Algorithmen, Datenstrukturen und Programmieren II

Ferenc Domes

16. Oktober 2005

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.

2. (GUIs, Textverarbeitung) Schreiben Sie ein JApplet, das auch als Applikation ausführbar ist und eine Textdatei einliest (Dateiname und Pfad sind in eigenen Textfenstern innerhalb des Applets abzufragen). Zusätzlich sollte eine Wörterbuch-Datei eingelesen werden, die Deutsch – Englische Wortpaare enthält. Durch betätigen eines `JButtons` sollte der eingelesene Text mit Hilfe des Wörterbuches übersetzt und alle im Wörterbuch nicht gefundenen Wörter rot markiert werden (Verwenden Sie ein `JTextPane`, um einen Text mit verschiedenen Schriftfarben darstellen zu können). Achten Sie darauf, dass Sie objektorientiert programmieren! Z.B. verwenden Sie neue oder existierende Klassen, um das Wörterbuch oder die Wortpaare darzustellen.
   (*) Die rot markierten Wörter sollten der Reihe nach nach ihrer Sprache und Bedeutung abgefragt und dann zum Wörterbuch dazugefügt werden.

3. (Frames, Listeners) Studieren Sie die WindowListener Klasse: Erstellen Sie eine (fast) unsterbliche Applikation! Diese Applikation soll ein Fenster öffen, das einen gezeichneten Smiley :-) zusammen mit einer Aufforderung das Fenster zu schliessen enthält. Sollte der User dieses Fenster schließen, öffnen sie das Fenster erneut an einer anderen Bildschirmposition. Bestimmen Sie eine 'geheime Stelle' auf Ihrem Smiley. Wenn der User mit der Maus auf diese geheime Stelle klickt, schließen Sie das Fenster endgültig und beenden Sie die Applikation.
   (*) Die 'geheime Stelle' kann nur durch eine gewisse Kombination der Maustasten betätigt werden: zB. rechte Taste, 2-mal linke Taste.

4. (Grafik, Animation, Threads) Implementieren Sie ein JApplet, in dem ein willkürlich wanderndes aufgefülltes Rechteck dargestellt wird, das 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 `paint` Methode langsam ist, also zeichen Sie nur Teile Ihres Bildes neu.
   (*) Statt einen Rechteck 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. (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.

