Felder

Der Themenbereich *Algorithmen und Datenstrukturen* gehört zu den klassischen Themen der Informatik. Wir untersuchen verschiedene typische Lösungsstrategien, Standardalgorithmen und abstrakte Datentypen und lernen dabei die Methoden und Konzepte kennen, die für die systematische und erfolgreiche Entwicklung von Softwaresystemen nötig sind. Wir legen los mit ***Feldern***.

Ein Feld (engl. array) ist eine Reihe von Elementen gleichen Datentyps. Die einzelnen Elemente können über einen Index angesprochen werden. In Felder kann man eine begrenzte Anzahl gleichartiger Daten speichern. Im Gegensatz zu ***einfachen Datentypen*** wie int, double, char oder boolean hat ein Feld einen ***strukturierten Datentyp***. Die Struktur eines Feldes (hier mir dem Namen wert) wird im Bild beispielhaft deutlich.

wert[0]	wert[1]	wert[2]	wert[3]	...	wert[i]	wert[i+1]	...	wert[n-1]	wert[n]
17	64	4512	199		46				

Wir sehen ein Feld ganzer Zahlen der Länge n. Jede Zahl befindet sich in einer Zelle des Feldes. Alle Zellen sind gleich groß und können genau eine ganze Zahl aufnehmen. Nicht alle Zellen des Feldes sind belegt, es können noch weitere Zahlen im Feld gespeichert werden. Das erste Feldelement ist 17 und hat den Index 0 (Merke: Computer zählen ab Null), das zweite Element ist 64 und kann über den Index 1 angesprochen werden, das letzte belegte Feldelement ist 46. Daher ist es sinnvoll die Anzahl der belegten Felder zu speichern.

Deklariert werden Felder durch Kennzeichnung der Feldvariablen mit dem Klammernpaar []. In der Variablen ***anzahlElemente*** merken wir uns, wie viele Elemente im Feld tatsächlich gespeichert sind. Diese Variable braucht man nicht, wenn das Feld immer komplett gefüllt ist.

```
int[] werte;
int anzahlElemente;
```

Zum Anlegen eines Feldes benötigt man die ***new***-Methode. Ein Feld ist der einzige Datentyp, der auf diese Art initialisiert werden muss, da Felder in Java Objekte sind (im Gegensatz zu anderen Programmiersprachen). Im Beispiel wird das Feld ***werte*** zur Aufnahme von maximal 100 Integerzahlen angelegt.

```
werte = new int[100];
```

Das Deklarieren und Initialisieren kann auch gleichzeitig erfolgen.

```
int[] werte = new int[100];
```

Auf ein einzelnes Feldelement greift man über den Index zu. Im Beispiel wird dem Feldelement mit dem Index 7 die Zahl 15 zugewiesen, danach werden allen Zellen mit den Indizes 7 bis 19 Zufallszahlen von 0 …49 zugewiesen und ausgegeben.

```
werte[7] = 15;
for (int i= 7; i < 20; i++) {
  werte[i]= (int) Math.floor(50*Math.random());
  System.out.println(werte[i]);
}
```

Die Kapazität eines Feldes ermittelt man mit der Methode length:

```
System.out.println("Maximales Fassungsvermögen: " + werte.length);
```

Einfügen, Suchen und Löschen eines Elements, sowie Anlegen, Füllen und Sortieren eines Feldes sind Standardoperationen auf Feldern. Mit einem Java-Programm sollen diese Operationen realisiert werden.

Aufgabe 1

Schreibe ein Konsolenprogramm, das ein Feld der Kapazität 20 mit einer gewünschten Anzahl Zufallszahlen füllt und dann folgendes ausgibt: alle Zahlen, die größte Zahl, die kleinste Zahl, die Summe, den Mittelwert der Zahlen.

Schaffst du es auch die Zahl auszugeben, die dem Mittelwert am nächsten kommt?

Die einzelnen Aufgaben sollten in geeigneten Methoden bearbeitet werden.

Aufgabe 2

Schreibe ein Konsolenprogramm das ein Feld mit 100 ganzzahligen Zufallszahlen von 1 und 9 füllt und dann die Häufigkeitsverteilung dieser Zahlen ermittelt und ausgibt.

Aufgabe 3

Tauziehen

In einem Feld der Länge 31 soll ein „Tauziehen“ stattfinden.

- Zu Beginn steht der „Zeiger“ in der Mitte.
- Abwechselnd werden zufällig 1, 2 oder 3 Schritte nach links oder rechts gemacht.
- Beendet ist das Spiel, wenn das linke oder rechte Ende erreicht ist.
- Nach jedem Zug wird das aktuelle Bild ausgegeben.

