

Java – Methoden für Zeichenketten

public char charAt(int index)

gibt das Zeichen an der Position *index* eines Strings zurück. Dabei hat das erste Zeichen eines Strings den Index 0.

public int compareTo(String anotherString)

String1.compareTo(String2) vergleicht *String1* mit *String2*. Das Ergebnis ist kleiner als 0, falls *String1* lexikographisch vor *String2* kommt, und größer als 0, falls *String1* lexikographisch nach *String2* kommt. Ist das Ergebnis 0, sind die beiden Strings gleich.

public boolean contains(String s)

gibt *true* zurück, falls der String den übergebenen (Teil-)String *s* enthält.

public int indexOf(String s)

gibt die Position innerhalb eines Strings zurück, an dem der übergebene (Teil-)String *s* zum ersten Mal auftritt. Die Methode gibt -1 zurück, falls der (Teil-)String *s* nicht gefunden wird.

public int length()

gibt die Länge eines Strings zurück.

public String substring(int begin, int end)

gibt den Teilstring eines Strings zurück, der an Position *begin* beginnt und an Position *end* endet. Dabei hat das erste Zeichen eines Strings den Index 0.

Java – Methoden für Listen

public boolean add(E element)

fügt das angegebene Element am Ende einer Liste hinzu (und gibt *true* zurück, wenn die Liste geändert wurde, sonst *false*).

public void add(int index, E element)

fügt das angegebene Element an der Position *index* in eine Liste ein. Bestehende Elemente ab dieser Position werden nach rechts verschoben.

public boolean contains(Object o)

gibt *true* zurück, wenn das angegebene Objekt *o* in der Liste enthalten ist.

public E get(int index)

gibt das Element zurück, das an der angegebenen Position *index* in der Liste steht.

public E remove(int index)

entfernt das Element, das an der angegebenen Position in einer Liste steht, und verschiebt alle nachfolgenden Elemente nach links.

public int size()

gibt die Anzahl der Elemente in der Liste zurück.

Python – Methoden für Zeichenketten

`find(s: str) -> int`

gibt die Position innerhalb eines Strings zurück, an dem der übergebene (Teil-)String *s* zum ersten Mal auftritt. Die Methode gibt -1 zurück, falls der (Teil-)String *s* nicht gefunden wird.

`len(text: str) -> int`

gibt die Länge des Strings *text* zurück.

`s in text`

gibt *True* zurück, falls der (Teil-)String *s* in dem String *text* enthalten ist.

`text[index]`

gibt das Zeichen an der Position *index* zurück. Dabei hat das erste Zeichen des Strings *text* den Index 0.

`text[begin:end]`

gibt den Teilstring, der an Position *begin* beginnt und an Position *end* endet, zurück. Dabei hat das erste Zeichen des Strings *text* den Index 0.

Python – Methoden für Listen

`append(element)`

fügt das angegebene Element am Ende einer Liste hinzu.

`extend(liste: list)`

fügt die Elemente der übergebenen Liste am Ende der Liste an.

`index(element)`

gibt den Index innerhalb der Liste zurück, an dem das übergebene Element zum ersten Mal auftritt. Wenn der Eintrag nicht existiert, wird ein *ValueError* ausgelöst.

`insert(index: int, element)`

fügt das angegebene Element an der Position *index* ein. Bestehende Elemente ab dieser Position werden nach rechts verschoben.

`len(liste: list)`

gibt die Anzahl der Elemente in der Liste zurück.

`pop(index: int)`

entfernt das Element, das an der angegebenen Position in einer Liste steht, und verschiebt alle nachfolgenden Elemente nach links.

