Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Institut für Angewandte und Numerische Mathematik

Programmieren: Einstieg in die Informatik mit Java

WS 2006/2007

Dr. G. Bohlender
Dipl.–Math. techn. M. Richter 18.12.2006

Aufgabenblatt 8

Bearbeitungszeitraum: 20.12.2006 – 17.01.2007

Aufgabe 20 (Pflichtaufgabe): *Matrizen*

Unter einer *reellen* $(m \times n)$*-Matrix* versteht man die tabellarische Anordnung von mn reellen Zahlen in m *Zeilen* und n *Spalten*. Eine solche Matrix A kann allgemein wie folgt dargestellt werden:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \dots & a_{m,n} \end{pmatrix}$$

Die Zahlen $a_{1,1}, a_{1,2}, \dots$ werden die *Komponenten* von A genannt. Dabei bezeichnet man mit $a_{i,j}$ die Komponente in der i-ten Zeile und der j-ten Spalte von A. Die Zahl m wird die *Zeilendimension* von A genannt. Die Zahl n heißt die *Spaltendimension* von A. Eine $(m \times 1)$-Matrix heißt m*-dimensionaler Spaltenvektor*, und eine $(1 \times n)$-Matrix heißt n*-dimensionaler Zeilenvektor*.

Für jede $(m_A \times n_A)$-Matrix A mit den Komponenten $a_{1,1}, a_{1,2}, \dots$ und jede $(m_B \times n_B)$-Matrix B mit den Komponenten $b_{1,1}, b_{1,2}, \dots$ ist das Produkt AB definiert, sofern n_A mit m_B übereinstimmt. Das Ergebnis dieser sogenannten *Matrixmultiplikation* AB ist eine $(m_A \times n_B)$-Matrix C mit den Komponenten $c_{1,1}, c_{1,2}, \dots$. Dabei gilt:

$$c_{i,j} = \sum_{k=1}^{n_A} a_{i,k} b_{k,j} \qquad i = 1, \dots, m_A, \quad j = 1, \dots, n_B \tag{1}$$

Schreiben Sie ein Java-Programm mit dem Namen `Matrizen`, welches zwei Matrizen miteinander multipliziert. Gehen Sie dabei wie folgt vor:

(a) Schreiben Sie eine statische Methode mit dem Namen `matrixEinlesen` und einer leeren Argumentenliste. Lesen Sie innerhalb der Methode zunächst zwei natürlichen Zahlen m und n von der Konsole ein. Erstellen Sie danach ein zweidimensionales `double`-Feld `A` mit den Dimensionen m und n. Dieses `double`-Feld soll eine $(m \times n)$-Matrix A mit den Komponenten $a_{1,1}, a_{1,2}, \dots$ repräsentieren. Lesen Sie die einzelnen Komponenten des Feldes von der Konsole ein. Benutzen Sie dazu zwei geschachtelte `for`-Schleifen. Beachten Sie, dass die erste Komponente eines Feldes mit Null indiziert ist! Das bedeutet, dass Sie die Matrixkomponente $a_{i,j}$ in der Feldkomponente `A[i-1][j-1]` speichern müssen. Die Methode soll das eingelesene Feld `A` zurückgeben.

(b) Schreiben Sie eine statische `void`-Methode mit dem Namen `matrixAusgeben`, welche ein übergebenes, zweidimensionales `double`-Feld `A` auf der Konsole ausgibt. Stellen Sie in der Methode zunächst die Dimensionen von `A` fest. Verwenden Sie dazu `A.length` sowie `A[0].length`. Geben Sie das Feld `A` anschließend in tabellarischer Form aus, wobei die Feldkomponente `A[i-1][j-1]` in der i-ten Zeile und der j-ten Spalte der Ausgabe erscheinen sollte.

