

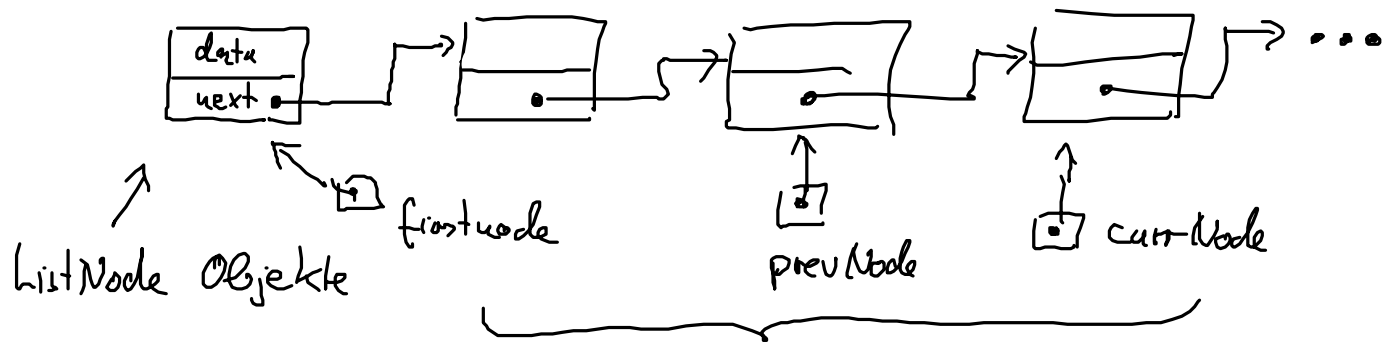
Schriftliche Rücksprache

Di 14.12.04 12⁰⁰ - 14⁰⁰ Uhr

Listen als Datenlinker

Klasse LinkedList selber geschrieben

Idee:



private Datenfelder der Klasse
LinkedList zur Implementation
(implementationsabhängig)

machen nur die Operationen public

bereit implementiert:

Konstruktor

isEmpty()

reset()

endOfList()

advance()

gibt

currentData()

gibt Daten des momentanen Listenelementes

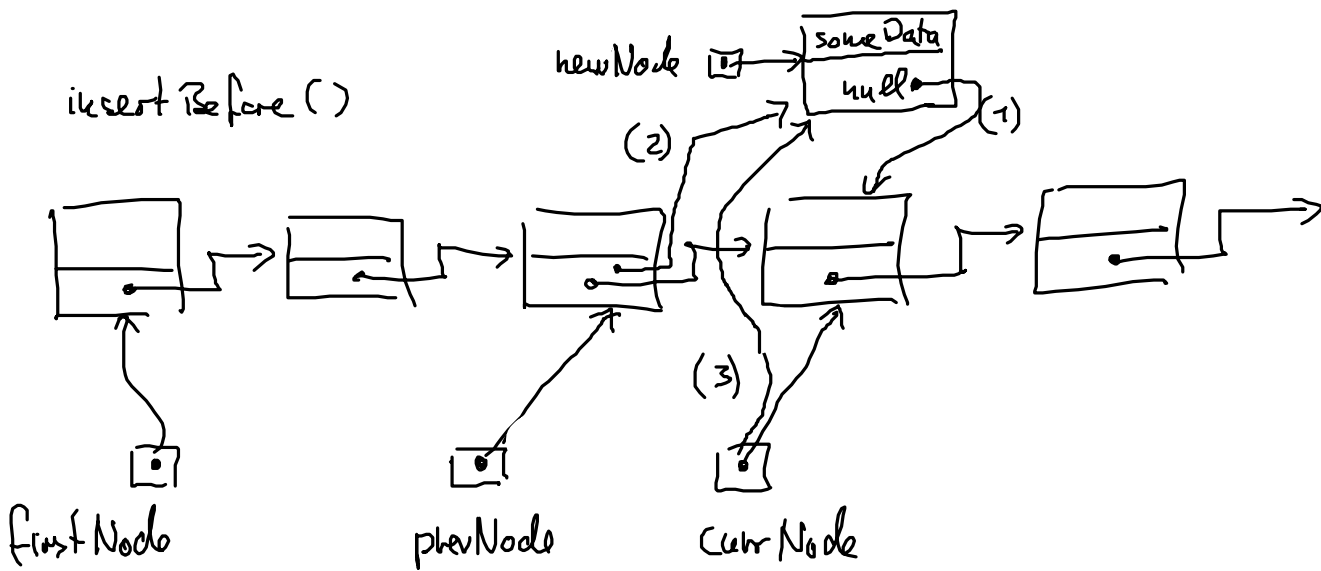
zurück

insert Before () fügt neuen Knoten vor aktuellem Listenelement ^(er)

insert After () fügt ... hinten ...

