Verwaltung eines kleinen Fuhrparks in Java
mit Datei-Operationen und Sortieren

Der Quelltext besteht aus vier Dateien, nämlich

1. Kfz - Beschreibung eines Autos mit den notwendigen Methoden
2. KfzFuhrpark - Verwaltung von bis zu 25 Autos in einer Reihung
3. KfzGUI - grafische Oberfläche zur Verwaltung der Autos
4. KfzStart - Start der grafischen Oberfläche als Application oder als Applet

---

```
1   //Kfz-Verwaltung, Datei 1/4
2   //Java JDK 1.1.18 -- Krell 21.2.2002/5.3.2002
3   import java.io.*;  //für Serializable
4
5   public class Kfz implements Serializable //implements S.. für Datei-op.!
6   {
7     //Hier stehen die Daten, die ein (jedes) Auto beschreiben.
8     //Das vorangestellte private sorgt dafür, dass nur die Methoden
9     //dieser Klasse auf die Daten zugreifenb dürfen (verhindert Fehler)
10    private String kennzeichen; // z.B. "D-AB 123"
11    private String hersteller;  // z.B. "VW"
12    private String typ;         // z.B. "Polo"
13    private int baujahr;        // z.B. 1998
14    private int kmStand;        // z.B. 42389
15
16    //Und hier kommen die Methoden, um die Daten zu ändern oder anzusehen:
17
18    public void merken (String nummer, String firma, String modell,
19      int jahr, int fahrtstrecke)
20      //Beim Aufruf auto.merken ("D-CD 234","Opel","Astra",2001,10893)
21      //werden die übergebenen Daten in die fünf Variablen eingetragen:
22      {
23        kennzeichen = nummer;
24        hersteller = firma;
25        typ = modell;
26        baujahr = jahr;
27        kmStand = fahrtstrecke;
28      }
29
30    public String nennen ()
31      //Gibt eine Zeichenkette mit allen Daten des Autos aus
32      {
33        String daten = "Das Auto mit der Nr. '"+kennzeichen+"', ein "+
34         hersteller+" "+typ+" von "+baujahr+", ist bisher "+kmStand+
35             " km gelaufen.";
36        return (daten);
37      }
38
39    public String nummer()
40      //Nennt nur das amtl. Kennzeichen des Autos, sonst nichts
41      {
42        return (kennzeichen);
43      }
44
45    public String hersteller()
46      //Nennt nur den Hersteller des Autos, sonst nichts
47      {
48        return (hersteller);
49      }
50
51
52    public int kmZahl ()
```

```
53        //Nennt nur den kmStand des Autos, sonst nichts
54        {
55          return (kmStand);
56        }
57
58      public void neuerStand (int km)
59        //Ändert nur den kmStand des Autos, sonst nichts
60        {
61          kmStand = km;
62        }
63
64      public void kopiereVon (Kfz anderesAuto)
65        //Kopiert alle Daten vom anderen Auto in eigene Daten
66        {
67          kennzeichen = anderesAuto.kennzeichen;
68          hersteller  = anderesAuto.hersteller;
69          typ         = anderesAuto.typ;
70          baujahr     = anderesAuto.baujahr;
71          kmStand     = anderesAuto.kmStand;
72        }
73    }
```

```
1     //Kfz-Verwaltung, Datei 2/4
2     //Java JDK 1.1.18 -- Krell 21.2.2002/5.3.2002
3     import java.io.*;
4
5     //Erzeugt und verwaltet Reihung fuhrpark aus 25 Objekten der Klasse Kfz
6     public class KfzFuhrpark
7     {
8       //Daten des Fuhrparks
9       private Kfz[] fuhrpark = new Kfz[25]; //(bis zu) 25 Autos im Fuhrpark
10      private int anzahlDerAutos = 0; //zunächst ist noch kein Auto bekannt.
11        //Erlaubt ist/sind 0 <= anzahlDerAutos <= 25
12
13      //Methoden zur Verwaltung des Fuhrparks
14
15      public KfzFuhrpark() //Automatische Initialisierung
16      {
17        for (int i=0; i<25; i++)
18        {
19          fuhrpark[i] = new Kfz(); //Erzeugen der 25 Objekte vom Typ Kfz
20        }
21      }
22
23      public int wagenAnzahl () //Sagt, wie viele Autos es gibt
24      {
25        return (anzahlDerAutos);
26      }
27
28      public void neuesAuto (String nummer, String produzent, String modell,
29        int jahr, int kmStand) //Erweitert fuhrpark um das angegebene Auto
30      {
31        fuhrpark[anzahlDerAutos].merken (nummer,produzent,modell,jahr,kmStand);
32          //Speichert Daten im fuhrpark an der indizierten Stelle durch Aufruf
33          //der Kfz-Methode merken. Jede der 25 fuhrpark-Komponenten ist ein Kfz!
