Lehrstuhl Stuckenschmidt
Javakurs FSS 2012
Tag 2 - Schleifen und Arrays

Tim Endlich, Thorsten Knöller, Niklas Dürr, Christian Meilicke

23. Juli 2012

Zum Inhalt Mit den Aufgaben dieses Blattest werden Arrays (Feldvariablen) und Methoden eingeführt. Mittels Methoden kann und sollte der Code besser strukturiert werden. Zudem sind nun viele Aufgaben mit Testmethoden ausgestattet.

Aufgabe 1: Geometrische Berechnungen 2

`Geo2`

In der Klasse `Geo2` finden Sie drei Methoden, die Umfang, Fläche und Volumen eines Kreises, beziehungsweise einer Kugel berechnen. Zurückgegeben wird jeweils ein `double` Wert. Schreiben Sie die entsprechenden Formeln in die dazugehörigen Methoden und testen Sie ihre Ergebnisse mit der Klasse `Geo2Test`!

Aufgabe 2: Alphabet

`Alphabet`

Schreiben Sie eine Methode, die jeden zweiten Buchstaben der Eingabe als Großbuchstaben ausgibt. Jeder Buchstabe soll durch eine Leerstelle getrennt sein. Testen Sie die Methode mit dem gegebenen String und auch mit einer anderen Eingabe. Die Ausgabe für den `String alphabet` muss so aussehen:

```
 B D F H J L N P R T V X Z
```

Aufgabe 3: Wetterstation

`Wetter`

Eine Wetterstation hat für 14 Tage beispielsweise folgende Temperaturwerte aufgenommen:

```
Tag: 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Temperatur: 12 14 9 12 15 16 15 15 11 8 13 13 15 12
```

Schreiben Sie eine Methode, die für die gegebene Zeitspanne die Durchschnittstemperatur berechnet!
Schreiben Sie eine Methode, welche die beiden aufeinanderfolgenden Tage angeben kann, an denen es den größten Temperaturumschwung gab!

Aufgabe 4: Mona-Lisa

`Mona-Lisa`

Gegeben ist ein Programm, dass ein Bild in ein eine zweidimensionales Array speichert. Das Array `pixel[x][y]` speichert die Grausutfe des Pixels an der Stelle (x,y). Schreiben Sie die Bildbearbeitungsmethoden `invertieren()`, `aufhellen()` und `schwarzweiss()`. Überprüfen Sie anhand Ihres Ergebnisses selbständig ob die Funktionen wie erwartet funktionieren.

Aufgabe 5: Bubblesort

`Bubblesort`

Schreiben Sie eine Methode `bubblesort`, die ein gegebenes `int` Array mit dem Bubblesort Algorithmus sortiert.

Aufgabe 6: Primzahlen

`Primzahlen`

Schreiben Sie eine Methode, die alle Primzahlen bis `n` berrechnet! Benutzen Sie ein `boolean` Array und speicheren Sie von `0,..,n` für jede Zahl ob sie prim (`true`) oder nicht prim (`false`) ist.

Aufgabe 7: Prüfbits *

`Pruefbits`

Schreiben Sie eine Methode, die für ein quadratisches zweidimensionales Array vom Typ `boolean`, in dem sich Daten und Checkbits befinden, entscheidet, ob es sich um einen validen Datenblock handelt, d.h. ob die Daten- und Checkbits korrekt zueiander passen. Dabei stehen die Checkbits in der Spalte ganz rechts und in der Zeile ganz unten. Die Werte der Checkbits ergeben sich jeweils durch die XOR Verknüpfung der jeweiligen Spalten bzw. Zeilen. Dem Checkbit unten rechts kommt eine besondere Rolle zu. Dieses Checkbit ergibt sich aus der XOR Verknüpfung aller anderen Checkbits. Im folgenden ein Beispiel für einen validaen Datenblock (noch einmal ausführlicher skizziert auf der letzten angehefteten Seite)

- Schreiben Sie zunächst eine Methode, die einen Datenblock wie unten ausgibt.
- Schreiben Sie dann die Methode, die bestimmt, ob es sich um einen validen Datenblock handelt und testen Sie diese mit den vorhanden Testfällen!

Beispiel eines validen Datenblocks:

```
1 0 1 0 1 1
0 0 0 0 0 0
1 1 1 1 0 0
1 1 1 1 1 1
0 0 1 1 0 0
1 0 0 1 0 0
```

Aufgabe 8: Palindrom Rekursiv

`Palindrom`

Schreiben Sie eine Methode `isPalindrom`, die für ein char-Array rekursiv erkennt, ob es sich dabei um ein Palindrom handelt. Ein Palindrom ist ein Wort, welches vorwärts und rückwärts gelesen identisch ist. Beispiele sind: 'ABBA', oder 'lagerregal'. Lagerregal' muss nicht als Palindrom erkannt werden, d.h. Groß und und kleingeschriebene Buchstaben gelten als verschieden.

