



## 5.4 Strings

Zeichenkette, folgende Kernzeichen:

- Komponenten homogen vom Typ char
- direkter Zugriff auf Komponenten
- variable Länge

typ. Operationen:

- Verkettung 
- Substrings finden 
- Ändern, Einfügen

Im JAVA gibt es 2 Klassen für Strings

String



„ändern sich nach  
Erzeugung nicht  
mehr

StringBuffer



sind variabel, können  
verlängert/verkürzt werden

Im Unterschied zu C (C++ gilt:

String  $\neq$  char[]

char[] string = { 'H', 'a', 'l', 'l', 'o' };

string → [ H | a | l | l | o ]    habe Zugriff  
string[3] = 'l'

String string = "Hallo";

string → [ Hallo ]    habe Zugriff  
string.charAt(3) = 'l'

Operationen auf Strings:

public String ()    // konstruiert leeren String  
public String (String str)    // kopiert str in neuen String  
public String (char[] arr)    // kopiert arr in neuen String

literale Initialisierung: String str = "Hallo";  
                                  " " = new String("Hallo");

public String trim ()    // schneidet vorne und hinten  
                                  // whitespace aus String weg

String str = " \t A B \t C \n ";  
                                  |                                   |  
                                  Tabulator                                   Zeilentrenner

str.trim() → "A B C"

public char charAt (int pos)    // Index 0, ..., length() - 1

String str = "Hallo";  
str.charAt(1) → 'a'

public int length ()

str.length() → 5

public void getChars( Anfangs-Pos im String,  
Ende-Pos im String,  
char-Array zum Schreiben,  
Anfangs-Pos im char-Array)

String str = "Hallo CoMa";  
char[] charArr = new char[12];  
str.getChars(3, 9, charArr, 0) → ['l', 'o', 'o', 'C', 'o', 'M']

Beispiel in Java:

Eingabe: Folge von Zahlen  $n, a_1, a_2, \dots, a_m$   
getrennt durch whitespace in TextArea

Ausgabe: formatierte nummerierte Liste  $a_1, \dots, a_n$

4	10	20
	30	40 50

→

0	: 10
1	: 20
2	: 30
3	: 40

StringDemo(str)

Input: String str gemäß Format  
Output: nummerierte Liste der ersten  $n$  Zahlen

$n := \text{read}(\text{str})$

init. Array  $\text{vec}[]$  der Länge  $n$

for  $i := 0$  to  $n-1$  do

$\text{vec}[i] := \text{read}(\text{str})$

for  $i := 0$  to  $n-1$  do

write  $i$  ':'  $\text{vec}[i]$  '\n'

read(str)

Input: String str gemäß Format

Output: erste Zahl aus str

str verkürzt um erste Zahl

(Beispiel:)

str = "4 10 20 30 40 50"

read

↓  
['4'] [' ', '1', '0', ' ', ..., '5', '0']

↓  
"4"

↓  
" 10 ... 50"

↓  
 $n := 4$

↓  
"10 ... 50"

↓  
['1', '0']

↓  
"10"

↓

$\text{vec}[0] := 10$

suche 1. ' '

bilde char-Array  
von ' ' bis Ende

bilde daraus  
String

trim() - löscht  
Whitespace am  
Anfang

suche 1. ' '

Applet String macht keine Fehlerbehandlung

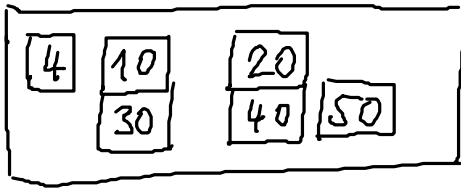
Zwei Verbesserungen:

① Fehlerbehandlung

② Einlesen vereinfachen

→ Klasse StringTokenizer

StringTokenizer (String str) // zerlegt str in Teilstrings,  
die durch whitespace  
getrennt sind



"Inseln im  
"Whitespaces-See"  
heißen "Tokens"

Methoden:

boolean hasMoreTokens ()

// sagt, ob noch Tokens vorhanden

String nextToken ()

// gibt nächstes Token in Reihen-  
folge des Strings

int countTokens ()

// sagt, wieviele Tokens noch kommen

Ausnahmen behandeln (Exception Handling)

String  
Demo

bisheriger Code

```
try  
{
```

```
}  
catch (Exceptionklasse1 Exceptionvariable1)  
{
```

---

```
}  
catch (Exceptionklasse2 Exceptionvariable2)  
{
```

```
}  
---
```

```
[ finally  
{  
---
```

optionaler Teil

} ]

typische Exceptions:

NumberFormatException  
NoSuchElementException  
IndexOutOfBoundsException  
NegativeArraySizeException  
DivisionByZeroException

...  
...

Laufzeit-  
Fehler