Westfälische Wilhelms-Universität Münster

Übungen zur Vorlesung „Informatik I: Grundlagen der Programmierung“ im WS 2012/13

Prof. Dr. Jan Vahrenhold
Dipl.-Inf. Andreas Thom

Blatt 11

Abgabe via LearnWeb bis zum 14.01.2013, 14:00 Uhr

Alle von Ihren programmierten Methoden sind zu kommentieren!

Aufgabe 41: (BA, 5 Punkte) Erstellen Sie ein Java-Programm, mit dessen Hilfe Sie nachrechen können, ob der 13. eines Monats im Gregorianischen Kalender (also in unserem Kalender) häufiger auf einen Freitag als auf jeden anderen Wochentag fällt.

Zeigen Sie zunächst, dass die Anzahl Tage innerhalb von 400 Jahren ohne Rest durch 7 teilbar ist. Somit reicht es, die Behauptung für 400 Jahre zu verifizieren.

Erstellen Sie dann ein Feld mit sieben Einträgen (Montag bis Sonntag), wobei für jeden Tag initial die Null gespeichert wird. Dieses Feld soll zeigen, wie häufig ein Wochentag am 13. eines Monats innerhalb von 400 Jahren vorkommt.

Iterieren Sie nun vom 01.01.1600 bis zum 31.12.1999 und lösen Sie die Aufgabe.

Da es bei der Bearbeitung dieser Aufgabe um die Einübung verschiedener imperativer Konstrukte der Programmiersprache Java geht, ist die Verwendung der Gauß'schen Wochentagsformel explizit nicht erlaubt.

Aufgabe 42: (BA, 5 Punkte) Erstellen Sie ein Java-Programm, mit dessen Hilfe zu einem Geldbetrag in Euro und Cent eine Stückelung mit möglichst großen Münzen und Banknoten berechnet wird. Verwenden Sie hierbei die in Deutschland herausgegebenen Münzen und Banknoten:

- Münzen: 1 Cent, 2 Cent, 5 Cent, 10 Cent, 20 Cent, 50 Cent, 1 Euro, 2 Euro.
- Banknoten: 5 Euro, 10 Euro, 20 Euro, 50 Euro, 100 Euro, 200 Euro, 500 Euro.

Aufgabe 43: (5 Punkte) In dieser Aufgabe soll eine Klasse `Matrix` zur Repräsentation von $m \times n$-Matrizen (im mathematischen Sinn) modelliert und implementiert werden; gehen Sie davon aus, dass in jedem Eintrag der Matrix ein Wert vom Typ `long` gespeichert werden soll; m und n sollen vom Typ `int` sein.

(a) Erstellen Sie einen Konstruktor, der ein zweidimensionales Feld sowie Werte für m (Zeilen) und n (Spalten) entgegen nimmt und hieraus eine vollständige $m \times n$-Matrix erstellt, bei der die in der Eingabe ggfs. fehlenden Felder mit Nullen besetzt werden bzw. überzählige Einträge ignoriert werden.
Beispiel: Für `long[][] eingabe = { {1, 2}, {3, 4, 5}, {6} }` soll der Aufruf des Konstruktors mit $m = 5$ und $n = 4$ ein Objekt erzeugen, das die links angegebene Matrix repräsentiert; für die Parameter $m = 2$ und $n = 2$, soll der Aufruf ein Objekt erzeugen, das die rechts angegebene Matrix repräsentiert.

$$\begin{pmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 5 & 0 \\ 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

(b) Erstellen Sie einen Konstruktor, der ein zweidimensionales Feld entgegen nimmt, hieraus selbstständig die minimal notwendigen Werte für m und n berechnet und anschließend wie oben eine interne Darstellung erzeugt.

(c) Erstellen Sie Methoden `setEntry` und `getEntry` zum Lesen und Schreiben einzelner Einträge der Matrix. Überprüfen Sie, ob die übergebenen Parameter gültig sind.

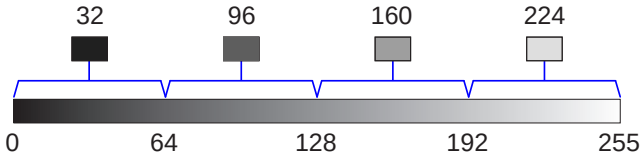
(d) Erstellen Sie eine Methode `isUpperTriangular`, die überprüft, ob es sich bei der repräsentierten Matrix um eine obere Dreiecksmatrix, d.h. eine $n \times n$-Matrix, bei der alle Einträge unterhalb der Hauptdiagonalen den Wert Null haben, handelt.

Die letzte Aufgabe auf diesem Zettel beschäftigt sich mit dem Thema „Bildkompression“. Verwenden Sie hierzu – wie bereits auf Blatt 10 – die auf der Seite `http://mediacomputation.org` bereit gestellten Klassen, die Sie in der Datei `http://coweb.cc.gatech.edu/mediaComp-plan/uploads/101/bookClasses-7-22-09.zip` finden. Die Dokumentation dieser Klassen ist in dem Unterverzeichnis `doc` des beim Entpacken dieser Datei erzeugten Verzeichnisses `bookClasses` zu finden.

Zur Vereinfachung der Berarbeitung (bzw. wenn Sie sich (noch) nicht mit dem Konzept eines *classpath* auskennen) sollten Sie die Java-Dateien Ihrer Lösung in dem Ordner `bookClasses` erzeugen.

Aufgabe 44: (5 Punkte) Ein Graustufenbild ist dadurch gekennzeichnet, dass für jedes Pixel die Farbwerte in allen Kanälen übereinstimmen. Ein Verfahren zur (verlustbehafteten) Kompression von Graustufenbildern ist die so genannte „Quantisierung“. Hierbei wird die Anzahl der pro Pixel und Farbkanal verwendeten Bits (üblicherweise und konkret im Rahmen dieser Aufgabe sind dies 8 Bits) wie folgt reduziert: Bei einer Originalanzahl von n Bits und einer Zielanzahl von ℓ Bits wird der Wertebereich pro Kanal, d.h. das Intervall $[0, 2^n - 1]$ so aufgeteilt, dass 2^ℓ Teilintervalle mit je $2^{n-\ell}$ Werten entstehen. Alle (Farb-)Werte im Inneren eines Intervalls I werden dann auf den (Farb-)Wert abgebildet, der dem Mittelpunkt des Intervalls I entspricht.

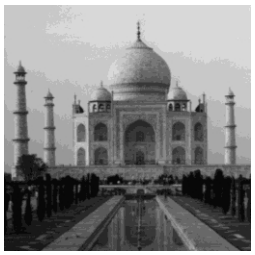
Nachfolgend ist ein Beispiel für $\ell = 2$, also für die Reduktion von $2^8 = 256$ Graustufen auf $2^\ell = 4$ Graustufen angegeben.



Erstellen Sie ein Java-Programm, das aus einem gegebenen Graustufenbild (Beispiel im LearnWeb) zu einem Parameter ℓ eine Quantisierung wie oben beschrieben durchführt.

Die nachfolgenden Bilder sind zur visuellen Kontrolle des eigenen Programms gedacht:


256 Graustufen


16 Graustufen


8 Graustufen


2 Graustufen

Verwenden Sie als Rahmen für die Bearbeitung dieser Aufgabe die im LearnWeb bereit gestellte Datei `Quantisierung.java`. Reichen Sie bei der Abgabe sowohl die ergänzte Datei `Quantisierung.java` als auch die erzeugten Bilder ein.