34        anzahlDerAutos = anzahlDerAutos + 1;
35          //Durch die Neuaufnahme hat sich die Zahl der Autos um 1 erhöht!
36      }
37
38      public String ändereKmStand (String amtlKennzeichen, int neueKm)
39      //Ändert den kmStand des Autos mit dem angeg. Kennzeichen
40      {
41         String meldung = "Auto "+amtlKennzeichen
42                          +" nicht gefunden -- km-Zahl nicht geändert";
```

```
43          //Finden des Index i vom Auto mit der gesuchten Nummer
44          int i = anzahlDerAutos - 1; //von hinten beginnen
45          while ((i>=0)&&(! fuhrpark[i].nummer().equals(amtlKennzeichen)))
46          {           //equals statt == , da versch. Objekte mit gleichem Inhalt
47            i = i-1;
48          }
49          //KmStand ändern
50          if (i>=0) //Auto gefunden -- sonst ist i = -1
51          {
52            fuhrpark[i].neuerStand (neueKm); //Aufruf der Kfz-Methode
53            meldung = "km-Zahl des Autos "+amtlKennzeichen+
54                            " auf "+neueKm+" verändert";
55          }
56          return (meldung);
57        }
58
59        public String löschen (String altesKennzeichen)
60        //Löscht Auto mit dem angegebenen Kennzeichen aus dem Fuhrpark
61        {
62          String meldung = "Auto "+altesKennzeichen
63                              +" nicht gefunden -- nicht gelöscht";
64          //Finden des Index i vom Auto mit der gesuchten Nummer
65          int i = anzahlDerAutos - 1; //von hinten beginnen
66          while ((i>=0)&&(! fuhrpark[i].nummer().equals(altesKennzeichen)))
67          {           //equals statt == , da versch. Objekte mit gleichem Inhalt
68            i = i-1;
69          }
70          //Auto löschen durch Vorziehen des Fuhrpark-Restes
71          if (i>=0) //Auto gefunden -- sonst ist i = -1
72          {
73            anzahlDerAutos--; //Löschen verkleinert Anzahl
74            for (int j=i; j < anzahlDerAutos; j++)
75            {
76              fuhrpark[j].kopiereVon (fuhrpark[j+1]); //nutzt Kfz-Methode
77            }
78            meldung = "Auto "+altesKennzeichen+" wurde gelöscht!";
79          }
80          return (meldung);
81        }
82
83        public String listeAutosVon (String produzent)
84        //Erzeugt Text mit den Daten der Autos vom genannten Hersteller
85        {
86          String ausgabe = "";
87          for (int i=0; i<anzahlDerAutos; i++)
88          {
89            if (fuhrpark[i].hersteller().equals (produzent))
90            {
91              ausgabe = ausgabe+(i+1)+".) "+fuhrpark[i].nennen()+"\n";
92            }
93          }                                  //nutzt Kfz-Methode
94          return (ausgabe);
95        }
96
97        private void tausche (int i, int j)
98        // vertauscht innerhalb des Fuhrparks die beiden Autos
99        // an den Positionen i und j. Wird beim Sortieren benötigt.
