Algorithmen, Datenstrukturen und Programmieren II

Ferenc Domes

23. Januar 2006

Wiederholung ALGODAT I

1. (Datentypen, Fehlerbehandlung) Schreiben Sie eine Applikation, die eine beliebige Zeile vom Standard-Input einliest und entscheidet ob es ein `boolean` (`true` oder `false`), ein `int`, ein `double`, oder ein `String` ist. Benutzen Sie vorgefertigte Konvertierungsbefehle, wie `Integer.parseInt(String s)` und Fehlerbehandlung um festzustellen, ob die Konvertierung erfolgreich war oder nicht.
2. (Rekursion, Mathematik) Implementieren Sie eine einfache Rekursion, die die Fakultät einer Zahl ausrechnet, und zeichnen Sie genau auf, was bei der Berechnung von $5!$ passiert. Schreiben Sie Ihr Protokoll in Word oder scannen Sie die lesbar(!) geschriebene Lösung ein.
3. (Objekte, Mathematik) Schreiben Sie eine Klasse names `Matrix` die die wichtigsten Matrix-Operationen implementiert. Die Matrizen sollten in allgemeiner Form sein, also nicht nur quadratisch und beliebige (nicht integer) Einträge haben. Folgende Methoden sollten im Program enthalten sein:

```
//Default constructor
public Matrix()
//Constructor generating a matrix of dimension d
public Matrix(Dimension d)
//Get the dimension of the matrix
public Dimension dim()
//Get the (c,r)th element of the matrix
public double get(int c, int r)
//Set the (c,r)th element of the matrix as d
public void set(int c, int r, double d)
//Matrix addition
public Matrix add (Matrix a)
//Matrix addition (static)
public static Matrix add(Matrix a, Matrix b)
//Matrix-Scalar multiplication
public Matrix mult (double a)
//Matrix-Scalar multiplication (static)
public static Matrix mult (double a, Matrix b)
//Matrix-Matrix multiplication
public Matrix mult (Matrix a)
```

```

//Matrix multiplication (static)
public static Matrix mult (Matrix a, Matrix b)
//Sub-matrix generated by deleting the c-th column and the r-th row
public Matrix submatrix(int c, int r)
//Recursion for finding the determinant
(*)private static det(Matrix d)
//Determinant of a quadratic Matrix
(*)public double det()

```

(Für die Berechnung der Determinante benutzen Sie den Laplaceschen Entwicklungssatz). Wenn Sie genug Lust und Zeit aufbringen können, schreiben Sie die `Matrix`-Klasse im sparse-Format, das heißt jedes von 0 verschiedene Element wird als ein Tripel (i,j,d) gespeichert, wobei der Integer i den Zeilenindex, der Integer j den Spaltenindex und der Double d den entsprechenden Eintrag symbolisiert. Implementieren Sie die obigen Methoden für sparse-Matrizen.

4. (GUIs, Textverarbeitung) Schreiben sie ein Applet, das auch als Applikation ausführbar ist, das eine Textdatei einliest (Dateiname und Pfad sind in eigenen Textfenstern innerhalb des Applets abzufragen) und alle Wörter, die mit Großbuchstaben beginnen, rot und alle anderen Wörter schwarz in ein Textfeld ausgibt. (Hilfe: Durchsuchen Sie die API nach einer Klasse, die geeignet ist, innerhalb des Textes die Schriftfarbe zu ändern, und für die Zerlegung des Textes verwenden Sie die Klasse `StringTokenizer`)
 (*) Schreiben Sie ein Rechtschreib-Kontroll-Program, wobei alle Wörter, die in der zusätzlich angelegten Wörterbuch-Datei nicht enthalten sind, rot geschrieben werden.
5. (Grafik, Animation, Threads) Implementieren Sie ein JApplet, in dem ein willkürlich wandernder aufgefüllter Kreis dargestellt wird, der während seiner Bewegung die Größe ändert. Der Hintergrund sollte ein Bild sein das aus einer Datei eingelesen wird. Beachten sie, daß das Neuzeichnen eines Applets oder einer Komponente durch die `draw` Methode langsam ist, also verwenden Sie möglichst die Klasse `BufferedImage`, um Teile Ihres Bildes zu speichern und wieder herzustellen.
 (*) Statt einen Kreis zu bewegen, zeichnen Sie den Bewegungsablauf eines Vogels (oder eines anderen Lebewesens bzw. einer Maschine), und verwenden Sie die gezeichneten Bilder, um eine sich bewegende Animation zu konstruieren.

