Allgemeines Programmierpraktikum **SS 2008**
Dr. Henrik Brosenne Georg-August Universität Göttingen
Institut für Informatik

Übung 4

Testat nach Mittwoch, 21.05., 14.00 Uhr in der Gruppenübung.

Vorbereitung

Java

Informieren Sie sich in der Literatur, z.B. unter `http://www.galileocomputing.de/openbook/javainsel7/`, darüber wie Java ein- und mehrdimensionale Felder verwaltet.

Aufgabe 1 – 100 Punkte

Algebra–Bibliothek

Als Beispiel einer Algebra-Bibliothek sollen das Gauß–Jordan–Verfahren zur Lösung linearer Gleichungssysteme über den rationalen Zahlen implementiert werden.

Anforderungen

Kommentieren Sie Ihren Programmtext **ausführlich**. Spärliche und/oder schlechte Kommentierung führt zu Punktabzug.

Klasse `Rational` für die Verarbeitung rationaler Zahlen.

- Zähler und Nenner werden als `long` dargestellt.
- Überschreiben Sie die von der Superklasse `java.lang.Object` geerbten Methoden `equals` und `toString` sinnvoll.

  Hinweis. Der Test, ob Zähler und Nenner paarweise gleich sind, reicht für den Test auf Gleichheit zwei rationaler Zahlen nicht aus, Beispiel $\frac{1}{2} = \frac{2}{4}$.
- Implementieren Sie alle weiteren Attribute, die Sie bei der Umsetzung des Gauß–Jordan–Verfahrens benötigen.

Klasse `Matrix` zum Rechnen mit Matrizen über den rationalen Zahlen.

- Die in der Matrix gespeicherten Werte sind Objekte der Klasse `Rational`.
- Eine leere Matrix beliebiger Dimension kann erzeugt werden.

- Für die Dimension der Matrix ist ein Datenfeld vorgesehen, das über eine entsprechende Methode gelesen werden kann.
- Jeder einzelne Matrixeintrag kann, über entsprechende Methoden, gesetzt und gelesen werden.
- Überschreiben Sie die von der Superklasse `java.lang.Object` geerbten Methoden `equals`, `clone` und `toString` sinnvoll.
- Schnelles Vertauschen (ohne Umspeichern der Einträge) zweier Zeilen ist möglich.

  Hinweise Die Matrix wird intern durch einem Feld von Referenzen auf die Zeilen der Matrix realisiert. Beim Vertauschen von Zeilen müssen so nur zwei Referenzen vertauscht werden. Trotzdem kann direkt auf jedes einzelne Element zugegriffen werden.
- Objekt- und Klassenmethoden für die Muliplikation zweier Matrizen sind implementiert. Sind die Dimensionen der Matrizen nicht passend, wird das mit einer `Exeception` gemeldet.

Klasse `Algebra` beinhaltet die Methoden der Algebra-Bibliothek.

- Matrizen und Vektoren sind Objekte der Klasse `Matrix`.
- Es wird eine Klassenmethode für das Gauß–Jordan–Verfahren exportiert. Diese Methode erwartet als Parameter eine Matrix und einen Vektor. Das Ergebnis wird in einem Vektor zurückgeliefert. Erfolg oder auftrettende Fehler werden über `Execeptions` gemeldet. Die Methode verändert die Parameter Matrix und Vektor nicht, sie sind nach Ablauf der Methode unverändert.

Testklasse.

- Der Dateiname der Datendatei wird in der Kommandozeile übergeben.
- Die in der Datei enthaltenen Zahlen sind ausschließlich ganze Zahlen.
- Die Datendatei enthält zunächst die rechte Seite des Gleichungssystems, dann ein Dollarzeichen, dann zeilenweise die Koeffizienten der Matrix. Die Datei kann beliebig viele *whitespaces* enthalten.
- Das Dateiformat ermöglicht es zuerst die rechte Seite einmal zu lesen, um die Anzahl der Variablen zu bestimmen und entsprechend grosse Matrizen zu erzeugen. Anschliessend wird die Datei zurückgesetzt, und erst beim erneuten Lesen werden die Daten gespeichert.
- Falls beim Lesen der Datei ein Fehler auftritt, wird das Programm mit einer entsprechenden Meldung beendet.