100       {
101         Kfz drittesAuto = new Kfz();  //Zwischenspeicher
102         drittesAuto.kopiereVon (fuhrpark[i]);
103         fuhrpark[i].kopiereVon (fuhrpark[j]);
104         fuhrpark[j].kopiereVon (drittesAuto);
105       }
106
107       public void minSortNr () // MinSort nach Autonummer
108       // Sortieren durch Auswahl des jeweils kleinsten Elements im Rest
109       // und Tausch dieses Elements nach vorne. Nach dem 0. Durchgang steht
110       // also das kleinste Element ganz links auf Platz 0, beim nächsten Durchgang
```

```
111     // kommt das nächtkleinere Element auf Platz 1 usw., so dass die Sortierung
112     // von links nach rechts wächst
113     {
114       int minStelle, durchgang, stelle;
115
116       for (durchgang=0; durchgang<anzahlDerAutos; durchgang++)
117       {
118         minStelle = durchgang; // vorderstes El. im unsortierten Teil zum Vergleich
119         for (stelle=durchgang+1; stelle<anzahlDerAutos; stelle++)
120         {
121           if (fuhrpark[stelle].nummer().compareTo(fuhrpark[minStelle].nummer())<0)
122           {
123             // wird ein kleineres Element gefunden, so wird dessen Index gemerkt
124             minStelle = stelle;
125           }
126         }
127         // das gefundene kleinste Element wird nach vorne (auf den Platz mit
128         // dem Index durchgang) getauscht, sofern es nicht schon dort steht:
129         if (minStelle > durchgang)
130         {
131           tausche (durchgang, minStelle);
132         }
133       }
134     }
135
136     public void minSortKm () // MinSort nach km-Stand
137     // Sortieren durch Auswahl des jeweils kleinsten Elements im Rest
138     // und Tausch dieses Elements nach vorne. Nach dem 0. Durchgang steht
139     // also das kleinste Element ganz links auf Platz 0, beim nächsten Durchgang
140     // kommt das nächtkleinere Element auf Platz 1 usw., so dass die Sortierung
141     // von links nach rechts wächst
142     {
143       int minStelle, durchgang, stelle;
144
145       for (durchgang=0; durchgang<anzahlDerAutos; durchgang++)
146       {
147         minStelle = durchgang; // vorderstes El. im unsortierten Teil zum Vergleich
148         for (stelle=durchgang+1; stelle<anzahlDerAutos; stelle++)
149         {
150           if (fuhrpark[stelle].kmZahl() < fuhrpark[minStelle].kmZahl())
151           {
152             // wird ein kleineres Element gefunden, so wird dessen Index gemerkt
153             minStelle = stelle;
154           }
155         }
156         // das gefundene kleinste Element wird nach vorne (auf den Platz mit
157         // dem Index durchgang) getauscht, sofern es nicht schon dort steht:
158         if (minStelle > durchgang)
159         {
160           tausche (durchgang, minStelle);
161         }
162       }
163     }
164
165     public String aufDisk (String dateiName)
166     //schreibt alle Autos des Fuhrparks unter dateiName auf Disk
167     {
168       ObjectOutputStream oos;
169       String meldung = "";
170       try
171       {
172         oos = new ObjectOutputStream(new FileOutputStream (dateiName));
173           //öffnet die Datei zum Schreiben. Evtl. alte Datei wird überschrieben.
174         for (int i=0; i<anzahlDerAutos; i++) //geht ganzen Fuhrpark durch
175         {
176           oos.writeObject (fuhrpark[i]); //schreibt jeweils ein Auto vom Typ Kfz
177         }
178         oos.close(); //schließt die Datei
179         meldung = ""+anzahlDerAutos+" Auto(s) in '"+dateiName+"' gespeichert.\n";
```

```
180
181       }
182       catch (IOException ex)
183       {
184         meldung = "Fehler beim Speichern: "+ex+"\n";
185       }
186       return (meldung); //gibt Erfolgs- oder Fehlermeldung zurück
187     }
188
189     public String vonDisk (String dateiName)
190     //ersetzt den bisherigen Fuhrpark durch die Autos von Disk
191     {
192       ObjectInputStream ois;
193       String meldung = "";
194       int i = 0; //hier deklariert, damit noch nach for-Schleife verfügbar
195       try
196       {
197         ois = new ObjectInputStream(new FileInputStream(dateiName));
198           //öffnet Datei zum Lesen (Fehler, falls nicht gefunden)
199         try
200         {
201           for (i=0; i<25; i++) //Schleife wird durch EOF abgebrochen!
202           {
203             fuhrpark[i] = (Kfz) ois.readObject (); //liest ein Auto
204           }
205           //Diese Stelle wird nie erreicht: Durch EOF aus Schleife in catch
206         }
207         catch (EOFException eex) //Fehler entsteht durch Dateiende
208         {
209            ois.close();          //Datei schließen
210            anzahlDerAutos = i;   //anzahlDerAutos = gelesene Zahl
211              //letztes gelesenes Auto hat Wert null => nicht berücksichtigen,
212              //also nicht anzahlDerAutos = i+1;
213            meldung = ""+anzahlDerAutos+" Auto(s) aus Datei '"+dateiName
214                           +"' gelesen.";
215         }
216       }
217       catch (Exception ex) // nicht nur IO-, auch ClassNotFound-Exception!