SQL – Structured Query Language

Syntax einiger grundlegender Befehle, wobei optionale Teile in eckigen Klammern stehen
(Create, Insert, Update und Delete: nur für den Leistungskurs).

```
SELECT [DISTINCT] * | spalte1 [AS alias1], spalte2 [AS alias2],  
..., spalten [AS aliasn]  
FROM (tabelle1, tabelle2, ..., tabellem) |  
(tabelle1 INNER JOIN tabelle2 ON key1 = key2 ...) |  
(tabelle1 NATURAL JOIN tabelle2 ...)  
[WHERE bedingung1 [(AND | OR) bedingung2 ... (AND | OR) bedingungk] ]  
[GROUP BY spalte1[, spalte2, ..., spaltep] ]  
[HAVING gruppenBedingung1 [(AND | OR) gruppenBedingung2 ...  
(AND | OR) gruppenBedingungs] ]  
[ORDER BY spalte1 [ASC | DESC] [, spalte2 [ASC | DESC],  
..., spaltet [ASC | DESC] ] ]
```

Operatoren für Berechnungen: +, -, *, /

Operatoren für Vergleiche in Bedingungen: =, != (ungleich), >, <, >=, <=, NOT,
LIKE (mit den Platzhaltern _ und %), BETWEEN, IN, IS NULL

Aggregatfunktionen: AVG, COUNT, MAX, MIN, SUM

```
INSERT INTO tabellex [(spalte1, spalte2, ..., spalten)]  
VALUES (wert1, wert2, ..., wertn)
```



```
UPDATE tabellex  
SET spalte1 = wert1 [, spalte2 = wert2, ..., spalten = wertn]  
WHERE bedingung1 [(AND | OR) bedingung2 ... (AND | OR) bedingungk]
```



```
DELETE FROM tabellex  
WHERE bedingung1 [(AND | OR) bedingung2 ... (AND | OR) bedingungk]
```

Relationenalgebra

Operation	Bedeutung
$R \cup S$	Mengenoperatio- nen
$R \cap S$	
$R \setminus S$	
$R \times S$	
$\sigma_{\text{Bedingung}}(R)$	Selektion
$\pi_{\text{Attribute}}(R)$	Projektion
$\rho_{\text{alt} \rightarrow \text{neu}}(R)$	Umbenennung
$R \bowtie S$	Natural Join
$R \bowtie S$ tabelle1.key1 = tabelle2.key2	Equi Join

Registermaschine

	Operand	Bedeutung	Beispiel
Konstanten	#i	der Operand ist die Zahl i	LOAD #7
direkte Adressierung	i	der Operand ist das Register i	ADD 4
indirekte Adressierung	*i	der Operand ist das Register, dessen Nummer im Register i steht	STORE *3

Befehlsübersicht

Befehl	Bedeutung
LOAD x	Lädt x in den Akkumulator.
STORE x	Speichert den Wert des Akkumulators in x. Dabei darf x keine Konstante #i sein.
ADD x	Addiert x zum Akkumulator.
SUB x	Subtrahiert x vom Akkumulator. Falls x größer ist als der Wert des Akkumulators, wird der Wert des Akkumulators auf 0 gesetzt.
MUL x	Multipliziert x mit dem Akkumulator.
DIV x	Dividiert den Akkumulator durch x. Für x = 0 wird das Programm beendet.
GOTO M	Ein unbedingter Sprung zur Marke M.
JZERO M	Falls der Akkumulator den Wert 0 hat, erfolgt ein Sprung zur Marke M.
JNZERO M	Falls der Akkumulator einen Wert größer 0 hat, erfolgt ein Sprung zur Marke M.
END	Das Programm wird beendet.

Regulärer Ausdruck

Metazeichen	Bedeutung
.	Platzhalter für ein beliebiges Zeichen außer für neue Zeile: \n
\	\ hebt die besondere Bedeutung von Metazeichen auf, um diese als Text suchen zu können, bzw. macht aus Buchstaben Steuerzeichen.
... ...	Alternativen für das Suchmuster
[...]	Die in den eckigen Klammern stehenden Zeichen werden als Alternative verwendet. Es können Bereiche angegeben werden, z.B.: [a-p], [3-8]. [^...] negiert die Klasse. Die Zeichenklasse steht für genau ein Zeichen, kann aber mit Wiederholungszeichen (*, ?, +, {n,m}) vervielfältigt werden.
(...)	Gruppierung von Suchmustern
?	Erkennt das vorhergehende Element 0- oder 1-mal.
*	Erkennt das vorhergehende Element 0-, 1- oder n-mal.
+	Erkennt das vorhergehende Element 1- oder n-mal.
{n} {n,m}	Erkennt das vorhergehende Element (mindestens) n-mal bis höchstens m-mal.