Aufgabe 9: Permutationsvektoren *

`Permutation`

Ein Array `int[n]` enthält eine Permutation, wenn alle Zahlen `1,...,n` genau einmal als Element vorkommen. Schreiben Sie eine boolsche Methode `isPermutation`, die prüft, ob ein übergebenes Array zu einer Permutation ergänzt werden kann. Eine `0` spielt dabei die Rolle eines Jokers, der beliebige Werte `1,...,n` annehmen kann.[1]
Beispiele:

- `(2,1,4,3)` ist eine Permutation der Zahlen `1,2,3,4`.
- `(1,2,2,3)` ist keine Permutation.
- `(2,0,0,3)` kann zu Permutation `(2,1,4,3)` ergänzt werden.
- `(2,0,0,2)` kann nicht zu Permutation ergänzt werden.

Aufgabe 10: Galgenmännchen *

`Hangman`

Schreiben Sie ein Programm, mit dem man Galgenmännchen spielen kann. Ein- und Ausgabe erfolgt über die Konsole. Für die Ausgabe wurde die Methode `fehlerAusgabe(int fehler)` bereits hinzugefügt. Diese gibt je nach Anzahl der Fehler das Galgenmännchen auf der Konsole aus. Außerdem wurde die Klasse `Scanner` bereitgestellt um Eingaben auf der Konsole zu verarbeiten. Mit dem Befehl `.nextLine()` wird die Eingabe auf der Konsole bis zu einem Zeilenumsprung ausgelesen und in einen `String` umgewandelt.[2]

Aufgabe 11: Damenproblem **

`Dame`

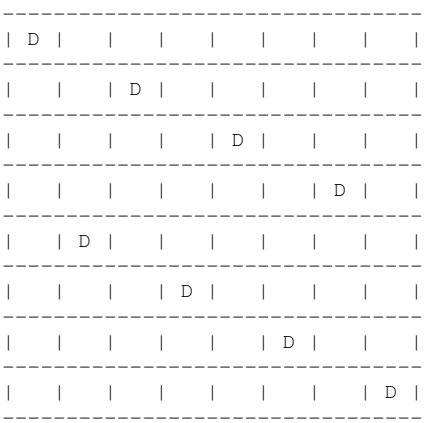
Das Damenproblem besteht darin, acht Damen auf einem Schachbrett so zu positionieren, dass diese sich nicht gegenseitig schlagen können (d.h. keine zwei Damen dürfen auf derselben Diagonale, Reihe oder Spalte stehen). Wir suchen eine mögliche Lösung des Problems. Hierzu verwenden wir eine lokale Suche, genaugenommen eine Variante des Hillclimbing.

[1] Entnommen aus: 'Das Java-Praktikum', von Reinhard Schiedermeier und Klaus Köhler
[2] Entnommen aus: http://wiki.freitagsrunde.org/Javakurs/Übungsaufgaben

Entwicklen Sie schrittweise ein Programm, das eine mögliche Lösung konstruiert! Für die folgenden Teilaufgaben soll das Schachbrett mit den Damen mittels einer 8-elementigen `int`-Feldvariable repräsentiert werden. Dabei wird augenutzt, dass in einer Lösung keine zwei Damen in derselben Reihe stehen können, d.h. man weiss bereits, dass in jeder Reihe genau eine Dame stehen muss und betrachtet daher nur Zustände, in denen dies der Fall ist. Wir betrachten folgendes Beipiel:

```
int[] field = new int[]{0,2,4,6,1,3,5,7};
```

entspricht dem Schachbrett



bei dem sich genau ein Damenpaar bedroht (die Dame ganz links oben und die Dame ganz rechts unten).

a) Vervollständigen Sie die Methode `printField`, mit deren Hilfe Sie eine Visualisierung wie in obigem Beispiel ausgeben können!

b) Vervollständigen Sie die Methode `createRandomField`, mit deren Hilfe sich zufällig generierte Felder erezugen lassen! Überprüfen Sie die Methode, in dem Sie einige der erzeugten Felder anschauen!

c) Vervollständigen Sie die Methode `getNumOfBadPairs`! Diese Methode berechnet für ein gegebenes Feld, Paare von Damen sich gegenseitig bedrohen. Würde man diese Methode auf eine Lösung anwenden, so würde der Rückgabewert 0 sein.

d) Vervollständigen Sie die Methode `transformToBetterNeighbor`! Diese Methode ist genauer in ihrem Kommentar erklärt.

e) Vervollständigen Sie die Methode `searchSolution`! Diese Methode erhält als Eingabe ein beliebiges Feld und wendet solange die Methode `transformToBetterNeigbor` an, bis das Feld entweder kein Paar von Damen mehr enthält, die sich gegenseitig schlagen, oder bis sich das Feld nicht mehr verbessern läßt.

f) Verwenden Sie alle gegebenen Methoden, um so eine Lösung des Problems zu finden und anzuzeigen. Dies ist sehr einfach möglich. Rufen Sie einfach die Methode `searchSolution` so lange für ein zufällig generiertes Feld auf, bis Sie eine Lösung gefunden haben