Arrays

Gilbert Beyer und Annabelle Klarl

Zentralübung zur Vorlesung Einführung in die Informatik

`http://www.pst.ifi.lmu.de/Lehre/wise-11-12/infoeinf`

Inhalte der heutigen Vorlesung:

- Arraytypen
- Speicherdarstellung von Arrays
- Auswertung von Arrays
- Zugriff auf Arrayelemente
- Erzeugung von Arrays
- Initialisierung von Arrays
- Zuweisungen und Arrays
- Algorithmen auf Arrays

Aufgabe 1)

Implementieren Sie eine Methode average zur Berechnung des Arithmetisches Mittels eines Arrays.

Arithmetisches Mittel

```
public static double average(int[] daten) {
//Array daten ist mit mind. 1 Element initialisiert
  int sum = 0;
  for (int i = 0; i < daten.length; i++)
    sum = sum + daten[i];

  return (double) sum / daten.length;

}
```

Aufgabe 2)

Es soll geprüft werden, ob ein Array sortiert ist.

Implementieren Sie eine Methode isSorted.

Ihr Algorithmus soll dabei das Array durchlaufen und prüfen,

ob ein Element grösser oder gleich seinem linken Nachbarn ist.

Prüfen auf Sortierung

```
static boolean isSorted(double[] arr) {
  for (int i = 1; i < arr.length; i++) {
    if (arr[i - 1] > arr[i]) {
      return false;
    }
  }
  return true;
}
```

Aufgabe 3)

Implementieren Sie die Methode equalArray, die prüfen soll, ob zwei Arrays gleich sind. Zwei Arrays sind genau dann gleich, wenn sie die gleiche Länge haben und für alle $0 \leq i \leq n-1$ gilt, dass das Element an der i-ten Stelle im ersten und zweiten Array gleich ist.

Prüfen auf Gleichheit

```
static boolean equalArray(double[] a1, double[] a2) {
    if (a1.length != a2.length) {
      return false;
    }
    for (int i = 0; i < a1.length; i++) {
      if (a1[i] != a2[i]) {
        return false;
      }
    }
    return true;
  }
```

Aufgabe 4)

Unter einer Matrix versteht man eine rechteckige Anordnung (Tabelle) von Elementen bzw. mathematischen Objekten, mit denen man rechnen (z. B. addieren, multiplizieren) kann.

Wie können eine Matrix und Berechnungen mit Matrizen in Java realisiert werden?

zweidimensionale Arrays

```
public class Matrix {
  public static void main(String[] args) {
    int[][] elements = new int[10][8];
    for (int i = 0; i < 10; i++)
      for (int j = 0; j < 8; j++)
        elements[i][j] = (int) Math.pow(i + 1, j + 1);
  }
}
```

