehe:
https://www.qmethods.de/blog-antipattern-java-performance-code-rules-string-concatenation.html

Die Zeichenkette einer Instanz der Klasse String ist nach der Initialisierung nicht mehr veränderlich.

Beispiel1:
```
String str1 = "abc";
String str2 = new String("def");
```

Beispiel2:
```
String str="rst";
str = str + "xyz";
```

Man könnte jetzt Folgendes denken:
In Beispiel 2 wird doch die Zeichenfolge verändert.
Zuerst zeigt str auf "rst"; dann auf "rstxyz"
Also wurde doch die Zeichenfolge von "rst"zu "rstxyz" abgeändert.

Dies ist aber falsch:
Nach Ausführung des Plus-Operators wird im Arbeitsspeicher eine neue Variable (mit dem gleichen Namen) str angelegt, die auf die (unveränderliche) neu erzeugte Zeichenfolge "rstxyz" zeigt. Auf die alte Variable str kann nicht mehr zugegriffen werden (Speicherleiche) und wird deshalb zur Löschung durch den Garbage Collector der JVM (Java Virtual Machine) freigegeben (darum muss sich der Programmierer nicht kümmern).

Wird z.B.
```
str = str + "xyz";
```
in einer Schleife sehr oft aufgerufen, so wird kurzzeitig eine große Menge an String-Instanzen erzeugt und zeitnah wieder zum Löschen freigegeben. Dieses führt zu einer starken Beanspruchung des Java-Speichers sowie Performanceeinbußen durch die Speicherbereinigungen des Garbage Collectors.

2)
Die Methode splits
https://www.heise.de/tipps-tricks/Java-Einen-String-aufteilen-4060739.html

Ein regulärer Ausdruck ist eine Kombination von Zeichen mit sogenannten Metazeichen
Die Metazeichen sind:
```
< ( [ { \ ^ - = $ ! | ] } ) ? * + . >
```

Beispiel:
Will man die folgenden Zeichenfolge an den Punkten (das sind metazeichen) splitten, geht dies nicht durch die Angabe `".", sondern wie folgt:`

```
String mystring = "www.web.de";
// String[] splittArray = mystring.split(".");  <-- falsch !!!!
String[] splittArray = mystring.split("\\.");
```