218       {
219         meldung = "Fehler beim Lesen: "+ex+"\n";
220       }
221       return (meldung);
222     }
223
224     public String listeAlle()
225     //Erzeugt Text mit den Daten aller Fahrzeuge: 1 Zeile pro Auto
226     {
227       String ausgabe = "";
228       for (int i=0; i<anzahlDerAutos; i++)
229       {
230         ausgabe = ausgabe+(i+1)+".) "+fuhrpark[i].nennen()+"\n";
231       }                                    //nutzt Kfz-Methode
232       return (ausgabe);
233     }
234   }
```

```
1   //Kfz-Verwaltung, Datei 3/4
2   //Java JDK 1.1.18 -- Krell 21.2.2002/5.3.02
3   import java.awt.*;
4   import java.awt.event.*;
5   import java.io.*;
6
7   //Benutzer-Oberfläche der Kfz-Verwaltung, die ihrerseits ein Objekt
8   //namens alleAutos der Klasse KfzFuhrpark anlegt und verwaltet
9
```

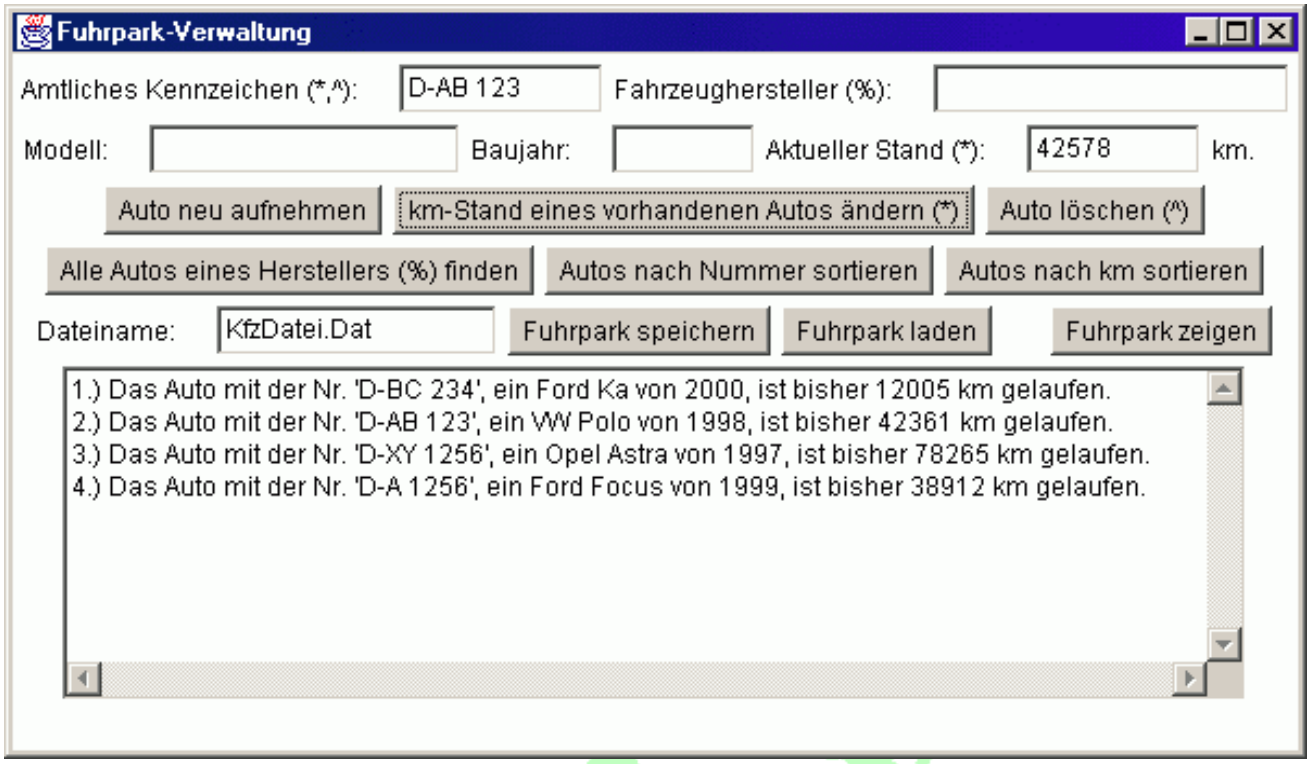
```
10  public class KfzGUI extends Frame
11  {
12    KfzFuhrpark alleAutos   = new KfzFuhrpark();  //Def. und Erzeugen
13
14    TextField tfKennzeichen = new TextField("",10);
15    TextField tfHersteller  = new TextField("",20);
16    TextField tfModell      = new TextField("",17);
17    TextField tfBaujahr     = new TextField("",6);
18    TextField tfKmStand     = new TextField("",8);
19    TextField tfDateiname   = new TextField("KfzDatei.Dat",15);
20    Button    btNeuesAuto   = new Button("Auto neu aufnehmen");
21    Button    btNeueKm      = new Button("km-Stand eines vorhandenen Autos ändern (*)");
22    Button    btLöschen     = new Button("Auto löschen (^)");
23    Button    btHersteller  = new Button("Alle Autos eines Herstellers (%) finden");
24    Button    btSortNr      = new Button("Autos nach Nummer sortieren");
25    Button    btSortKm      = new Button("Autos nach km sortieren");
26    Button    btSpeichern   = new Button("Fuhrpark speichern");
27    Button    btLaden       = new Button("Fuhrpark laden");
28    Button    btZeigen      = new Button("Fuhrpark zeigen");
29    TextArea  taAusgabe     = new TextArea(9,75);
30
31    public void richteFensterEin() // Fenster initalisieren und beschreiben
32    {
33      //WindowsListener hinzufügen, damit Schließknopf funktioniert
34      addWindowListener (
35        new WindowAdapter ()
36        {
37          public void windowClosing (WindowEvent ereignis)
38          {                 //ersetzt bisher leere Methode
39            setVisible (false);
40            dispose();
41            System.exit(0);
42          }
43        }
44      ); // runde Klammer vom Windowlistener geschlossen;
45    }
46
47    public void richteKnöpfeEin()
48    {
49       btNeuesAuto.addActionListener (
50         new ActionListener ()
51         {
52           public void actionPerformed (ActionEvent e)
53           {
54             taAusgabe.setText ("Bitte alle Felder ausfüllen -- Ganze Zahlen bei
55                                 Baujahr und km-Stand!");
56             alleAutos.neuesAuto (tfKennzeichen.getText(), tfHersteller.getText(),
57               tfModell.getText(), Integer.parseInt(tfBaujahr.getText()),
58               Integer.parseInt(tfKmStand.getText())); //Auto aufnehmen
