Zeichenketten (Strings)

Java behandelt Zeichenketten wegen ihrer häufigen Nutzung in vielerlei Hinsicht als Objekte besonderer Art. So ist zum Beispiel die Objekterzeugung erleichtert. Die Java-API-Klasse `String` erlaubt zwei Möglichkeiten der Objekterzeugung:

- Langform mit explizitem Konstruktoraufruf (entspricht der üblichen Art und Weise der Objekterzeugung):

```
String vorname = new String(„Hans“);
```

- Kurzform mit direkter Zuweisung des Stringinhalts (ohne Konstruktoraufruf):

```
String vorname = „Hans“;
```

- Bei verzögerter Initialisierung auch in zwei getrennten Anweisungen:

```
String vorname;
. . .
vorname = “Hans“;
```

String-Objekte sind nicht veränderbar

Wenn ein String-Inhalt nach seiner Initialisierung verändert wird -

```
String vorname = "Heinz";

vorname = "Albert";
```

dann bewirkt dies etwas Unerwartetes – nämlich die Anlage eines völlig neuen Stringobjekts, das zwar die gleiche Referenz besitzt, aber in der Tat ein völlig neues Objekt ist. Das alte String-Objekt wird von der automatischen Speicherbereinigung entsorgt. Diese Tatsache kann im Allgemeinen ignoriert werden (und ist in der Tat kaum bekannt), aber in bestimmten Programmsituationen (z.B. in Einlese-Schleifen) kann die ständige Objekterzeugung und –zerstörung im Hintergrund sehr ressourcenfressend wirken und im äußersten Fall ein Programm sogar zum Absturz bringen. Für solche Fälle empfiehlt sich die Verwendung eines **StringBuffer**-Objekts, das veränderbar ist.

String-Verknüpfung mit + Operator

Mehrere String-Objekte können durch den Operator **+** verknüpft werden, der für Strings eine besondere Bedeutung hat. Dies wird zum Beispiel für formatierte Bildschirmausgaben häufig benutzt:

```
String vorname = "Hans ";
String nachname = "Mueller ";
String name;

name = vorname + nachname;

System.out.println(name);     // „Hans Mueller "
```

Formatumwandlung (Zahlen in Strings und umgekehrt)

Auch Zahlen (`int`, `double` und andere Objektdatentypen) können mit dem + Operator an Strings angehängt und ausgegeben werden. Bei der Verknüpfung findet eine automatische Konvertierung in eine darstellbare Zeichenkette statt. Dies wird für Zahlenausgaben häufig benutzt.

```
int var1 = 47, var2 = 12;

System.out.println("Die Zahlen sind: "+var1+" und "+var2);
```

Bei Berechnungen ist es oft nötig, dass Strings in ein darstellbares Zahlenformat umgewandelt werden (Swing-Textfelder interpretieren bekanntlich alle Eingaben als Strings). Dafür kann der + Operator *nicht* verwendet werden, sondern es müssen spezielle Umwandlungsfunktionen verwendet werden:

```
double d = Double.ParseDouble( String a );

int i    = Integer.parseInt( String b );
```

```
String zahl1  = "1.2";
String zahl3  = "3";

double z1 = Double.parseDouble(zahl1);
int z3    = Integer.parseInt(zahl3);

System.out.println(z1 + z3);          // -> 4.2

System.out.println(zahl1 + zahl3);    // -> 1.23
```

Weitere nützliche Stringfunktionen

Es gibt zahlreiche weitere Attribute und Methoden, die den Umgang mit Zeichenketten erleichtern. Die wichtigsten sind:

```
concat();          // Verkettung
replace();         // Ersetzung
charAt(x);         // liefert Buchstabe an Position x.
equals();          // Inhaltsvergleich
equalsIgnoreCase(); // Ignoriert Unterschied Gross /
                       Kleinbuchstaben
compareTo();       // lexikal. Vergleich
indexOf();         // Position
lastIndexOf();     // letzte Position
toUpperCase();     // in Grossbuchstaben
toLowerCase();     // in Kleinbuchstaben
subString();       // sucht Teilstring
length();          // liefert Stringlänge
```

Diese Methoden werden für einen String aufgerufen, indem der zweite Parameter in Klammern übergeben wird. Der Ergebniswert ist je nach Art der Methode entweder eine neuer String, eine Zahl oder ein Wahrheitswert.

```
String vorname  = "Hans ";
String nachname = "Maier ";

nachname = nachname.replace('a','e');
System.out.println(nachname);           // -> Meier

String name1 = "Anna";
String name2 = "Anne";

System.out.println(name1.indexOf('a'));  // -> 3


String name1 = "Anna B";
String name2 = "Anna A";

System.out.println(name1.equals(name2)); // -> false

System.out.println(name1.compareTo(name2));  // -> 1
```