2 Arbeiten mit Arrays

2.1 Vektor-Matrix-Multiplikation

Diese Aufgabe korrespondiert mit den Slides zur Fallstudie "Zufalls-Surfer", lässt sich aber völlig unabhängig davon lösen. Ziel ist es zu zeigen, wie ein Vektor zu einem Fixwert konvergiert, wenn er wiederholt mit der gleichen Wahrscheinlichkeits-Matrix multipliziert wird. Wobei die Matrix gewisse Bedingungen erfüllen muss.

Schreiben Sie eine statische Methode, die das Produkt von einem beliebigen Vektor der Länge n mit einer $n \times n$-Matrix berechnet. Sie können die Berechnung z. B. nach dem üblichen Falk-Schema durchführen.

```
1  public static double[] multiply(double[] v, double[][] m) {
2      assert (v.length == m.length); // vector length & matrix height must be equal
3      double[] res = new double[m.length]; // result vector
4      // TODO: Add code here
5      return res;
6  }
7
8  ...
9  public static void main(String[] args) {
10     double[] v = {1,0,0,0,0}; // starting state
11     double[][] m = { // transition probabilities
12             {0.1,0.3,0.2,0.1,0.3},
13             {0.1,0.2,0.1,0.3,0.3},
14             {0.3,0.1,0.3,0.2,0.1},
15             {0.5,0.5,0.0,0.0,0.0},
16             {0.9,0.0,0.0,0.1,0.0}};
17     show(calculate(v,m,0)); // the input data
18 ...
```

Aufgaben:

a) Erstellen Sie ein neues Java-Projekt (z.B. "MatrixVectorMultiplication") und fügen Sie in das Projekt eine (z. B. gleichnamige) Klasse ein.

b) Die Java-Datei mit sämtlichem vorgegeben Code ist auf der Website verfügbar. Nebenbei: Sie können den Code mittels Copy & Paste in Ihre eigene Klasse übernehmen. Beachten Sie, dass die öffentliche Klasse immer den gleichen Namen wie die Datei tragen muss.

c) Implementieren und testen Sie die Methode "`multiply`".

d) Folgender Testcode sollte den Output "`7.0 10.0`" produzieren.

```
1 double testV[] = {1,2};
2 double testM[][] = {{1,2},{3,4}};
3 show(multiply(testV,testM));
```

e) Zu welchen Werten (auf je drei signifikante Ziffern) konvergiert der Vektor bei den gegebenen Werten im Java-File?

f) Was passiert, wenn Sie in der Matrix Werte grösser als 1 einfügen?

g) Was passiert, wenn Sie eine Zeile der Matrix mit Nullen füllen?

2.2 Highscore

In dieser Aufgabe soll eine Highscore-Tabelle, wie sie in unzähligen Spielen vorkommt, implementiert werden. Der simple Teil des Codes ist auf der Website vorgeben. Ihre Hauptaufgabe ist es die Methode zum Einfügen eines neuen Resultats (`score`) in die Tabelle zu implementieren.

Aufgaben:

a) Übernehmen Sie das Codegerüst.

b) Implementieren und testen Sie die Methode `showHighscore`. Das sollte an sich kein grosses Problem darstellen.

c) Implementieren und testen Sie die Methode `insertScore`. Diese Aufgabe hingegen ist schwieriger, als sie auf den ersten Blick aussieht. Das heisst, die Lösung erfordert etwas Denksport. Beachten Sie auf jeden Fall alle drei Fälle:

- Der gegebene Score ist **grösser als alle** Scores im Array.
- Der gegebene Score ist **kleiner als alle** Scores im Array.
- Der gegebene Score liegt **zwischen** den Scores im Array.

Zum Test können Sie die Methode `TestScore` in Ihrer `main`-Methode verwenden. Das Programm muss so ohne Absturz (`Exception`) durchlaufen und die Highscores müssen immer strikte absteigend bleiben.

d) **Freiwillig:** Verwenden Sie den Highscore-Array für ein einfaches Spiel z. B. Zahlenraten. Fühlen Sie sich frei etwas Eigenes zu Implementieren. Falls Sie irgendwo nicht weiterkommen, hilft Ihnen vielleicht das Beispielprogramm im Template auf der Website.

```
1  public class HiLoGame {
2
3      private static int[] highscore = new int[10];
4
5      /**
6       * Runs through the array in a for loop and prints the scores.
7       * The top score should be printed first.
8       */
9      public static void showHighscore() {
10         System.out.println("*** HIGHSCORE ***");
11         // TODO: Add code here
12     }
13
14     /**
15      * Enters a new score into the highscore array. The order must be
16      * maintained and no lower score should ever replace a higher one.
17      */
18     public static void insertScore(int score) {
19         // TODO: Add code here
20         // Some ideas but you may achieve the goal completely different:
21         // At first check if the score is good enough for entering the
22         // highscore list. Second: use a while loop or something similar
23         // to determine where the new score needs to be put. Third: use a for-
24         // loop to shift every score lower than the new score in the array.
25     }
26
27     ...
28 }
```