Hochschule Harz	**FB Automatisierung und Informatik**
Versuch: E/A mit java	Betriebssysteme WI/MI Thema: Dateioperationen mit Java zum Einlesen einer dBase-Datei

Versuchsziele

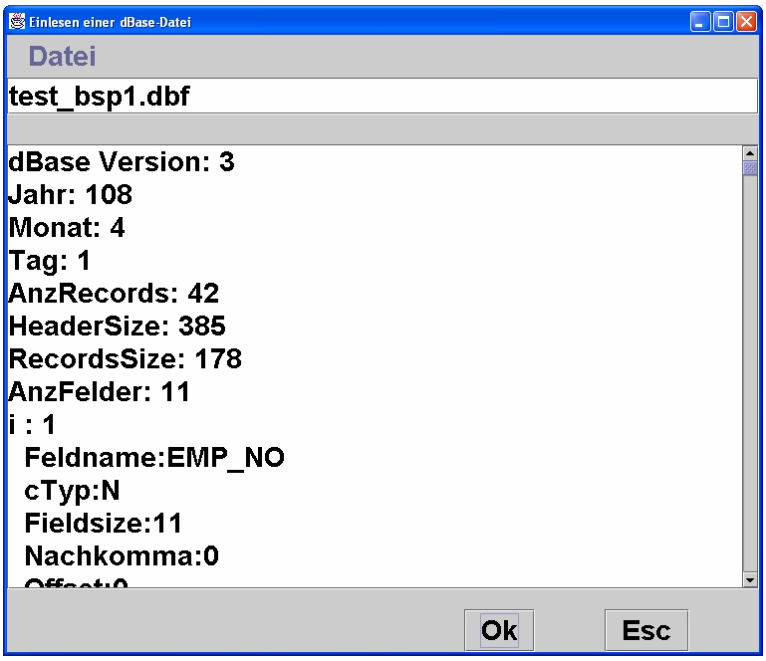
Vertiefung im Verständnis der einfachen Dateioperationen mit Java. Eingelesen wird eine binäre dBase-Datei.

Aufgabenstellung:

Entwickeln Sie ein Java-Programm, welches die Struktur einer dbasse-Datei analysiert und mittels eines Editors vereinfacht ausgibt.

Voraussetzung:

Die Dokumentation des internen Formats einer dBase-Datei ist bekannt (siehe Homepage).



GUI-Oberfläche

Übersicht über das Java-Programm

```
public class Read_dBase extends JFrame {

   // globale GUI-Elemente
   JTextField Editzeile;
   JTextArea Editor;

   //Frame konstruieren
   public Read_dBase() {
      setSize(1000, 700);
      setTitle("Einlesen einer dBase-Datei");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setGUI();
   } // create


   // setzen der GUI-Elemente, vollständig
   void setGUI() {
      …
   }



   // wird ausgelöst beim Drücken des Schalters ESC
   void BnEsc_Click() {
      System.exit(0);
   }



   public static void main(String[] args) {
      Read_dBase frame = new Read_dBase();
      frame.setVisible(true);
   }

   // testet, ob die Datei vorhanden ist
   public boolean FileExists(String sFileName) {
      File f = null;
      try {
         // evnetuell kann auch f.pathSeparatorChar verwendet werden
         f = new File(sFileName);
         if (f.exists())
            return true;
         else
            return false;
         }
      catch (Exception e) {
         return false;
      }
   }  // FileExists
```

```
 // wird ausgelöst beim Drücken des Schalters Ok
 // hier fehlt etwas
   void BnOk_Click() {
      String NL = "\n\r";
      FileInputStream Fin;
      DataInputStream Din;
      String sFilename;
      dBase_Header header;
      dBase_Feld felder[];
      int i, j, k;
      int i1,i2,i3,i4;
      String sStr;
      char ch;

      // Test, ob Datei vorhanden ist
      Editor.setText("");  // Editorinhalt löschen
      /*
      try {
      }
      catch (IOException ee) {
         System.err.println("IOException: " + ee);
      }
      */
   }  // BnOk_Click

}


// Klasse fuer den Header
// oeffentliche Variablen, vereinfacht
class dBase_Header {
   public int Version;  // byte
   public int jahr;  // byte
   public int monat;  // byte
   public int tag;  // byte
   public int anzRecords;   // Anzahl der Datensaetze
   public int headersize;   // WORD Headergroesse in Byte
   public int recordsize;   // Word Laenge eines einzelnen Datensatzes */
   private byte  dummy[] = new byte[20];
   public int AnzFelder;

   // liest den Header
   public void loadFromFile(DataInputStream Din) {
      byte b, b1, b2, b3, b4;
      int i1,i2;
       /*
      try {
      }
      catch (IOException ee) {
         System.err.println("IOException: " + ee);
      }
      */
   }

} // dBase_Header
```

```
// Klasse fuer einen Feldeintrag, keine Daten
//   nur Definition
class dBase_Feld {
   public String FeldStr;      // Feldname   aber nur Grossbuchstaben  11 Zeichen
   public char cTyp;           //Feldtyp  C=String; N=Zahl; D=Datum; L=Boolean
   private byte  dummy1[] = new byte[4];

   public int  fieldsize;          // Feldlaenge byte
   public int  nk;             // Nachkommastellen byte
   private byte  dummy2[] = new byte[14];
   public  int offset;  // speichert den Offset jedes Feldes, eventuell zum zurückschreiben

   // liest die definition eines Feldes
   public void loadFromFile(DataInputStream Din) {
      byte b, b1, b2;
      char ch;
      int i, i1,i2;
       /*
      try {
      }
      catch (IOException ee) {
         System.err.println("IOException: " + ee);
      }
      */
   }

} // dBase_Feld
```

