

Details zu Strings

Codebeispiel - Was wird ausgegeben?

```
public class StringTest {  
    private String s1 = "Hallo";  
    private String s2 = "Hallo";  
    private String s3 = new String("Hallo");  
  
    public void teste() {  
        System.out.println(s1.equals(s2));  
        System.out.println(s1.equals(s3));  
  
        System.out.println(s1==s2);  
        System.out.println(s1==s3);  
    }  
}
```

- Die Ausdrücke liefern folgende Ergebnisse:

```
s1.equals(s2)  true
s1.equals(s3)  true
s1==s2         true
s1==s3         false
```

- Zur Erinnerung: `equals` testet auf inhaltliche Gleichheit, `==` vergleicht die Referenzen
 - Ist die Ausgabe damit zu erklären?
 - Wieso gilt `s1 == s2`?

Aufgabe 1

- Erstelle ein neues BlueJ-Projekt und implementiere die Klasse `StringTest` von der ersten Folie
- Lasse die Klasse übersetzen und rufe die Methode `teste` auf
- Öffne die Datei `StringTest.class` in einem Hexeditor
 - Gibt es auch online, z. B. unter <https://hexed.it>
- Prüfe, ob und wie oft Du in der Datei den String "Hallo" findest.
- Zusatzaufgabe:
 - Betrachte die ersten 4 Bytes der Datei ;-)

Analyse (1)

- In der Datei `StringTest.class` befindet sich der compilierte Java-Bytecode der Klasse `StringTest`
- Die Zeichenfolge "Hallo" taucht nur ein Mal in der Datei auf, obwohl sie an drei Stellen in der Klasse `StringTest` steht
 - Was ist da los?
- Alle **Stringliterale** (Zeichenfolgen, die im Quellcode zwischen " " stehen und somit zur Compilezeit bekannt sind), werden im sogenannten **constant pool** der class-Datei abgelegt
 - Der Java-Compiler erkennt identische Strings und legt sie nur ein Mal ab
- Beim Ausführen des Programms wird jeder dieser Strings also auch nur ein Mal in den Speicher geladen
 - Daher verweisen die Referenzen `s1` und `s2` im Beispielprogramm auf die gleiche Stelle im Speicher

- Wird ein String-Objekt hingegen mittels **new** erzeugt, so wird **zur Laufzeit** Speicher auf dem Heap reserviert und der String darauf abgelegt
 - Daher verweisen die Referenzen **s1** und **s3** im Beispielprogramm auf unterschiedliche Stellen im Speicher