zweidimensionale Arrays

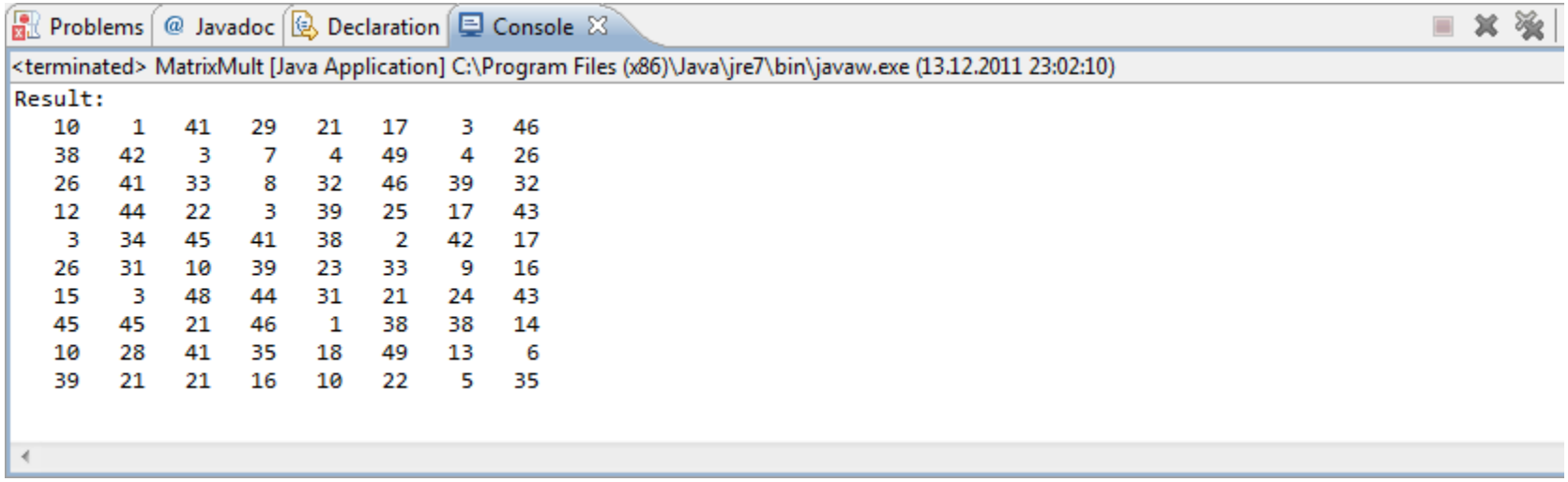
```
import java.util.Random;

public class Matrix {
  public static void main(String[] args) {
    Random rnd = new Random();
    int[][] elements = new int[10][8];
    for (int i = 0; i < 10; i++)
      for (int j = 0; j < 8; j++)
        elements[i][j] = rnd.nextInt(50);
  }
}
```

Aufgabe 4b)

Erweitern Sie das Programm um eine Methode printMatrix, welche Matrizen auf dem Bildschirm ausgeben kann. Damit die Spalten der ausgegebenen Matrix alinieren, benötigen Sie eine weitere Methode formatOutput, die jeden Zahlwert in einen String konvertiert und solange zusätzliche Lehrzeichen hinzufügt, bis der String die gewünschte Spaltenbreite erreicht.

Ausgabe von Matrizen



Ausgabe von Matrizen

```
public static void printMatrix(int[][] matrix, int width) {
    for (int zeile = 0; zeile < matrix.length; zeile++) {
      for (int spalte = 0; spalte < matrix[zeile].length;
            spalte++) {
        System.out.print(formatOutput(
            matrix[zeile][spalte], width));
      }
      System.out.println(); // Zeilenumbruch
    }
  }
```

Ausgabe von Matrizen

```
public static String formatOutput(int n, int width) {
   String formattedString = "" + n;

   while (formattedString.length() < width) {
     formattedString = " " + formattedString;
   }

   return formattedString;
 }
```

Aufgabe 4c)

Erweitern Sie obiges Programm so, dass es eine Matrix mit einer Einheitsmatrix kompatibler Größe multipliziert, und das Ergebnis auf dem Bildschirm ausgibt.

Multiplikation von Matrizen

Zwei Matrizen $A = (a_{ij})_{i=1...l,\ j=1...m}$ und $B = (b_{ij})_{i=1...m,\ j=1...n}$ werden nach folgender Produktsummenformel auf Paare aus einem Zeilenvektor der ersten und einem Spaltenvektor der zweiten Matrix multipliziert:

$$A \cdot B = (c_{ij})_{i=1...l,\ j=1...n} \text{ und } c_{ij} = \sum_{k=1}^{m} a_{ik} \cdot b_{kj}$$