59             taAusgabe.setText ("Auto "+tfKennzeichen.getText()+" aufgenommen!\n");
60             tfKennzeichen.setText(""); //und Eingabefelder löschen
61             tfHersteller.setText("");
62             tfModell.setText("");
63             tfBaujahr.setText("");
64             tfKmStand.setText("");
65           }
66         }); //runde Klammer (von btNeuesAuto.addActionListener)
67
68       btNeueKm.addActionListener (
69         new ActionListener ()
70         {
71           public void actionPerformed (ActionEvent e)
72           {
73             taAusgabe.setText ("Bitte Kennzeichen eingeben und eine ganze Zahl für
74                                 den (neuen) km-Stand!");
75             taAusgabe.setText(alleAutos.ändereKmStand (tfKennzeichen.getText(),
76               Integer.parseInt(tfKmStand.getText()))); //km ändern, Erfolg melden
77             tfKennzeichen.setText(""); //und Eingabefelder löschen
```

```
78              tfKmStand.setText("");
79            }
80          }); //runde Klammer (von btNeueKm.addActionListener)
81
82        btLöschen.addActionListener (
83          new ActionListener ()
84          {
85            public void actionPerformed (ActionEvent e)
86            {
87              taAusgabe.setText(alleAutos.löschen (tfKennzeichen.getText()));
88                 //Auto löschen, Erfolg melden
89              tfKennzeichen.setText(""); //und Eingabefeld löschen
90            }
91          }); //runde Klammer (von btLöschen.addActionListener)
92
93        btHersteller.addActionListener (
94          new ActionListener ()
95          {
96            public void actionPerformed (ActionEvent e)
97            {
98              taAusgabe.setText(alleAutos.listeAutosVon (tfHersteller.getText()));
99              tfHersteller.setText(""); //und Eingabefeld löschen
100           }
101         }); //runde Klammer (von btHersteller.addActionListener)
102
103       btSortNr.addActionListener (
104         new ActionListener ()
105         {
106           public void actionPerformed (ActionEvent e)
107           {
108             alleAutos.minSortNr ();
109             taAusgabe.setText("Autos wurden lexikografisch nach Kennzeichen
110                                sortiert");
111           }
112         }); //runde Klammer (von btSortNr.addActionListener)
113
114       btSortKm.addActionListener (
115         new ActionListener ()
116         {
117           public void actionPerformed (ActionEvent e)
118           {
119             alleAutos.minSortKm ();
120             taAusgabe.setText("Autos wurden nach dem km-Stand sortiert");
121           }
122         }); //runde Klammer (von btSortKm.addActionListener)
123
124       btSpeichern.addActionListener (
125         new ActionListener ()
126         {
127           public void actionPerformed (ActionEvent e)
128           {                    //alle Autos auf Disk speichern
129             taAusgabe.setText( alleAutos.aufDisk (tfDateiname.getText()));
130           }
131         }); //runde Klammer (von btSpeichern.addActionListener)
132
133       btLaden.addActionListener (
134         new ActionListener ()
135         {
136           public void actionPerformed (ActionEvent e)
