2. Übung zur Veranstaltung *Proinformatik: Objektorientierte Programmierung*

Freie Universität Berlin
Fachbereich Mathematik und Informatik
Institut für Informatik, SoSe 2012
Dr. Marco Block-Berlitz

1. Wir haben in der Vorlesung gesehen was passiert, wenn weniger Parameter übergeben werden als das Programm sie verlangt oder erwartet. Testen Sie, wie das Programm reagiert, wenn Sie mehr als die erwartete Anzahl von Parametern übergeben.

2. Erzeugen Sie mit der Klasse `ErzeugeVieleWerte` eine Datei, die 100 Zeilen à 10 Spalten besitzt und in jedem Eintrag folgenden Wert enthält: Eintrag(Zeile i, Spalte j) $= i^2 + j$.

3. Schreiben Sie die Funktion `istPrimzahl`, die als Eingabe einen Integerwert n erhält und mit einem `boolean` zurückgibt, ob n eine Primzahl ist oder nicht. Eine Primzahl n ist eine Zahl, die nur durch 1 und sich selbst ganzzahlig ohne Rest teilbar ist. Die ersten Primzahlen sind: $2, 3, 5, 7, 11, 13, ...$

4. Verändern Sie die Funktion aus Aufgabe 3 so, dass alle Primzahlen bis n auf dem Bildschirm ausgegeben werden.

5. Implementieren Sie das Verhalten der beiden in der Vorlesung verwendeten Funktionen:

   (a) `boolean Character.isUpperCase(char)` und

   (b) `char Character.toLowerCase(char)`

   Ersetzen Sie die Passagen in den Beispielen und überprüfen Sie damit die Korrektheit Ihrer Funktionen.

6. Schreiben Sie eine Methode `addVektoren`, die als Eingaben zwei `int`-Arrays `a` und `b` erhält. Die Funktion liefert das Ergebnis der elementeweisen Addition beider Vektoren, falls Sie die gleiche Dimension besitzen und ansonsten `null` (damit repräsentieren wir ein leeres Objekt). Geben Sie weiterhin eine Funktion zeigeVektor an, die den Inhalt eines Vektors ausgibt. Anschließend sollten beide Funktionen wie folgt ohne Fehlermeldungen getestet werden können:

```
int[] a = {1,2,3};            // funktionierendes Beipiel
int[] b = {4,5,6};
zeigeVektor(a);
zeigeVektor(b);
int[] c = addVektoren(a, b);
zeigeVektor(c);

int[] d = {1,2};              // fehlerhaftes Beispiel
int[] e = {4,5,6};
zeigeVektor(d);
zeigeVektor(e);
int[] f = addVektoren(d, e);
zeigeVektor(f);
```

7. Analog zu Aufgabe 6 sollen Funktionen für Vektorsubtraktion und das Produkt aus Vektor und Skalar (bzw. Skalar und Vektor) implementiert werden. Alle Parameter sind wieder ganzzahlig.

8. In dem Projekt Conway's Game of Life haben wir die Randproblematik dadurch gelöst, dass wir einen zusätzlichen Rand bereitgestellt haben. Nehmen Sie den Rand wieder heraus und ändern Sie die Version derart, dass die Überprüfung der Nachbarschaft intelligent und fehlerfrei erfolgt.