Multiplikation von (kompatiblen) Matrizen

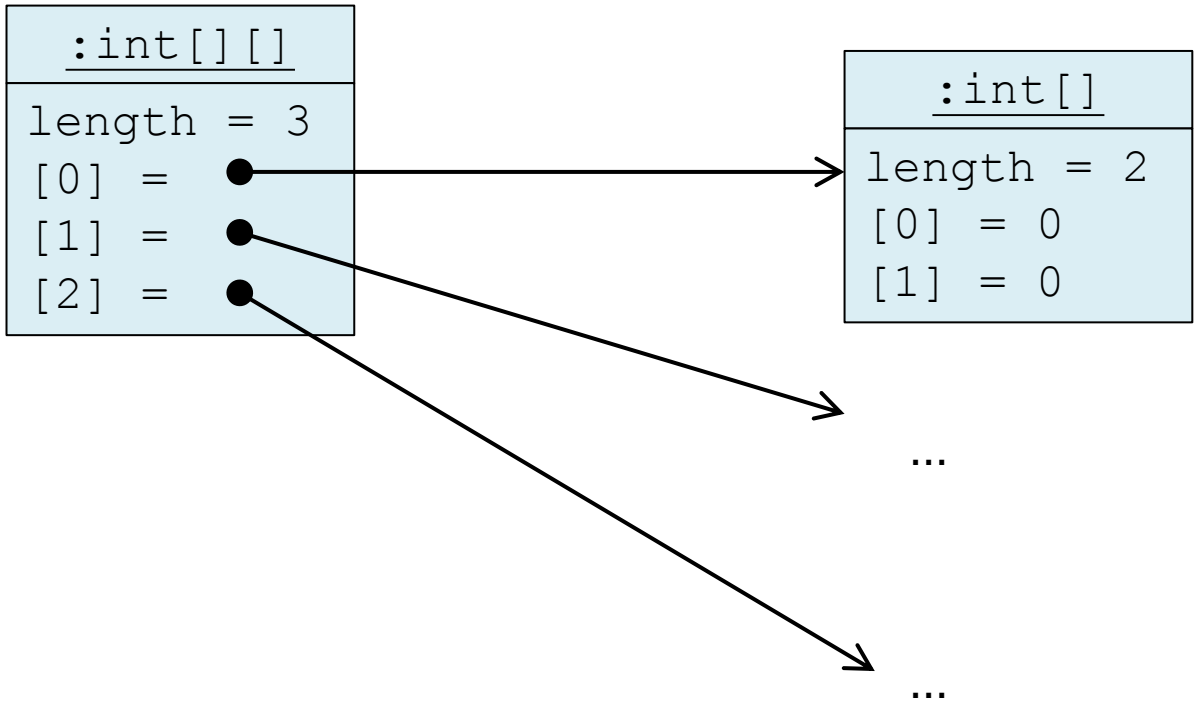
```
public static int[][] matrixMult(int[][] a, int[][] b) {
   if (a[0].length != b.length) {
     // a hat nicht genausoviele spalten wie b zeilen --> mult nicht möglich
     return null;
   }

   int[][] ergebnis = new int[a.length][b[0].length];

   for (int zeile = 0; zeile < a.length; zeile++) {
     for (int spalte = 0; spalte < b[0].length; spalte++) {
       int zwischenSumme = 0;
       for (int i = 0; i < b.length; i++) {
         zwischenSumme += a[zeile][i] * b[i][spalte];
       }
       ergebnis[zeile][spalte] = zwischenSumme;
     }
   }
   return ergebnis;
}
```

Initialisierung von Arrays

```
int [][] a = new int[3][2];
```



Initialisierung von Arrays

```
int [][] b = new int[3][];
```

:int[][]
length = 3 [0] = null [1] = null [2] = null

Kopieren von Arrays

Zum vollständigen Kopieren von Arrays kann man eine der beiden folgenden Varianten nehmen:

```
int a[] = {1, 2, 3, 4, 5};
int b[] = null;
int c[] = new int[5];


b = a.clone(); //public Object clone()
System.arraycopy(a, 0, c, 0, a.length);
```

Kopieren von Arrays

```
double[] data = { 1, 2, 3 };
  double[] prices = new double[data.length];
  for (int i = 0; i < data.length; i++) {
    prices[i] = data[i];
  }
```

Dynamisch wachsende Arrays

```
double[] data = new double[DATA_LENGTH];
int dataSize = 0;  //Auslastungs-Variable
if (dataSize < data.length) {
        data[dataSize] = price; //price vom Typ double
        dataSize++;
}
else {
double[] newData = new double [2* data.length];
System.arraycopy(data, 0, newData, 0, data.length);
}
```