Versuchsdurchführung

1) BnOk_Click()

- Die Methode testet, ob die Datei vorhanden ist und öffnet die Datei zum Lesen.
- Diese Methode liest den Header und die Felder nur indirekt aus.
- Danach werden dann die einzelnen Datensätze gelesen.
- Als erstes sollte nur das Einlesen des Headers entwickelt und getestet werden
- Als zweites wird das Einlesen der Felder entwickelt und getestet werden

2) Klasse dBase_Header, Methode loadFromFile

Diese Funktion liest den Header aus der dBase-Datei und speichert die Werte als lokale Variable

3) Klasse dBase_Feld, Methode loadFromFile

Diese Funktion liest genau EIN Feld aus der dBase-Datei und speichert die Werte als lokale Variable

Ergebnis der Musterlösung:

dBase Version: 3
Jahr: 108
Monat: 4
Tag: 17

AnzRecords: 30
HeaderSize: 225
RecordsSize: 124
AnzFelder: 6

i : 1
 Feldname:NUMMER
 cTyp:C
 Fieldsize:30
 Nachkomma:0
 Offset:0

i : 2
 Feldname:KREIS
 cTyp:C
 Fieldsize:30
 Nachkomma:0
 Offset:0

i : 3
 Feldname:GEWNAME
 cTyp:C
 Fieldsize:30
 Nachkomma:0
 Offset:0

i : 4
 Feldname:GEWORDNUNG
 cTyp:N
 Fieldsize:16
 Nachkomma:0
 Offset:0

i : 5
 Feldname:LAENGE
 cTyp:N
 Fieldsize:16
 Nachkomma:3
 Offset:0

i : 6
 Feldname:KATALOG
 cTyp:L
 Fieldsize:1
 Nachkomma:0
 Offset:0

Datensatz:1

01
SAW
Uchte
1
84440.288
true

Datensatz:2

01-02
SAW
Jeetzel Graben
1
62683.380
false

Datensatz:3
01-03
SAW
Aller
1
47236.219
true

Datensatz:4
01-04
MD
Obere Ohre
1
36576.825
true

Datensatz:5

01-05
JL
Untere Ohre
1
123299.518
false

Datensatz:6
01-02-06
SAW
Jeetze Graben
2
27582.797
true

Binäres Einlesen

Verwendet werden die Klassen FileInputStream und DataInputStream
Da Fehler auftreten können, werden die Lese-Operationen mit einer Exception geschützt.

```
try {
      // Verknüpfung mit der Datei
   FileInputStream Fin = new FileInputStream(sFilename);
      // Umwandlung in eine Daten-orientierte Verbindung
   DataInputStream Din = new DataInputStream(Fin);
      // Einlesen eines double-Wertes
   d = din.readDouble();
      // Einlesen eines int-Wertes
   i = din.readInt();
   d = din.readDouble();
   din.close();
}
catch (IOException e) {
   System.err.println("IOException: " + e);
}
```