delete () löscht akt. Listenelement

Java code



// Erzeuge neuen Knoten mit Konstruktor der Klasse ListNode

```
ListNode newNode = new ListNode ( someData );
```

in Rot: die zu verändernden Referenzen

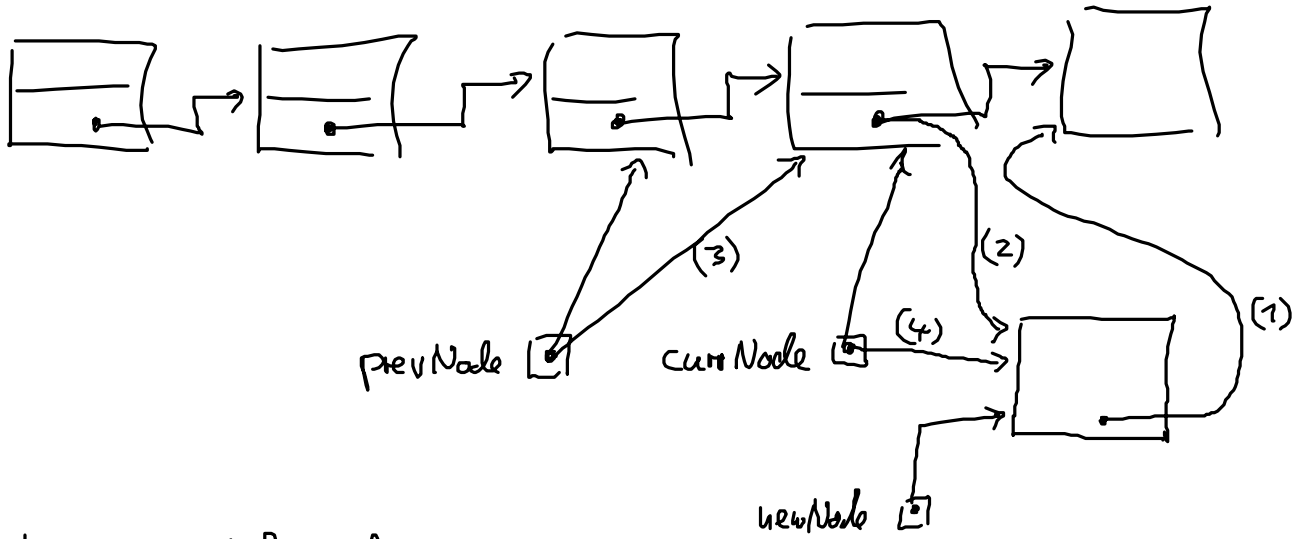
```
newNode.setNext ( currNode ); (1)
```

```
prevNode.setNext ( newNode ); (2)
```

```
currNode = newNode; // Alternative
```

```
currNode = prevNode.getNext();
```

insert After () Standardfall



rot = gegenseitige Referenzen

`newNode.setNext(currNode.getNext());` (1)

(2)

`currNode.setNext(newNode);`

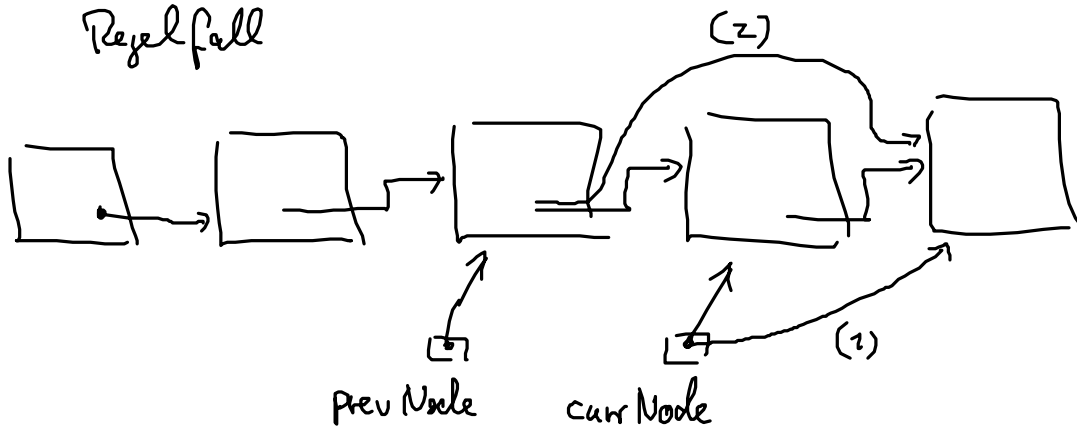
(3)