(c) Schreiben Sie eine statische Methode mit dem Namen `matrixMultiplikation`. Definieren Sie zwei zweidimensionale `double`-Felder `A` und `B` als formale Argumente der Methode. Stellen Sie die Dimensionen m_A, n_A, m_B und n_B der übergebenen Matrizen fest. Überprüfen Sie, ob die Spaltendimension n_A von `A` mit der Zeilendimension m_B von `B` übereinstimmt. Ist dies der Fall, so erzeugen Sie ein zweidimensionales `double`-Feld mit den Dimensionen m_A und n_B. Setzen Sie die Komponenten von `C` gemäß Formel (1). Achten Sie auf korrekte Indizierung! Die Methode soll das Feld `C` anschließend zurückgeben.

Geben Sie, falls n_A und m_B nicht übereinstimmen, die Meldung `Fehler: Innere Matrixdimensionen müssen übereinstimmen` auf dem Bildschirm aus. Die Methode soll in diesem Fall den Wert `null` zurückgeben.

(d) Lesen Sie in der `main`-Methode zwei Matrizen ein, multiplizieren Sie diese und geben Sie das Ergebnis der Multiplikation am Bildschirm aus. Sie können Ihr Programm mit folgenden, bekannten Ergebnissen testen:

$$\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 2 \\ -2 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} -2 \\ 7 \\ 0 \end{pmatrix}, \qquad \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 4 & 1 \end{pmatrix}$$

Aufgabe 21: *Weihnachtsbaum*

Etwas weihnachtliches ...

(a) Kochen Sie sich einen Topf voll Glühwein oder eine Kanne voll Tee, und setzen Sie sich damit an Ihren Rechner. Öffnen Sie in weihnachtlicher Stimmung ihren Editor und schreiben Sie darin ein Java-Programm mit dem Namen `Weihnachtsbaum`.

(b) Definieren Sie eine konstante `int`-Klassenvariable `N=3` und eine konstante `double`-Klassenvariable `KUGELDICHTE=0.1`. Schreiben Sie eine statische `void`-Methode `luft` mit einem `int`-Argument `n`. Diese Methode soll `n` Leerzeichen ohne Zeilenumbruch auf der Konsole ausgeben.

(c) Schreiben Sie eine statische `void`-Methode `zweig`. Definieren Sie für diese Methode ein formales `int`-Argument `n` sowie ein formales `char`-Argument `spitze`. Die Methode soll mittels `for`-Schleife insgesamt `n` Zeichen auf der Konsole ohne Zeilenumbruch ausgeben. Das erste und das letzte Zeichen soll dabei das in `spitze` übergebene Zeichen sein. Mit dem zweiten, dem dritten, ..., `(n-1)`-ten Zeichen soll wie folgt Verfahren werden: Erzeugen Sie jedesmal mit der Methode `Math.random()` eine Zufallszahl. Ist diese Zufallszahl kleiner als die Konstante `KUGELDICHTE`, so geben Sie ein `'o'` aus. Andernfalls soll ein `'*'` ausgegeben werden.

(d) Schreiben Sie eine statische `void`-Methode mit dem Namen `stamm` ohne formale Argumente. Rufen Sie innerhalb dieser Methode die Methode `luft()` mit dem Argument `2*N-1` auf und geben Sie anschließend drei Rauten `###` mittels `System.out.println()` aus. Führen Sie beide Befehle ein zweites Mal aus.

(e) Übernehmen Sie die folgende Methode in Ihr Programm:

```
static void ast(int k) {
  int l = 2*(N - k + 1) + 1;
  int n = 4*(k-1) - 1;
  if ( k > 1 ) {
    luft(l);
    zweig(n,'*');
    System.out.println();
  }
  luft(l-1);
  zweig(n+2,'*');
  System.out.println();
  luft(l-3);
  zweig(n+6,'|');
  System.out.println();
  luft(l-3);
  zweig(n+6,'*');
  System.out.println();
}
```

(f) Rufen Sie in der `main`-Methode die Methode `ast()` mit den Argumenten `k = 1,2 ... N` nacheinander auf (`for`-Schleife). Rufen Sie abschließend die Methode `stamm()` auf. Jetzt ist Ihr Tee oder Glühwein bestimmt kalt geworden. Aber dafür sollten Sie auf Ihrem Bildschirm eine schöne Ausgabe sehen.

Das Java-Team wünscht Ihnen frohe Weihnachten
und einen guten Rutsch ins neue Jahr!