AWT / SWING – Programmierung

Als Konsolenprogramm lassen sich die Grundoperationen auf Feldern schlecht darstellen. Wir erstellen daher mit den AWT (Abstract Window Toolkit) – Komponenten eine GUI (Graphical User Interface) – Anwendung. Die grafische Oberfläche des späteren Programms soll wie in folgendem Bild aussehen:

Im oberen Teil des Fensters sind 15 Textfelder platziert, in denen später die Werte des Feldes angezeigt werden sollen. Da es mühsam ist, 15 einzelne Textfelder zu verwalten, sollen diese beim Programmstart erzeugt werden. Dies gibt uns auch die Möglichkeit, die Textfelder mit einem Datentyp Feld zu verwalten. Im unteren Teil des Fensters befinden sich die zur Arbeit notwendigen Buttons. Rechts daneben gibt es noch ein weiteres Textfeld, um eine Zahl bequem ein- bzw. ausgeben zu können.

Der Java-Editor bietet eine einfache Möglichkeit, GUI-Applikationen zu erstellen. Über den Button *JFrame* der Komponentenleiste *Programm* wird eine solche automatisch erzeugt.

> *Hinweis: Java bietet sowohl AWT als auch SWING als Bibliotheken zum Erstellen grafischer Oberflächen an. AWT war von der Firma SUN bewusst knapp gehalten und bietet daher auch nur wenige Elemente an. Die SWING-GUI-Komponenten wurden später innerhalb des Toolkits JFC programmiert. Allerdings bezieht sich SWING nur zum Teil auf AWT, d. h. es besteht hier keine Kompatibilität. Was heißt das für uns? Entweder wir arbeiten komplett mit Frame und AWT oder komplett mit JFrame und SWING ...*

Unser automatisch erzeugtes Programm *Felder* hat nun folgendes Aussehen:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class Felder extends JFrame {
  // Anfang Attribute
  // Ende Attribute

  public Felder(String title) {
    // Frame-Initialisierung
    super(title);
    setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
    int frameWidth = 300;
    int frameHeight = 300;
    setSize(frameWidth, frameHeight);
    Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
    int x = (d.width - getSize().width) / 2;
    int y = (d.height - getSize().height) / 2;
    setLocation(x, y);
    Container cp = getContentPane();
    cp.setLayout(null);
```

```
    // Anfang Komponenten
    // Ende Komponenten

    setResizable(false);
    setVisible(true);
  }

  // Anfang Methoden
  // Ende Methoden

  public static void main(String[] args) {
    new Felder("Felder");
  }
}
```

Über die import-Direktiven (zu Beginn des Quelltextes) haben wir direkten Zugriff auf die AWT- und SWING-Komponenten. Die Klasse ***Felder*** wird von der Frame-Klasse abgeleitet (**extends** Frame). Ein ***JFrame*** ist ein Windows-Fenster mit Titelleiste und Systemschaltern zum Minimieren, Wiederherstellen und Schließen und entspricht einem Programmformular in Delphi. Die Kommentare `// Anfang Attribute` und `// Ende Attribute` **müssen** unangetastet bleiben, weil der Java-Editor diese Kommentare benutzt, um in diesen Bereich Variablendeklarationen für AWT- und SWING-Komponenten unterzubringen. Analoges gilt für die Kommentare `//Anfang/Ende Komponenten` sowie `// Anfang/Ende Methoden`.

Die Hauptmethode `main` erzeugt mittels `new Felder("Felder")` das Programmformular. Der sogenannte Konstruktor `Felder`, mit dem das Formular erzeugt wird, setzt mittels `super(titel)` den Fenstertitel, erstellt eine Ereignisprozedur für das Schließen des Fensters, setzt die Fenstergröße auf 300 x 300 und erstellt ein Panel (cp) für den Inhalt des Fensters. Ab `// Anfang Komponenten` werden die Komponenten in das Programm eingefügt und die gewünschte Benutzungsoberfläche aufgebaut.

Die Vorlage kann nun nach eigenen Wünschen geändert werden, sowohl im Quelltext als auch im GUI-Designer des Java-Editors. Änderungen werden jeweils im anderen Teil automatisch übernommen.

Mit dem GUI-Designer können die Schalter im unteren Bereich erzeugt werden. Setzt man einen Schalter *JButton* auf ein Formular, so wird in den drei Abschnitten *Attribute*, *Komponenten* und *Methoden* entsprechender Quellcode erzeugt. In der Variablendeklaration wird der Schalter *Neu* vom Typ *JButton* deklariert und mittels `new JButton();` erzeugt.