137           {                     //alle Autos von Disk lesen
138             taAusgabe.setText( alleAutos.vonDisk (tfDateiname.getText()));
139           }
140         }); //runde Klammer (von btLaden.addActionListener)
141
142       btZeigen.addActionListener (
143         new ActionListener ()
144         {
145           public void actionPerformed (ActionEvent e)
```

```
146             {                    //Liste aller Autos anzeigen
147               taAusgabe.setText( alleAutos.listeAlle() );
148             }
149           }); //runde Klammer (von btZeigen.addActionListener)
150     } //Ende von richteKnöpfeEin
151
152
153     public void führeAus ()
154     {
155       setTitle("Fuhrpark-Verwaltung"); // Fenster-Titel
156       setSize (600,350); // Fenstergröße (Breite und Höhe in Pixeln)
157       setLayout (new FlowLayout());
158       richteFensterEin();
159       richteKnöpfeEin();
160       add (new Label("Amtliches Kennzeichen (*,^):"));
161       add (tfKennzeichen);
162       add (new Label("Fahrzeughersteller (%):"));
163       add (tfHersteller);
164       add (new Label("Modell:"));
165       add (tfModell);
166       add (new Label("Baujahr:"));
167       add (tfBaujahr);
168       add (new Label("Aktueller Stand (*):"));
169       add (tfKmStand);
170       add (new Label("km."));
171       add (btNeuesAuto);
172       add (btNeueKm);
173       add (btLöschen);
174       add (btHersteller);
175       add (btSortNr);
176       add (btSortKm);
177       add (new Label("Dateiname:"));
178       add (tfDateiname);
179       add (btSpeichern);
180       add (btLaden);
181       add (new Label(" "));
182       add (btZeigen);
183       add (taAusgabe);
184       setVisible(true);
185     }
186   }
```

```
1   //Kfz-Verwaltung, Datei 4/4
2   //Java JDK 1.1.18 -- Krell 21.2.2002
3   //Definiert, erzeugt und startet ein Objekt der Klasse KfzGUI
4
5   public class KfzStart extends java.applet.Applet
6   {
7     //Start als selbständige Application
8     public static void main (String[] s)
9     {
10      KfzGUI autoVerw = new KfzGUI();
11      autoVerw.führeAus();
12    }
13
14    //Start als Applet aus dem Web-Browser
15    public void init()
16    {
17      KfzGUI autoVerw = new KfzGUI();
18      autoVerw.führeAus();
19    }
20  }
```

Auf der nächsten Seite folgt noch ein Bildschirmabdruck der grafischen Oberfläche des Programms:


Fuhrpark-Verwaltung
Amtliches Kennzeichen (*,^):
D-AB 123
Fahrzeughersteller (%):
Modell:
Baujahr:
Aktueller Stand (*):
42578
km.
Auto neu aufnehmen
km-Stand eines vorhandenen Autos ändern (*)
Auto löschen (^)
Alle Autos eines Herstellers (%) finden
Autos nach Nummer sortieren
Autos nach km sortieren
Dateiname:
KfzDatei.Dat
Fuhrpark speichern
Fuhrpark laden
Fuhrpark zeigen
1.) Das Auto mit der Nr. 'D-BC 234', ein Ford Ka von 2000, ist bisher 12005 km gelaufen.
2.) Das Auto mit der Nr. 'D-AB 123', ein VW Polo von 1998, ist bisher 42361 km gelaufen.
3.) Das Auto mit der Nr. 'D-XY 1256', ein Opel Astra von 1997, ist bisher 78265 km gelaufen.
4.) Das Auto mit der Nr. 'D-A 1256', ein Ford Focus von 1999, ist bisher 38912 km gelaufen.