252-0842-00L Programmieren und Problemlösen **Aufgabenblatt 2**
FS 2016 Herausgabe: 02.03.2016
Dr. A. L. Schüpbach, G. Zellweger Besprechung: 09.03.2016
Departement Informatik, ETH Zurich

1 Methoden und Rekursion

1.1 Fibonacci-Folge

Die Fibonacci-Folge ist eine unendliche Folge von Zahlen (1, 1, 2, 3, 5, 8, 13, ...), die durch folgende *Rekursion* gegeben ist:

$$f_n = f_{n-1} + f_{n-2}, \quad (\text{für } n > 2)$$

mit Anfangswerten $f_0 = 0$ und $f_1 = 1$. Programmieren Sie eine statische Methode, welche die nte Fibonacci-Zahl mittels Rekursion berechnet (z.B. `fibonacci(13) = 7`). Benutzen Sie dann diese Methode, um die ersten 50 Zahlen der Fibonacci-Folge auszugeben.

1.2 Rekursiver ggT

In der letzten Übung wurden Sie gebeten, den grössten gemeinsamen Teiler (ggT) von zwei ganzen Zahlen mit einem iterativen Algorithmus zu berechnen. Bitte berechnen Sie den ggT in dieser Aufgabe mittels einer rekursiven, statischen Methode.

2 Komplexe Zahlen

Schreiben Sie eine Java Klasse, welche das Rechnen mit komplexen Zahlen unterstützt und implementieren Sie die arithmetischen Operationen: Addition, Subtraktion, Multiplikation und Division.

Schreiben Sie Ihre Klasse so, dass man nicht nur mit komplexen Zahlen rechnen kann, sondern zum Beispiel auch eine reelle Zahl zu einer komplexen hinzufügen kann (Tipp: Überladen von Methoden).

Sie können die neue Klasse für komplexe Zahlen als innere (statische) Klasse einfach zu ihrer Main-Klasse hinzufügen.

3 Matrix Multiplikation

Für diese Aufgabe steht dir eine Template-Datei auf der Webseite zur Verfügung.

3.1 Multiplizieren via Methode

Programmieren Sie eine Methode `transpose_matrix(...)`, die eine Matrix A als Argument nimmt und die *transponierte* Matrix A^T als Resultat zurückgibt.

Programmieren Sie nun eine zweite Methode `multiply_matrices(...)`, die zwei Matrizen multipliziert, welche als Argumente übergeben werden. Sie können in dieser Übung davon ausgehen, dass die zwei Matrizen von Klienten der Methode richtig dimensioniert werden (Sie müssen keine ArrayIndexOutOfBoundsExceptions abfangen).

Berechnen Sie nun $A \cdot A^T$ für die Matrix A aus Übung 1 nochmals, aber verwenden Sie dieses Mal die neu erstellten Methoden.

3.2 Komplexe Matrizen Multiplizieren

Fügen Sie nun die in Aufgabe 2 erstellte Klasse für komplexe Zahlen dem Programm aus 3.1 hinzu und überladen Sie die Methoden `transpose_matrix(...)` und `multiply_matrices(...)`, so dass diese Methoden auch für komplexe Matrizen aufgerufen werden können.

3.3 Fourier-Transformation

Berechnen Sie nun die Fourier-Transformation der Vektoren $x_1 = [0, 1, 2, 3]$ und $x_2 = [0, 1, 2, 3, 4, 5, 6, 7]$. Die Einträge der $n \times n$ Transformationsmatrix F_n sind wie folgt definiert:

$$F_{jk} = e^{2\pi ijk/n} = \omega^{jk}$$

Somit ergibt sich die 4-dimensionsale Transformationsmatrix:

$$F_4 = \frac{1}{\sqrt{(4)}} \times \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{pmatrix}$$

und die 8-dimensionsale Transformationsmatrix:

$$F_8 = \frac{1}{\sqrt{(8)}} \times \begin{pmatrix} \omega^0 & \omega^0 & \omega^0 & \dots & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \dots & \omega^7 \\ \omega^0 & \omega^2 & \omega^4 & \dots & \omega^{14} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^7 & \omega^{14} & \dots & \omega^{49} \end{pmatrix} \quad , \quad \omega = \frac{1}{\sqrt{(2)}} + \frac{i}{\sqrt{(2)}}$$

Tipps: Es könnte sich lohnen, eine `power(...)`-Methode zu ihrer Klasse für komplexe Zahlen hinzuzufügen. Die Wurzel einer reellen Zahl kann mit `Math.sqrt(...)` berechnet werden. Um Ihre Lösung zu überprüfen, geben Sie in www.wolframalpha.com "fft[0,1,2,3]", bzw. "fft[0,1,2,3,4,5,6,7]" ein.