Felder (Arrays) in Java

In der letzten Doppelstunde stellten wir fest, dass es bei unserem kleinen Malprogramm in Java ein größeres Problem gab. Wenn man die Größe des Fensters änderte oder das Fenster durch andere verdeckt wurde, war die Zeichnung verschwunden. Mit Hilfe der Konzepte, die heute behandelt werden, kann eine Lösung dafür gefunden werden:
Um die Zeichnung zu speichern, müssen alle gezeichneten Endpunkte gespeichert werden. Allgemein heißt dies, dass ein Konzept gesucht wird, dass eine (feste) Anzahl von Variablen mit dem gleichen Datentyp gespeichert werden soll. Dafür verwendet man in der Informatikk sogenannte **Felder** bzw. **Arrays**.

Beispiele:

a) Wenn die durchschnittliche Höchsttemperatur in einem Monat bestimmt werden soll, ist es ungeschickt 31 double-Variablen `temp1, temp2, ... , temp31` zu verwenden. Geschickter ist ein Feld, da man dessen Elemente sehr geschickt in Schleifen bearbeiten kann.
b) Ein typisches Beispiel für Felder in der Mathematik sind Vektoren.

Eigenschaften eines Feldes (Arrays)

n Elemente $e_0, e_1, \dots, e_{n-1}$ sollen in einem Feld gespeichert werden:

e_0	e_1	...	e_{n-1}

- Alle Daten sind vom gleichen Datentyp
- Die einzelnen Elemente werden mit Hilfe eines ganzzahligen Index („Platznummer“) angesprochen
- Der Index in Java läuft von 0 bis Feldlänge-1

Definition von Feldern

Felder werden in Java folgendermaßen deklariert: `Datentyp[] Variablenname;`
Damit hat man eine Variable, die ein Feld vom angegebenen Datentyp enthalten kann. Das Feld erzeugt man durch `Variablenname=new Datentyp[AnzahlElemente];`

`Datentyp` kann jeder beliege Datentyp sein, also auch Double, String usw.
Nach der Definition eines Feldes ist seine Größe festgelegt, d.h. sie kann nachträglich nicht mehr geändert werden.

Beispiel:

Definition eines Feld aus 31 Double-Elementen mit dem Variablennamen Temperatur:

```
double[] Temperatur;
Temperatur=new double[31];
```

oder kurz: `double[] Temperatur=new double(31);`

Wenn man das Feld gleich mit von 0 verschiedenen Werten initialisieren möchte, geschieht dies durch z.B.:

```
int[] gerade = {2,4,6,8,10,12};
String[] namen = {"Michael","Maike","Jörg","Benjamin","Sebastian");
```

Zugriff auf Elemente eines Feldes

Um auf die einzelnen Eintragungen, also die Daten (auch *Komponenten*) eines Feldes zugreifen zu können, werden die Plätze, an denen die Daten stehen, nummeriert. Der erste Platz im Feld hat, wie oben angedeutet, die Nummer „0“, der zweite die Platznummer „1“ und der letzte Platz in einem Feld aus n Elementen hat die Platznummer „(n-1)“. Statt Platznummer verwendet man in der Informatik in diesem Fall den Ausdruck „*Index*“.

Auf die Elemente eines Feldes kann man mit `Variablenname[Index]` zugreifen. Wenn Index außerhalb von Feld-Anfang und Feld-Ende liegt, erhält man in der Regel eine Fehlermeldung.

Beispiele:

Mit Hilfe von `gerade[3]` liest man die Zahl 8 aus dem obigen Feld. namen[0] liefert „Michael“ zurück.
Nach der Anweisung `zahl = gerade[4];` har die Variable zahl vom Typ int den Wert 10.
Einer Komponente eines Feldes weist man den Wert durch

```
Temperatur[3]=25.4;
```

zu. Dadurch wird der vorherige Wert an der Position mit dem Index 3 überschrieben.

Beispiel: Berechnung der Summe von 5 Zahlen

```
import Prog1Tools.IOTools;

public class Summe{

  public static void main(String[] args){
    // Anzahl der zu speichernde Elemente festlegen
    int anzahl=5;  // (1)
    // Feld deklarieren
    double[] zahlen;
    // Feld erzeugen
    zahlen=new double[anzahl];
    // Zahlen einlesen
    for(int i=0;i<anzahl;i++){  // (2)
      zahlen[i]=IOTools.readDouble((i+1)+". Zahl eingeben: ");
    }
    // Zahlen summieren
    double summe=0;
    for(int i=0;i<anzahl;i++){ // (3)
      summe=summe+zahlen[i];
    }
    // Summe ausgeben
    System.out.println("Summe = "+summe);
  }
}
```

Im obigen Beispielprogramm wird eine wichtige Technik beim Umgang mit Feldern verwendet. Man definiert nur an einer Stelle, wie bei (1), in einer Variablen oder Konstanten die Größe des Feldes und arbeitet im Rest des Programms (z.B. bei (2) oder (3)) nur mit dieser Variablen und nicht mit konkreten Zahlenwerten.
Felder sind in Java Objekte. Deshalb kann man alternativ dazu auch die Eigenschaft `length` des Feldes nutzen, d.h. (2) und (3) würden so aussehen:

```
for(int i=0;i<zahlen.length;i++){
```

Aufgaben:

1. Ändere das obige Beispielprogramm so ab, dass man beliebig viele Zahlen addieren kann. Dazu muss zu Programmbeginn der Benutzer gefragt werden, wie viele Werte er eingeben will.
2. Schreibe ein neues Programm, basierend auf dem aus Aufgabe 1, das jetzt das Produkt der eingegeben Zahlen bestimmt werden soll.
3. Schreibe ein neues Programm, basierend auf dem aus Aufgabe 1, das den arithmetischen und geometrischen Mittelwert der eingegeben Zahlen bestimmt.