Übungsbeispiele ALGODAT II

1. (Objekte, Reflexion) Entwerfen Sie ein Programm, das die Datei `Classes.txt` einliest, und alle Klassen analysiert, deren Namen in dieser Datei enthalten sind, diese dann instanziiert und drei ihrer Methoden aufruft (falls die Klasse nicht abstrakt ist). Während der Analyse sollten die Klasseneigenschaften (`public`, `protected`, `private`, `final`, `static`, `abstract` und `interface`), die Superklasse und Subklassen, die verwendeten Interfaces und alle Methoden ermittelt und ausgegeben werden. Die Klasse `Class` und das Package `java.lang.reflect` stellen Ihnen alle nötigen Methoden zur Verfügung.
 (*) Schreiben Sie ein Applet, das die ermittelten Methoden tabelliert und durch Anklicken mit vom Anwender angegebenen Parametern ausführt.
2. (Datenstrukturen, Listen, Klonen, Sortieren) Schreiben Sie eine `MyList` Klasse, die aus Objekten des Typs `Comparable` besteht. Die `MyList` Klasse sollte eine Methode `public MyList sort()` enthalten, die anhand dieser Liste eine generierte, sortierte

Liste zurückliefert. Die Objekte in der neuen Liste sollten nicht (!) mit denen von der Originalliste übereinstimmen sondern neue duplikate Instanzen sein. Zum Duplizieren verwenden Sie die Methode `clone` der Klasse `Object`. Zum Testen erstellen Sie eine von `Object` abgeleitete `Comparable` Klasse `MyObject`, die einen String enthält, und anhand dieses Stringes mit einen anderen `MyObject`-Objekten vergleichbar ist. Vergessen sie nicht, auch die `clone` Methode zu überladen, und eine weitere Methode `toString()` zu implementieren, die für die Textausgabe Ihres Objektes zuständig ist! Nun erzeugen Sie eine `MyList` mit 10 dieser Objekte, und rufen Sie `sort` auf. Die zurückgegebene Liste sollte nun lexikographisch sortiert sein, und wenn sie den String eines Elementes ändern, sollte die Originalliste unverändert bleiben. (*) Suchen Sie im Internet nach verschiedenen Sortiermethoden, finden Sie die Methode die Ihre Liste möglichst schnell sortiert und implementieren Sie sie.

3. (Datenstrukturen, Bäume) Ordnen Sie die Elemente der im vorigen Beispiel entworfenen `MyList` in einen binären Baum ein, und zwar so, dass der linke Knoten immer ein kleineres Element und der rechte ein größeres Element enthält (was kleiner bzw. größer heißt entscheidet noch immer die `compareTo` Methode des Interfaces `Comparable`). Wenn Sie alle Elemente der Liste eingeordnet haben, laufen sie Ihren Baum in der richtigen Reihenfolge durch und geben Sie eine aufsteigend und eine absteigend sortierte Folge der Elemente aus.
4. (Datenstrukturen, Graphentheorie) Entwickeln Sie ein als Applikation ausführbares `JApplet`, das einen maximal 20 mal 20 großes `JButton`-Feld enthält. Diese Buttons repräsentieren die Adjazenzmatrix eines ungerichteten Graphen. Die Buttons in der Diagonale sollten deaktiviert werden und alle anderen den Wert 0 oder 1 anzeigen. Durch Betätigung eines Buttons sollte sein Wert von 0 auf 1 bzw. umgekehrt geändert werden. Weil es sich um einen ungerichteten Graphen handelt, sollten alle Einträge symmetrisch sein, also wenn der (i,j)-te Eintrag auf 1 gesetzt wird, sollte auch der (j,i)-te Eintrag 1 werden. Nun erzeugen Sie einen ungerichteten Graphen anhand dieser Matrix. Die Klasse `Node` sollte einen Knoten und die von diesem Knoten ausgehenden Kanten repräsentieren, und die Klasse `Graph` sollte einen Startknoten namens `root` enthalten, zusammen mit einer `public String toString()` Methode, die den gesamten Graphen auf irgendeine leicht nachvollziehbare Weise ausgibt.
(*) Stellen Sie den Graphen graphisch dar.
(**) Finden sie einen Eulerpfad (!!!) (falls einer existiert) in diesem Graphen!
(***) Der Prozess, in dem der Eulerpfad gesucht wird, sollte auch graphisch verfolgt werden. Also markieren Sie alle bereits durchlaufenen Knoten rot und die noch nicht durchlaufenen blau. Der Suchprozess sollte entweder verlangsamt dargestellt werden, oder es sollte im Nachhinein eine Simulation konstruiert werden, damit man den Prozess visuell leicht verfolgen kann.
5. (Datenbanken) Entwickeln Sie eine Applikation, die mittels eines `JFrames` eine Buchdatenbank verwaltet. Authornamen bzw. Buchnamen sollten man in ein Textfeld eingeben können wobei 'wildcards', wie '*' erlaubt sind. Die Access Datenbank namens 'Books' sollte nach passenden Büchern durchsucht werden und die Ergebnisse in ein mit `JScrollPane` versehenes Textfeld ausgegeben werden. Die Datenbank 'Books.mdb' kann in Access eingebunden und unter dem URL 'jdbc:odbc:Books', mit Username 'anonymous' und password 'guest' mittels eines 'JdbcOdbc' Treibers verwendet werden.
6. (Servlets) (*) Schreiben Sie ein `HttpServlet`, um die im vorherigen Beispiel gegebene Datenbank, auf die gleiche Weise durch eine HTML Seite zu verwalten. Um Ihr Servlet

zu testen, müssen sie es in einen Application Server wie 'JBoss' oder 'Sun Java System Application Server' einbinden. Finden Sie heraus, wie das zu machen ist.

(**) Verwenden sie die 'Google Web API', um bei einer Abfrage nicht nur in Ihrer Datenbank sondern auch durch Google im Netz eine Suche zu starten und die Ergebnisse im einem gemeinsamen Format auszugeben.