- Die Lösung wird mit Methoden der Algebra-Bibliothek berechnet. Im Falle einer singulären Koeffizientenmatrix wird der Tatbestand gemeldet und das Programm beendet. Sonst wird die Lösung und zur Kontrolle auch das Produkt aus Koeffizientenmatrix und Lösung ausgegeben.

Gauß–Jordan–Verfahren

Gegeben sei ein lineares Gleichungssystem $Ax = b$. Mit quadratischer Koeffizientenmatrix $A = (a_{ij})$, Variablenvektor $x = (x_j)$ und Ergebnisvektor $b = (b_i)$ für $i, j = 1, \ldots, n$ mit $n > 0$.

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Gesucht ist eine Belegung des Vektors x, die die Gleichung erfüllt.

Gauß–Jordan–Verfahren mit Spaltenpivotsuche

1. Setze $A^{(0)} := A$, $b^{(0)} := b$.
2. Wiederhole für $k = 1, \ldots, n$:

   (2a) Bestimme ℓ mit $k \leq \ell \leq n$, so daß gilt

   $$|a_{\ell k}^{(k-1)}| = \max_{k \leq j \leq n} |a_{jk}^{(k-1)}|$$

   (2b) Falls $a_{\ell k}^{(k-1)} = 0$ gilt, ist A singulär (nicht invertierbar) und das System nicht lösbar.

   (2c) Bilde aus $A^{(k-1)}$ und $b^{(k-1)}$ jeweils durch Vertauschung der k–ten und ℓ–ten Zeile $\hat{A}^{(k)}$ und $\hat{b}^{(k)}$.

   (2d) Berechne die Matrix $A^{(k)}$

   $$a_{ij}^{(k)} := \begin{cases} \hat{a}_{ij}^{(k)} & i = 1, \ldots, n; j = 1, \ldots, k-1 \\ \hat{a}_{kj}^{(k)} & i = k; j = k, \ldots, n \\ \hat{a}_{ij}^{(k)} - \hat{a}_{kj}^{(k)} \left(\hat{a}_{ik}^{(k)} / \hat{a}_{kk}^{(k)} \right) & \text{sonst} \end{cases}$$

   (2e) Berechne den Vektor $b^{(k)}$

   $$b_i^{(k)} := \begin{cases} \hat{b}_k^{(k)} & i = k \\ \hat{b}_i^{(k)} - \hat{b}_k^{(k)} \left(\hat{a}_{ik}^{(k)} / \hat{a}_{kk}^{(k)} \right) & \text{sonst} \end{cases}$$

3. Setze $x_k := b_k^{(n)} / a_{kk}^{(n)}$ für $k = 1, \ldots, n$.

Implementation

1. Es wird **nicht** in jedem Schritt eine Kopie der Matrizen und des Vektors erstellt. Die Berechnungen werden so durchgeführt, dass immer dieselben Matrizen und derselbe Vektor benutzt werden können.

2. Der Quotient $\left(\hat{a}_{ik}^{(k)} / \hat{a}_{kk}^{(k)}\right)$ muss in jedem Schritt nur $(n-1)$–mal berechnet werden. Deshalb wird ein Vektor erzeugt, der die Quotienten aufnimmt.

3. Aufgrund der Probleme mit der Gleitkomma-Arithmetik wird die Berechnung von $A^{(k)}$ noch stärker zerlegt.

$$a_{ij}^{(k)} := \begin{cases} \hat{a}_{ij}^{(k)} & i = 1, \ldots, n; j = 1, \ldots, k-1 \\ 0 & i = 1, \ldots, k-1, k+1, \ldots, n; j = k \\ \hat{a}_{kk}^{(k)} & i = k; j = k \\ \hat{a}_{kj}^{(k)} & i = k; j = k+1, \ldots, n \\ \hat{a}_{ij}^{(k)} - \hat{a}_{kj}^{(k)} \left(\hat{a}_{ik}^{(k)} / \hat{a}_{kk}^{(k)}\right) & \text{sonst} \end{cases}$$

Denn für die k–te Spalte $(j = k)$ ohne den k–ten Eintrag $(i \neq k)$ gilt:

$$a_{ij}^{(k)} = \hat{a}_{ik}^{(k)} - \hat{a}_{kk}^{(k)} \left(\hat{a}_{ik}^{(k)} / \hat{a}_{kk}^{(k)}\right) = 0 \text{ für } i = 1, \ldots, k-1, k+1, \ldots, n; j = k$$