Matthias Jauernig
Kurs: 03In1

03.05.04
linux-related.de

6. Übungsaufgaben zur LV
Algorithmen & Datenstrukturen

Abgabetermin: Do, 06.05.04

1. Aufgabe:

- Vervollständigung der Klasse HashKett:

```
public Elementtyp suchen(String s) {
  KettItem h=suchPos(s);
  return h.Inhalt;
}

public boolean einfuegen(Elementtyp e) {
  Elementtyp d=suchen(e.getSchluessel());
  if(d!=null && d.getSchluessel().equals(e.getSchluessel()))
    return false;
  else{
    Anz++;
    int pos=Hashfkt(e.getSchluessel());
    KettItem z=(KettItem)getElement(pos);
    if(z.Inhalt==null)
      z.Inhalt=e;
    else{
      while(z.nxt!=null)
        z=z.nxt;
      z.nxt=new KettItem(e);
    }
    return true;
  }
}
```

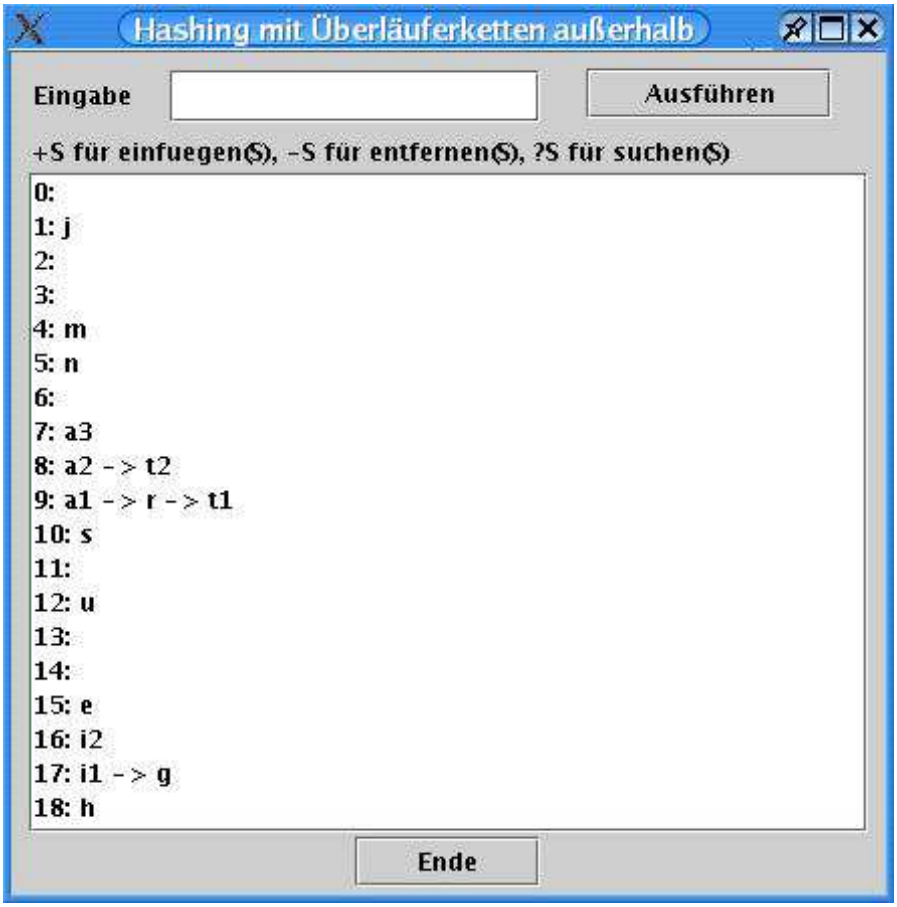
- Implementierung von Dialog_HashKett (auszugsweise):

```
public class Dialog_HashKett extends JDialog {
  String[] Zeilen={"0: ","1: ","2: ","3: ","4: ","5: ","6: ","7: ","8: ","9: ",
      "10: ","11: ","12: ","13: ","14: ","15: ","16: ","17: ","18: "};
  JList Ausg = new JList(Zeilen);
  HashKett hashkette=new HashKett(19);

  void OKBtn_actionPerformed(ActionEvent e1) {
    int i;
    ElementtypS e;
    String S = textfeld.getText();
    JOptionPane jOptPan = new JOptionPane();
    Ausg.clearSelection();
    if(S.length() > 0) {
      char c=S.charAt(0);
      S=S.substring(1);
      e=new ElementtypS(S);
      switch (c) {
        case '+':
          if(!hashkette.einfuegen(e))
            jOptPan.showMessageDialog(this,"Element '"+S+
                                      "' schon in der Hashkette vorhanden!");
          break;
        case '-':
          if(!hashkette.entfernen(e.getSchluessel()))
            jOptPan.showMessageDialog(this,"Element '"+S+
                                      "' konnte nicht gefunden werden!");;
          break;
```

```
        case '?':
          Elementtyp d=hashkette.suchen(e.getSchluessel());
          if(d==null || !d.getSchluessel().equals(e.getSchluessel()))
            jOptPan.showMessageDialog(this,"Element '"+S+
                                      "' wurde nicht gefunden!");
          else
            Ausg.setSelectedIndex(hashkette.Hashfkt(e.getSchluessel()));
          break;
      }
      if(c=='+' || c=='-'){
        Zeilen[hashkette.Hashfkt(e.getSchluessel())]=hashkette.Ausgabe
        ( hashkette.Hashfkt(e.getSchluessel()));
        Ausg.setListData(Zeilen);
      }
    }
  }
}
```

