Informatik mit Java: Binäre Suche nach Strings in einem Array

Gierhardt Städtisches Gymnasium Bad Laasphe

```
1  class BinSucheRekMitStrings
2  {
3    String[] feld;
4    final int MAXINDEX=79999;
5    Stoppuhr uhr;
6
7    public BinSucheRekMitStrings()
8    { uhr  = new Stoppuhr();
9      feld = new String[MAXINDEX+1];
10   }
11
12   public int ganzeZufallszahlMitRandwerten(int low, int high)
13   { // Bsp.: ganzeZufallszahlMitRandwerten(1, 6) liefert die
14     //       Augenzahl eines Wuerfels
15     high++;
16     return (int) (Math.random() * (high - low) + low);
17   }
18
19   public void feldFuellen()
20   {   int asciinr;
21       char zeichen;
22       String inhalt;
23
24       for (int i=0; i<=MAXINDEX; i++)
25       {    inhalt = "";
26            // Ein Wort mit 6 Buchstaben zufaellig erzeugen.
27            for (int nr=0; nr<=5; nr++)
28            {  asciinr = ganzeZufallszahlMitRandwerten(65, 70); // A bis F
29               zeichen = (char)asciinr;
30               inhalt = inhalt + zeichen;
31            }
32            feld[i] = inhalt;
33       } // for
34
35   } // feldFuellen
36
37   public void feldAusgeben()
38   { System.out.println("Das Feld enthaelt die folgenden Elemente: ");
39     for (int i=0; i<=MAXINDEX; i++)
40     System.out.println("An der Stelle " + i + " steht " + feld[i] + "  .");
41     System.out.println();
42   } // feldAusgeben
43
44   public  void vertausche (int a, int b)
45   {  String ablage = feld[a];
46      feld[a] = feld[b];
47      feld[b] = ablage;
48   } // vertausche
49
50
```

```
51  public void quicksort(int links, int rechts)
52  { int nachlinks = rechts; //Index, der vom rechten Ende nach links laeuft
53    int nachrechts = links; //Index, der vom linken Ende nach rechts laeuft
54
55    if (nachrechts < nachlinks)
56       { // Pivotelement bestimmen
57         String pivot = feld[(nachrechts + nachlinks)/2];
58         // int pivot = feld[links];
59
60         while (nachrechts <= nachlinks)
61         {    // Links erstes Element suchen, das
62              // groesser oder gleich dem Pivotelement ist
63              while ((nachrechts < rechts) &&
64                     (feld[nachrechts].compareTo(pivot) < 0))
65                   nachrechts++;
66
67              // Rechts erstes Element suchen, das
68              // kleiner oder gleich dem Pivotelement ist
69              while ((nachlinks > links) &&
70                     (feld[nachlinks].compareTo(pivot) > 0))
71                   nachlinks--;
72
73              // Wenn nicht aneinander vorbei gelaufen, Inhalte vertauschen
74              if (nachrechts <= nachlinks)
75                   { vertausche(nachrechts, nachlinks);
76                     nachrechts++;
77                     nachlinks--;
78                   }
79         } // end while
80
81         // Linken Teil sortieren
82         if (nachlinks > links) quicksort (links, nachlinks);
83
84         // Rechten Teil sortieren
85         if (nachrechts < rechts) quicksort (nachrechts, rechts);
86       } // end if
87  }  // quicksort
88
89  public int binaereSucheIterativ(String wort,int low, int high)
90  {  int l = low;
91     int mitte;
92     int h = high;
93     while (l <= h)
94     {  mitte = (l + h) / 2;
95        if (feld[mitte].compareTo(wort)==0)
96             return mitte;
97        else if (feld[mitte].compareTo(wort) < 0) // feld[mitte] < wort
98             l = mitte + 1;
99        else h = mitte - 1;
100    } // end while
101    return -1; // nicht gefunden
102 }  //
```

```
103
104   public int binaereSucheRekursiv(String wort, int low, int high)
105   {  if (low <= high)
106           {  int mitte = (low + high)/2;
107
108              if (feld[mitte].compareTo(wort)==0)
109                   // fertig, denn feld[mitte]==wort
110                   return mitte;
111
112              else if (feld[mitte].compareTo(wort) > 0)
113                   // rekursiv links weiter, weil feld[mitte] > wort
114                   return binaereSucheRekursiv(wort, low, mitte-1);
115
116              else // rekursiv rechts weiter, weil feld[mitte] < wort
117                   return binaereSucheRekursiv(wort, mitte+1, high);
118           }
119      else return -1; // wort nicht gefunden
120   }
121
122   public void jetzt_mach_mal_was()
123   {
124     String suchwort = "ABCDEF";
125     System.out.println("Binaere Suche: ");
126     feldFuellen();
127
128     System.out.println("Erst mal sortieren: ");
129
130     uhr.starte();
131     quicksort(0, MAXINDEX);  // 46 ms bei 80000
132     uhr.stoppe();
133
134     feldAusgeben();             // Zur Kontrolle
135
136     System.out.println("Beginn der Suche:");
137     //int pos=binaereSucheRekursiv(suchwort, 0, MAXINDEX);
138
139     int pos=binaereSucheIterativ(suchwort, 0, MAXINDEX);
140     if (pos>=0)
141          { System.out.print("Wort " + suchwort);
142            System.out.println(" steht an Stelle " + pos );
143            System.out.println("Kontrolle: feld[" + pos + "]=" + feld[pos]);
144          }
145     else System.out.println(suchwort + " wurde nicht gefunden.");
146
147     System.out.println("Zeit zum Sortieren: " + uhr.lies() + " ms.");
148
149   } // end jetzt_mach_mal_was
150
151 } // class BinSucheRekMitStrings
```