`prevNode = currNode;`

`currNode = newNode;`

(4)

delete() Regel fall



`currNode = currNode.getNext();`

`prevNode.setNext(currNode);`

denn ist Listenelement aus der Liste ausgehängt

es existiert physikalisch im Speicher noch weiter (kann von uns nicht mehr angesprochen werden). Es existiert so lange weiter bis die automatische Java-Garbage-Collection den Speicher wieder zur Verfügung stellt

Ein Test für List-Klasse:

- Einlesen eines Strings in eine Liste von Zeichen
- Ausgabe der Liste auf dem Bildschirm
- Löschen des Anfangs der Liste bis zu einem vorgegebenen ^{Zeichen}
- Ausgabe der jetzigen Liste

Liste enthält allgemeine Objekte

Zeichen sind char in Java

↑ eingebauter Typ ⇒ keine Objekte

daher Wrapper-Klasse Character von char verwenden

Umgekehrt beim Datenzugriff müssen die allgemeinen Objekte gecastet werden auf Character Objekte und aus denen der char Wert rausgeholt werden

```
Character ch;
```

```
ch = (Character) stringList.currentData();
```

```
char ch = ch.charValue();
```

Java Applet List Demo

5.7 Stacks

spezielle Listen, bei denen Einfügen und Löschen nur am Anfang der Liste erlaubt ist

Vorstellung: Stapel (= stack) von Tablets



=



← oberstes Element-
= Listenanfang
"top"

Operationen heißen dann:

top() Daten des obersten Elementes zurückgeben

pop() oberstes Element löschen

push() neues oberstes Element einfügen

isEmpty() Test auf leer

Klasse dafür schreiben

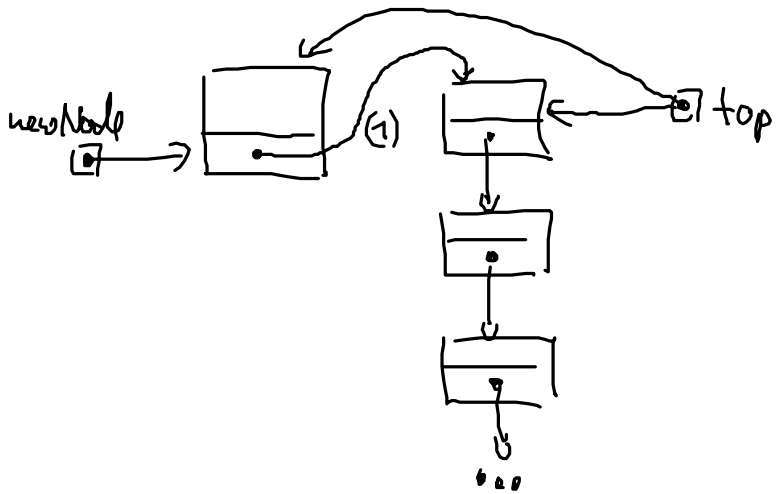
Anwendungen: Erkennung korrekter Klammerausdrücke

RPN-Redukter

Arbeitung von Methodenaufrufen in Prog. Sprache

Java Klasse ansehen

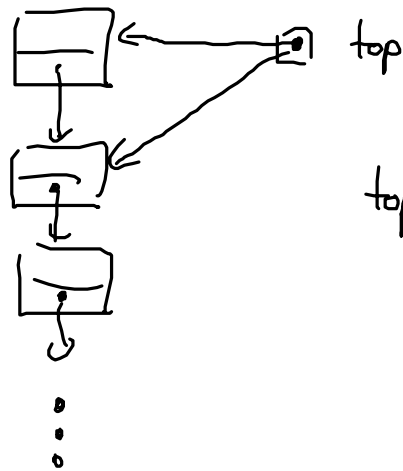
push() Standard full



`newNode.setNext(top);` (1)

`top = newNode;` (2)

pop() Standard full



`top = top.getNext();`