- Ausgabe (anhand des Beispiels meiner „name vorname“-Kombination:



2. Aufgabe:

- Quelltext von OKBtn_actionPerformed():

```
public class Dialog_HashCuD extends JDialog {
int hashart;
SuchStruk hash=null;

void OKBtn_actionPerformed(ActionEvent e1) {
  int i;
  boolean ok;
  String S = textfeld.getText();
  JOptionPane jOptPan = new JOptionPane();
  if (S.length() > 0) {
    char c = S.charAt(0);
    S = S.substring(1);
    ElementtypS e=new ElementtypS(S);
    jTable1.clearSelection();
    switch (c) {
      case '+':
        ok=hash.einfuegen(e);
        if(!ok) jOptPan.showMessageDialog(this,"Element '"+S+"' schon in der "+
                        "Hashtabelle vorhanden oder Hashtabelle voll!");
        break;
      case '-':
       if(!hash.entfernen(e.getSchluessel()))
              jOptPan.showMessageDialog(this,"Element '"+S+
                        "' konnte nicht gefunden werden!");
        break;
      case '?':
        Elementtyp d=hash.suchen(e.getSchluessel());
        if(d==null || !d.getSchluessel().equals(e.getSchluessel()))
          jOptPan.showMessageDialog(this,"Element '"+S+"' wurde nicht
                                        gefunden!");
        else{
          int suchpos;
          if(hashart==1) suchpos=((HashCoal)hash).suchPos(e.getSchluessel());
          else suchpos=((HashDopp)hash).suchPos(e.getSchluessel());
          jTable1.changeSelection(0,suchpos,false,false);
        }
        break;
    }
    if(c=='+' || c=='-')
      if(hashart==1) ((HashCoal)hash).Ausg(jTable1);
      else ((HashDopp)hash).Ausg(jTable1);
  }
}
```

- Ausgabe der mit meiner „name vorname“-Kombination gefüllten Tabellen:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	j	t2		m	n	r	a3	a2	a1	s	i1	t1	u		e	i2	g	h
0	1	1	-1	1	1	1	1	1	1	1	1	1	1	-1	1	1	1	1

- Nachweis-Überlegungen:

  Längste entstehende Überläuferketten beim Coalesced Hashing:

  Hierbei handelt es sich um "**i1 →g →i2 →s**" (Indizes 17–~~1~~6–~~1~~0–~~6~~)
  sowie "**a1 →r →t1 →h**" (Indizes 9–~~1~~8–~~1~~4–~~1~~1).

  Zum Nachweis dessen seien folgende Voraussetzungen gegeben:
  - Mein „name vorname“-Kombination gebe ich wie folgt ein:
    j a1 u e r n i1 g m a2 t1 t2 h i2 a3 s
  - Die Position in der Hashtabelle berechnet sich mit:
    $(z[0]-z[1]+z[2]-z[3]+z[4])\ \%\ 19$

  Unter diesen Voraussetzungen und der gegebenen Codierung der Zeichen von 0...36 wird wie folgt in die Hashtabelle eingefügt:

Zeichenkette	Beabsichtigte Pos.	Bemerkungen / Erläuterungen
j	20%19 = 1	Wird in 1 eingefügt.
a1	(11-2)%19 = 9	Wird in 9 eingefügt.
u	31%19 = 12	Wird in 12 eingefügt.
e	15%19 = 15	Wird in 15 eingefügt.
r	28%19 = 9	Will nach 9, ist aber besetzt. 9 hat keinen gültigen next-Zeiger. Größte freie Adresse ist 18, dort wird nun ausweichend eingefügt und verkettet. (9–~~1~~8)
n	24%19 = 5	Wird in 5 eingefügt.
i1	(19-2)%19 = 17	Wird in 17 eingefügt.
g	17%19 = 17	Will nach 17, ist aber besetzt. 17 hat keinen gültigen next-Zeiger. Größte freie Adresse ist 16, dort wird nun ausweichend eingefügt und verkettet. (17→16)
m	23%19 = 4	Wird in 4 eingefügt.
a2	(11-3)%19 = 8	Wird in 8 eingefügt.
t1	(30-2)%19 = 9	Will nach 9, ist aber besetzt. 9 ist verkettet mit 18, 18 hat keinen gültigen next-Zeiger. Größte freie Adresse ist 14, dort wird ausweichend eingefügt und verkettet. (9–~~1~~8–~~1~~4)
t2	(30-3)%19 = 8	Will nach 8, ist aber besetzt. 8 hat keinen gültigen next-Zegier. Größte freie Adresse ist 13, dort wird nun ausweichend eingefügt und verkettet. (8–~~1~~3)
h	18%19 = 18	Will nach 18, ist aber besetzt. 18 ist verkettet mit 14, 14 hat keinen gültigen next-Zeiger. Größte freie Adresse ist 11, dort wird ausweichend eingefügt und verkettet. (**9–~~1~~8–~~1~~4–~~1~~1**)
i2	(19-3)%19 = 16	Will nach 16, ist aber besetzt. 16 hat keinen gültigen next-Zeiger. Größte freie Adresse ist 10, dort wird nun ausweichend eingefügt und verkettet. (17–~~1~~6–~~1~~0)
a3	(11-4)%19 = 7	Wird in 7 eingefügt.
s	29%19 = 10	Will nach 10, ist aber besetzt. 10 hat keinen gültigen next-Zeiger. Größte freie Adresse ist 6, dort wird nun ausweichend eingefügt und verkettet. (**17–~~1~~6–~~1~~0–~~6~~**)