7. Strings

Datentyp String

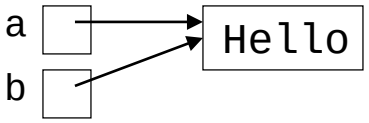
```
String a, b;
```

Bibliothekstyp für Zeichenketten ("Strings")

```
a = "Hello";
```

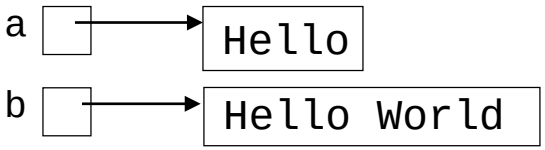
Stringkonstante (unter doppelten Hochkommas; leerer String: "")

```
b = a;
```



Stringvariablen sind Zeiger auf Stringobjekte
Stringzuweisung ist eine Zeigerzuweisung
Stringobjekte sind nicht als Arrays ansprechbar ~~a[i]~~ a.charAt(i)
Stringobjekte sind nicht veränderbar

```
b = a + " World";
```



Verkettung mit "+" erzeugt neues Stringobjekt (relativ teure Operation)

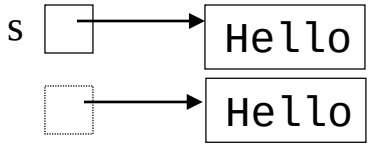
Verwendung z.B. in Out.println("Summe = " + sum);

automatisch in *String* umgewandelt und verkettet

Stringvergleiche

```
String s = In.readWord();
```
// liest ein Wort, z.B. *Hello*

```
if (s == "Hello") ...
```
// liefert false! (*Zeigervergleich*)

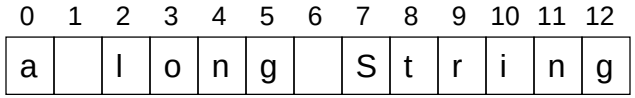


weil zwei verschiedene Objekte, trotz gleichem Inhalt

```
if (s.equals("Hello")) ...
```
// liefert *true*! (*Wertvergleich*)

=> **Stringvergleich immer mit *equals* durchführen!**

Stringoperationen




```
String s = "a long string";
```

```
int len = s.length();
```

liefert die Anzahl der Zeichen in *s*
(im Gegensatz zu array.length sind Klammern nötig!)

```
char ch = s.charAt(3);
```

liefert das Zeichen mit Index 3 (hier 'o')
s[3] nicht erlaubt!

```
int i = s.indexOf("ng");
i = s.indexOf("ng", 5);
i = s.indexOf('n');
i = s.lastIndexOf("ng");
```

liefert Index des 1. Vorkommens von "ng" in *s* (hier 4) oder -1
liefert Index des 1. Vorkommens von "ng" ab Index 5 (hier 11)
geht auch mit *char*
liefert Index des letzten Vorkommens von "ng" (Varianten wie oben)

```
String x;
x = s.substring(2);
x = s.substring(2, 6);
```

liefert Teilstring ab Index 2 (hier "long string")
liefert Teilstring *s*[2..6[, d.h. *s*[2..5] (hier "long")

```
if (s.startsWith("abc")) ...
if (s.endsWith("abc")) ...
```

liefert *true*, falls *s* mit "abc" beginnt
liefert *true*, falls *s* mit "abc" endet

Aufbauen von Strings

aus Stringkonstante

```
String s = "very simple";
```

aus char-Array

```
char[] a = {'a', 'b', 'c', 'd', 'e'};
String s1 = new String(a);          // s1 enthält Kopie der Zeichen in a
String s2 = new String(a, 2, len);  // s2 enthält Kopie von a[2..2+len-1]
```

Strings können nach dem Aufbauen nicht mehr verändert werden
s1 + s2 erzeugt einen neuen String

Aufbauen von Strings aus StringBuilder

mit einem StringBuilder (Bibliothekstyp wie *String*, aber modifizierbar)

`StringBuilder b;` `b = new StringBuilder();`	erzeugt einen leeren *StringBuilder* der Länge 0
`len = b.length();`	liefert Anzahl der Zeichen im *StringBuilder*
`b.append(x);`	hängt *x* an *b* an. *x* kann beliebigen Typ haben: *short*, *int*, *long*, *float*, *double*, *char*, *char*[], *String*, *boolean*

`b.insert(pos, x);`
`b.delete(from, to);`
`b.replace(from, to, "abc");`
	fügt *x* an der Stelle *pos* ein (Typ von *x* beliebig)
	löscht [*from*..*to*[aus *b*
	ersetzt *b*[*from*, *to*[durch "abc"

`ch = b.charAt(i);`	wie bei *String*
`b.setCharAt(pos, 'x');`	setzt *b*[*pos*] auf 'x'
`s = b.substring(from, to);`	liefert Teilstring *b*[*from*..*to*[
`s = b.toString();`	liefert Pufferinhalt als *String*

Stringkonversionen

```
int i    = Integer.parseInt("123");
long n   = Long.parseLong("3000000000");
float f  = Float.parseFloat("3.14");
double d = Double.parseDouble("1.2E40");
```

String Þ int
String Þ long
String Þ float
String Þ double

```
String s;
s = String.valueOf(123);
s = String.valueOf(3000000000);
s = String.valueOf(3.14);
s = String.valueOf(1.2E40);
```

int Þ String
long Þ String
float Þ String
double Þ String

```
char[] a = s.toCharArray();
String s = new String(a);
```

String Þ char[]
char[] Þ String

7. Strings

Beispiel: Kommandozeilenparameter

Programmaufruf mit Parametern

java *Programmname* par_1 par_2 ... par_n

Parameter werden als *String*-Array an die *main*-Methode übergeben

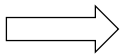
```
class Sample {
    public static void main (String[] arg) {
        for (int i = 0; i < arg.length; i++) {
            Out.println(arg[i]);
        }
    }
}
```

oder

```
for (String s: arg) {
    Out.println(s);
}
```

Aufruf

```
java Sample Anton /a 10
```



```
arg[0] == "Anton"
arg[1] == "/a"
arg[2] == "10"
```

Ausgabe

```
Anton
/a
10
```

Beispiel: Manipulation von Dateipfaden

dir1\dir2\name.java Þ name.class

- Verzeichnisse entfernen
- ".java" auf ".class" ändern (bzw. ".class" anhängen)

```
String path = ...;

StringBuilder b = new StringBuilder(path);
if (path.endsWith(".java")) {
    int len = path.length();
    b.delete(len-5, len);
}
b.append(".class");
int i = path.lastIndexOf('\\');
if (i >= 0) b.delete(0, i+1);

String result = b.toString();
```

z.B. path = "dir1\dir2\Sample.java";

erzeugt *StringBuilder* mit *path* als Inhalt

21
"dir1\dir2\Sample"

"dir1\dir2\Sample.class"
9
"Sample.class"

Beispiel: Wörter aus einem Text lösen

Eingabe: **`"Ein Text aus Woertern ..."`**

Ausgabe:

```
Ein
Text
aus
Woertern
```

```
String text = ...;

int i = 0;
int last = text.length() - 1;
while (i <= last) {
   //--- skip nonletters
   while (i <= last && !Character.isLetter(text.charAt(i))) i++;
   // end of text or text[i] is a letter
   //--- read word
   int beg = i;
   while (i <= last && Character.isLetter(text.charAt(i))) i++;
   // end of text or text[i] is not a letter
   //--- print word
   if (i > beg) Out.println(text.substring(beg, i));
}
```