2. (Objekte, Mathematik) Schreiben Sie eine Klasse names `Matrix`, die die wichtigsten Matrix-Operationen implementiert. Die Klasse soll allgemeine $m \times n$ Matrizen mint `double` Einträgen ermöglichen. Folgende Methoden sollten im `Matrix` 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) throws MatrixWrongIndexException
-----------------------------------------------------------------------
//Set the (c,r)th element of the matrix as d
public void   set(int c, int r, double d) throws
                MatrixWrongIndexException
-----------------------------------------------------------------------
//Matrix addition
public Matrix add (Matrix a) throws MatrixDimensionMismatchException
-----------------------------------------------------------------------
//Matrix addition (static)
public static Matrix add(Matrix a, Matrix b) throws
                MatrixDimensionMismatchException
-----------------------------------------------------------------------
//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) throws MatrixDimensionMismatchException
-----------------------------------------------------------------------
//Matrix multiplication (static)
public static Matrix mult (Matrix a, Matrix b) throws
                MatrixDimensionMismatchException
-----------------------------------------------------------------------
//Sub-matrix generated by deleting the c-th column and the r-th row
public Matrix submatrix(int c, int r) throws MatrixWrongIndexException
```

(*) 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. Dieses Format ist sinnvoll, um Matrizen mit sehr wenigen von 0 verschiedene Einträge speichersparend abzuspeichern.

3. (Rekursion, Mathematik) Ergänzen Sie die Klasse `Matrix`, mit einer zusätzlichen rekursiven Funktion `private static double det(Matrix d)`, die durch Verwendung den Laplaceschen Entwicklungssatzes die Determinante der Matrix `d` berechnet. Die Funktion `public double det()` sollte die obige rekursive Funktion verwenden um die Determinante der aktuellen Matrix zu bestimmen.
   Der Laplaceschen Entwicklungssatz lautet:
   Für $A \in \mathbb{R}^{n\times n}$ und $i \in 1 \dots n$ beliebig fix gewählt, gilt dass

$$\det A = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(A_{ij}).$$

   A_{ij} bezeichnet eine $(n-1) \times (n-1)$ Untermatrix von A, die durch das Streichen der $i$ten Zeile und $j$ten Splate entsteht. Der Ausdruck a_{ij} symbolisiert den (i,j)ten Eintrag von A.
   Den Index i können Sie beliebig wählen, und die Untermatrix A_{ij} bekommen Sie durch die schon im vorherigen Beispiel programmierte Methode `submatrix`.

4. (Objekte, Reflexion) Entwerfen Sie ein Programm, das die Datei `Classes.txt` einliest, und alle Klassen analysiert, deren Namen in dieser Datei enthalten sind. Wärend der Analyse sollten die Klasseneigeschaften (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.

5. (Objekte, Reflexion II) Instanzieren Sie die nicht abstrakte Klassen im vorherigen Beispiel und rufen Sie jeweils drei ihrer Methoden auf.

6. (Datenstrukturen, Listen) Implementieren Sie selbst eine doppelt verkettete Liste `MyList`. Die Daten der Knoten sollen Objekten des Typs `Comparable` sein. Entwickeln und testen Sie die Methoden für Einfügen, Entfernen und Verschieben von comparable Objekten in Ihrer Liste.

7. (Listen, Klonen) Um Objekte des Typs `MyList` kopieren (klonen) zu können schreiben Sie ein Copy-Konstruktor und eine Clone-Methode für die `MyList` Klasse.

8. (Listen, Sortieren) Die `MyList` Klasse sollte eine neue Methode `public MyList sort()` bekommen, die eine neu generierte (geklonte), sortierte Liste zurückliefert.
   Die Objekte in der neuen Liste sollten nicht (!) mit denen der Originalliste übereinstimmen, sondern neue, duplizierte (geklonte) Instanzen sein.

9. (Listen, Test) Testen Sie Bsp 6–8 dadurch, dass Sie eine `Comparable` Klasse `MyObject`, schreiben, die einen String entält, und anhand dieses Stringes mit anderen `MyObject` Objekten vergeichbar 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.

10. (Datenstrukturen, Generics) Lesen Sie 5 Strings in eine Datenstruktur ihrer Wahl ein. Löschen Sie diejenigen Strings, die mehr als 3 Buchstaben lang sind und kehren sie die Reihenfolge der übriggebliebenen Strings um. Achten Sie darauf dass Sie nur eine einzige Datenstruktur anlegen und außerdem keine dynamischen–casts verwenden dürfen.

11. (Datenverwaltung, Generics) Hopsi 12876 von der Kaninchenfarm B3 arbeitet für das IWVUE (Institut für Wissenschaft, Vermehrung und Entwicklung) und hat von seinem Chef Rowdy 3 als Auftrag die Entwickung einer Datenverwaltungssoftware bekommen, die Mitglieder der Kaninchen-Highsociety zu verwalten:
Zuerst sollte ein Menü mit den Optionen Kaninchen Dazufügen, Kaninchen Löschen und Kaninchen Auflisten erscheinen.
– Beim Hinzufügen soll man den Namen und die eindeutige Identifikationsnummer des Kaninchens erfragen, und falls noch kein Eintrag existiert, zum Datenbestand hinzufügen.
– Beim Löschen soll nach einem Namen oder nach einer Identifikationsnummer gefragt werden, und das entsprechende Kaninchen aus dem Datenbestand entfernt werden.
– Beim Auflisten soll entweder nach Namen oder nach Identifikationsnummer sortierte Liste aller bisher erfassten Kaninchen ausgegeben werden.
– Die Daten sollen natürlich gespeichert, und beim Neustart wieder initialisiert werden.
Hopsi hat eine große Familie aber leider keine Ahnung vom Programmieren, und Rowdy 3 ist ein sehr harter Vorgesetzter... Helfen Sie Hopsi 12876 aus der Patsche und entwickeln Sie die Datenverwaltungsoftware selbst mit Hilfe generischer Datenstrukturen.

12. (Datenverwaltung, Datenbanken) Rowdy 3 ist sauer: die Datenverwaltungssoftware funktioniert, ist aber mit dem Datenformat der AFEV (Abteilung für Futter, Ernährung und Vermehrung) nicht kompatibel! Modifizieren Sie Beispiel 11 und verwenden Sie ODBC, um die Kaninchen-Daten in einer SQL Datenbank zu speichern.

13. (Datenverwaltung, Datenbanken) Nun hat Hopsi 12876 (und natürlich auch Rowdy 3) viele Karotten mit Ihrer neuen Datenverwaltungssoftware gemacht... um die immensen Futter und Schulungskosten von Hopsis neuer Nachkommen Fressi 199999 zu decken, muss die Software weiterentwickelt werden:
–Die User-Interaktionen sollen durch ein Applet abgewickelt werden.
–Jedem Kaninchen soll noch eine der Kaninchenfarmen als Wohnort zugeordnet werden: eine neue SQL-Tabelle sollte die mögliche Kaninchenfarmidentifikatoren (A1-A12 und B1-B7) enthalten und jedes Kaninchen soll eine dieser als Wohnort bekommen.
* Realisieren Sie weitere vernünftige Erweiterungen, und fügen Sie zusätzliche Optische Elemente (Farben, Bilder, Animationen) dazu.

14. (Datenverwaltung, Servlets) Kehren Sie zu einer beliebigen funktionierenden Version der Kaninchen-Datenverwaltungssoftware zurück (Bsp 11-13) und realisieren Sie die User-Interaktionen (Hinzufügen, Löschen und Auflisten) mittels eines Servlets!

15. (Datenverwaltung, TCP) Ein Verbindungs-basierter Server soll alle Daten der Kaninchen-Highsociety verwalten. Die Clients sollen Applets sein und durch eine Eingabeleiste die

Befehle:

```
-- add (name) (id)      : Kaninchen mit Name 'name' und Id 'id' hinzufügen
-- del (name/id)        : Kaninchen mit Name 'name' oder mit Id 'id' löschen
-- list (spec)          : Kaninchen auflisten, 'spec' ist der Name oder die Id eines
                          Kaninchen wobei auch Wildcards erlaubt sind.
-- exit                 : beende Verbindung
```

auf dem Server ausführen können. Die Ergebnisse sollen in einer TextArea ausgegeben werden.
* Dem Client sollen die obigen Befehle durch Menüs, Buttons, Eingabefelder usw. erteilt werden

16. ***(Kaninchen Chat) Entwickeln Sie ein Kaninchen-Chat für die Elite der Kaninchen und machen sie Hopsi 12876 (und natürlich auch Rowdy 3) zu Karottenmilliadären!
◦ Zuerst sollen Sie die Kaninchen-Datenbank so erweitern, dass alle eingetragene Kaninchen ein Passwort haben.
◦ Dann sollen Sie einen UDP Chat-Server entwickeln. Zu diesem Server sollten die Clients durch das UDP Protokoll verbunden werden. Der Verbindungsaufbau sollte folgendermaßen ablaufen:

    (a) Handshake: der Client schickt ein Handshake-Paket zum Server, der jetzt seine IP-Adresse und Portnummer erfährt, und eine Aufforderung zur Authentifizierung zurückschickt.

    (b) Authentifizierung: Nun muss sich der Client authentifizieren: er schickt einen Kaninchennamen und ein (*verschlüsseltes) Passwort zum Server. Der Server überprüft die schon vorhandene Kaninchen Datenbank ob das angegebene Kaninchen existiert und sein Passwort korrekt ist. Falls ja, werden bis zur Beendigung der Session alle Pakete von dieser IP-Adresse und Portnummer akzeptiert. Wenn nicht, schickt der Server eine Nachricht zurück, dass die Authentifizierung fehlgeschlagen ist.

    (c) Einleitung: Wenn die Authentifizierung geklappt hat, schickt der Server an alle eingeloggten Kaninchen eine Nachricht dass sich das Mitglied nun im Chat-Netz befindet. Das Kaninchen, das sich gerade authentifiziert hat, bekommt eine Liste der Name aller Kaninchen, die sich zur Zeit im Chat-Netz befinden.

    (d) Chat: Nun kann man beliebige Texte an einzelne oder an alle im Chat-Netz vorhandenen Kaninchen senden: die Übertragung muss nicht verifiziert werden aber man sollte Texte beliebiger Länge übertragen können.

    (e) Logout: Wenn man nicht mehr weiter chaten will, schickt man eine Nachricht zum Server, dass man ausloggen will. Der Server akzeptiert nicht mehr beliebige Pakete von dieser IP-Adresse und diesem Port, außer man loggt sich wieder ein. Alle eingeloggten Kaninchen bekommen eine Nachricht, dass das Mitglied das Chat-Netz verlassen hat.

    ◦ Im Client sollte man sich authentifizieren können, und mit Hilfe einer ComboBox aussuchen können, an wen man Nachrichten schickt (auch an alle). Ein TextField sollte für die Texteingabe dienen und eine TextArea für die Ausgabe. Vergessen Sie nicht den Button für das Ausloggen!
    VIEL ERFOLG!