```
  // Anfang Attribute
  private JButton btnNew = new JButton();
```

Im Abschnitt Komponenten werden die Größe und Beschriftung gesetzt, der Schalter dem Programm-Panel cp (content-panel) zugeordnet und eine Ereignisprozedur für das Anklicken des Schalters implementiert.

```
    btnNew.setBounds(8, 96, 80, 25);
    btnNew.setText("Neu");
    btnNew.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent evt) {
        btnNew_ActionPerformed(evt);
      }
    });
    cp.add(btnNew);
```

Die Ereignisprozedur ruft die Methode ***btnNew_ActionPerformed*** auf, welche im Abschnitt `// Methoden` erzeugt wird. In dieser Methode wird programmiert, was beim Anklicken des Schalters *Neu* gemacht werden soll:

```
// Anfang Methoden
public void btnNew_ActionPerformed(ActionEvent evt) {

}
```

Aufgabe 3

Erstellen Sie nach obigem Vorbild die GUI mit den benötigten Buttons.

Erzeugen der Textfelder

Es wäre mühsam, die fünfzehn JTextField-Komponenten im oberen Teil des Formulars manuell anzulegen. Einfacher geht es mit einer Zählschleife, die die JTextField-Komponenten erzeugt und auf dem Formular platziert.

```
for (int i=0; i<15; i++) {
  JTextField einTextFeld = new JTextField("");
  einTextFeld.setBounds(10 + i*35, 50, 35, 25);
  add(einTextFeld);
}
```

Der Nachteil dieses Ansatzes ist, dass wir nach der Anzeige im GUI-Formular keinen Zugriff mehr auf die Textfelder haben. Die Variable *einTextFeld* ist nur in lokal der Schleife deklariert und kann auch nur ein Textfeld bezeichnen. Zur Verwaltung von 15 Textfeldern brauchen wir ein Feld, wobei jedes Element des Feldes ein Textfeld ist, sozusagen ein Feld aus Textfeldern:

```
// Anfang Attribute
private JTextField[] textFeld = new JTextField[15];
```

Die Zählschleife wird ergänzt um die einzelnen Textfelder im Feld für den späteren Zugriff zu speichern:

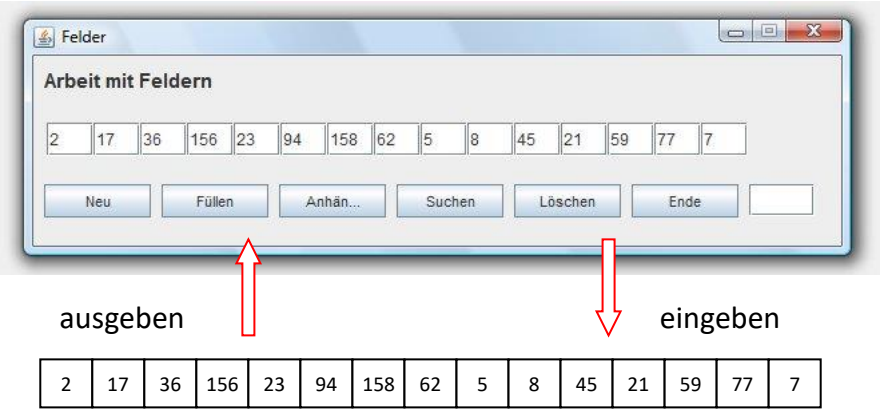
```
for (int i=0; i<15; i++) {
  JTextField einTextFeld = new JTextField("");
  einTextFeld.setBounds(10 + i*35 , 50, 35, 25);
  add(einTextFeld);
  textFeld[i] = einTextFeld;
}
```

Aufgabe 4

Erstellen Sie nach obigem Vorbild die GUI mit den restlichen benötigten Komponenten.

Trennung von Daten und ihrer Darstellung

Die heutigen grafischen Benutzungsoberflächen sind darauf eingerichtet, die Daten eines Programms möglichst flexibel darstellen zu können. Es gibt hochspezialisierte Komponenten, zum visuellen Aufbereiten von Daten. Zum Beispiel lassen sich in Excel Zahlen mit oder ohne Tausenderpunkte, als Währung (€ oder DM) oder gar als Kalenderdatum darstellen. Wir benutzen TextField-Komponenten, um Zahlen eines Feldes darzustellen. Mit der eigentlichen Verarbeitung der Daten haben die visuellen Komponenten in der Regel nichts zu tun. Sie beschränken sich im Wesentlichen auf die Ein- und Ausgabe der Daten. Die Verarbeitung findet auf der internen Datenebene statt. Auf dieser Ebene befinden sich unsere Daten in einem int-Feld.



Für die bequeme Arbeit mit diesen beiden Ebenen ist es sinnvoll, Methoden zu schreiben, die für den Datenaustausch sorgen. Beispielsweise gibt eine Methode ***ausgeben()*** alle Zahlen des internen Feldes in den TextField-Komponenten aus. Die Methode ***eingeben()*** liest aus dem Textfeld rechts neben dem Beenden-Schalter eine Zahl:

```
public int eingeben() {
  // Gibt den Wert im Eingabefeld zurück
  return Integer.valueOf(tfInput.getText());
}
```

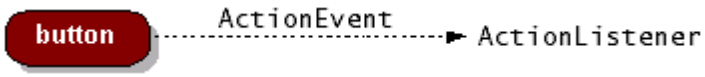
Es ist eine parameterlose Funktion, die einen int-Wert liefert, also einem Wert, der auf der Datenebene sofort verarbeitet werden kann. Bei der Arbeit mit dieser Funktion stellt man fest, dass sie empfindlich gegenüber Fehleingaben wie z. B. einem leeren Eingabefeld, unzulässigem Zahlenformat oder nicht numerischen Zeichen in der Eingabe ist. Es erscheinen so genannte ***NumberFormatExceptions*** (engl. exception = Ausnahme). Zum Abfangen solcher Eingabefehler setzt man die *try*-Anweisung ein. Sie sorgt dafür, dass auftretende Fehler erkannt und sinnvoll behandelt werden können.

```
public int eingeben() {
  // Gibt den Wert im Eingabefeld zurück
  try {
    return Integer.valueOf(tfInput.getText());
  } catch (NumberFormatException e) {
    return -1;
  }
}
```

Schalter - Ereignis - Ereignismethode

Beim Anklicken eines Schalters wird ein Ereignis (engl. event) ausgelöst, dass zur Unterscheidung von anderen Ereignissen ***ActionEvent*** genannt wird. Da der Anwender natürlich keinen realen Schalter klickt, sondern irgendwo auf den Bildschirm, auf dem verschiedene Fenster, möglicherweise sich verdeckend, angezeigt werden, muss Windows aus der geklickten Bildschirmposition das Programm ermitteln, das für den Bildschirmbereich zuständig ist. Es schickt dann das Klickereignis an das zuständige Programm.

Bei einem Java-Programm müssen sich die Objekte, die auf Ereignisse reagieren, anmelden. Ein Schalter erledigt dies mittels ***addActionListener.*** Der dabei angegebene ***ActionListener*** enthält eigentlich nur die Ereignismethode ***actionPerformed***, welche beim Anklicken ausgeführt wird. Für einen leichteren Einstieg in die Java-Programmierung erzeugt der Java-Editor



den benötigten Code automatisch:

```
btnNew.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    btnNew_ActionPerformed(evt);
  }
});
```

Die automatisch erzeugte Ereignismethode ***actionPerformed*** ruft eine Methode auf, die im Bereich `// Methoden` deklariert wird. Beim Neu-Schalter sieht die Implementierung dieser Methode wie folgt aus:

```
// Anfang Methoden
public void btnNew_ActionPerformed(ActionEvent evt) {

}
```

Aufgabe 5

Das Programm Felder soll vervollständigt werden:

- Definieren Sie die notwendigen Variablen für ein Feld von Integer-Zahlen der Kapazität 15 und einem Zählen für die Anzahl der derzeitigen Elemente im Feld.
- Implementieren Sie eine Methode *ausgeben()*, welche die im Feld gespeicherten Zahlen in die Textfelder schreibt.
- Der Button New soll den Inhalt aller Textfelder löschen.
- Der Button *Füllen* soll eine bestimmte Anzahl (<16, steht im Eingabefeld) der Textfelder mit Zufallszahlen füllen.
- Der Button *Anhängen* soll die im Eingabefeld stehende Zahl in das erste freie Textfeld schreiben.
- Der Button *Suchen* testet, ob die im Eingabefeld stehende Zahl in den Textfeldern vorkommt.
- Der Button *Löschen* löscht alle Zahlen in den Textfeldern, die mit der Zahl im Eingabefeld übereinstimmen.
- Der Button Ende beendet das Programm.