Java

Strings

D.A.Waldvogel

5. Strings / Zeichenketten in Java

5.1 Zeichenketten in Java

Genau wie die Felder sind auch Zeichenketten, die englisch *Strings* heissen, Objekte. Bei der Arbeit mit Zeichenketten muss man berücksichtigen, dass es zwei unterschiedliche Klassen von Strings gibt. Als erstes hat Java eine Klasse mit Namen *String*. Objekte dieser Klasse sind unveränderbar, sie stellen Zeichenketten mit fester Länge dar.
Die zweite Klasse nennt man *StringBuffer*. Diese Zeichenketten können zur Laufzeit des Programms verkürzt oder verlängert werden.

5.2.1 Literals

In Java werden *literal strings* zwischen Hochkommas geschrieben.

```
"Hello World!"
```

Literal strings können überall anstelle von Stringobjekten verwendet werden. System.out.println z.B. akzeptiert anstelle von einem Stringobjekt auch einen *literal string*.

```
System.out.println("Might I add that you look lovely today.");
```

Man kann auch direkt Methoden eines *literal strings* aufrufen.

```
int len = "Goodbye Cruel World".length();
```

Weile der Kompiler für jeden *literal string* automatisch `new String` durchführt, kann ein String auch wie folgt initialisiert werden:

```
String s = "Hola Mundo";
```

Das obige Konstrukt ist zwar gleichwertig und effizienter als das folgende. Hier werden eigentlich zwei Strings erzeugt anstelle von einem.

```
String s = new String("Hola Mundo!");
```

Der Kompiler kreiert das erste Stringobjekt wenn er den *literal string* "Hola Mundo!" erkennt, und das zweite bei `new String`.

5.2.2 String mit fester Länge

Ein String ist ein Objekt, bestehend aus mehreren Zeichen. Im Gegensatz dazu gibt es noch die Datentypen, *byte, short, int, long, float, double* und auch *char*, sie bestehen aus einzelnen Zeichen oder Zahlen. In Variablen von diesen Datentypen kann man Daten speichern und unter den Namen der Variablen auch wieder ansprechen bzw. aufrufen. Ein Objekt erweitert die Variablen zusätzlich um Methoden, die neben den Daten Bestandteil des Objektes sind. Da ein String ein Objekt ist, muss er instanziiert, also erzeugt werden. In Java geschieht dieses Erzeugen, also das Instanziieren, eines neuen Objektes durch den Befehl `new`.
Strings entstehen also, im Gegensatz zu den Variablen, durch Instanziierung von Objekten.

Wir wollen nun einen String definieren mit Namen „Nachname", der soll den Wert „Popp" tragen. Der java-Befehl lautet:

```
String nachname = new String("Popp");
```

dafür kann man auch kürzer schreiben

```
String nachname = "Popp";
```

Hat man noch die Definition

```
String vorname="Heribert";
```

Kann, mit dem + Befehl

```
System.out.println(vorname +", " +nachname);
```

der vollständige Name ausgeschrieben werden, wobei das + nach vorname die zwei Strings, den Inhalt von vorname und den Inhalt von "+ ", verbindet, genauso wie das + vor nachname.
+ verbindet also zwei Strings beim Ausschreiben.
Will man einen String mit einem anderen verbinden und das Ergebnis im ersten String abspeichern, verwendet man die Methode *concat*, die jeder erzeugte String hat.

Bei der Methode *concat* definiert man zuerst, von welchem Objekt, also String, sie ist und als Argument enthält sie den zweiten String, der mit dem ersten verbunden werden soll.
Dazu schauen wir uns ein Beispiel an:

Beispiel:

```
class stringtest
{
      public static void main(String args[])
      { String vorname = "Andreas";
        String nachname = new String("Popp");
        String sohnname = "Name: ";
        sohnname =sohnname.concat(vorname);
        sohnname =sohnname.concat(" ");
        sohnname =sohnname.concat(nachname);
        System.out.println(sohnname);
        System.out.println(sohnname.length());
      }
}
```

Nach der Klassen- und Methodendefinition werden dreimal Objekte vom Typ String angelegt (vorname, nachname und sohnname). Einmal wird dazu das Schlüsselwort *new* benutzt, und zwar bei nachname. Es würde aber auch hier die Zuweisung reichen, um ein Objekt zu erzeugen.
Dann wird die Methode *concat* benutzt, um eine Zeichenkette mit einer anderen zu verknüpfen.
Die letzte Methode in diesem Beispiel für das Objekt sohnname ist die Methode length. Sie ermittelt die Länge der Zeichenkette.

Das Ergebnis unseres Programmes lautet daher also:

```
Name: Andreas Popp
18
```

Umwandlung von Java Strings in numerische oder boolsche Werte

Java ist so konzipiert, dass immer der Typ, in den umgewandelt werden soll, auch die Methode für diese Umwandlung besitzt.
Um einen Wert in einen String umzuwandeln, benutzt man die Methode „*valueOf*“ der Klasse String. Der Aufruf der Methode kann erfolgen, indem man den Namen der Klasse String, gefolgt vom Methodennamen benutzt: `String.valueOf(...)`
z.B.

```
String test="";
boolean bool_var=true
test= String.valueOf(bool_var);
```

Die Variable „test“ enthielte dann die Zeichen „true“.
Eine Variante wäre hier, den Namen eines String-Objektes, gefolgt vom Methodennamen zu benutzen, also z.B. `test.valueOf(...).`

Auch die Umwandlung von String in andere Typen ist möglich, wie folgendes Beispiel anhand der vier Datentypen `boolean`, `Integer`, `short` und `double`, zeigt:

```
test="false";
bool_var=Boolean.valueOf(test).booleanValue();

test="567";
short_var=Short.parseShort(test);
```

```
test="9";
i=Integer.parseInt(test);

test ="8423.90321";
double_var = Double.valueof(test).doubleValue();
```

So liefert die Methode `valueOf` der Klasse Boolean ein Objekt vom Typ `boolean` zurück. Die Methode `BooleanValue` liefert dann zum Schluß den Wert `boolean` zurück.
Für die Ganzzahltypen, wie `short` und `int`, gibt es die Methode `parseShort` bzw. `parseInt`. Leider gibt es diese Mehode `parseInt` nicht für den Datentyp `Double`. Hier wird die gleiche Technik benutzt wie bei `Boolean`. Das gilt auch für die anderen Datentypen, die an dieser Stelle nicht explizit aufgeführt sind.

5.2.2 StringBuffer

StringBuffer dienen dazu, variable Zeichenketten zu verwalten. Die Methoden, die sie hauptsächlich unterstützen, sind Methoden zum Anhängen und Einfügen von Zeichenketten. Sie heissen `insert` und `append`.
Betrachten wir dazu ein Beispiel:

Beispiel:

```
class stringbuffer
{
      public static void main(String args[])
      {StringBuffer puffer = new StringBuffer();
       puffer.append("Andreas ").append("Popp");
       puffer.insert(8,"Heribert ");
       System.out.println(puffer);
      }
}
```

Mit den Befehl `new` wird ein neues Zeichenobjekt erzeugt.
Mit der Methode `append` kann man Teile an eine Zeichenkette anhängen.
Mit `insert` wird ab einer bestimmten Position ein neuer Teil in eine Zeichenkette eingefügt.
Beachten Sie, dass sich die Position auf den Index bezieht und daher mit dem Wert 0 anfängt.