Algorithmen, Datenstrukturen und Programmieren I

Ferenc Domes

31. Mai 2006

Übungsbeispiele ALGODAT I

1. (Algorithmen, Struktogramme) Schreiben Sie einen Algorithmus zur Bedienung eines Kaffeeautomaten mittels eines Struktrogamm-Editors.
   • Fangen Sie mit einem einfachen Algorithmus an (wenige Schritte).
   • Stellen Sie sicher, dass die Bedingungen Endlichkeit, Eindeutigkeit, Interpretierbarkeit für den Algoritmus erfüllt sind.
   • Verfeinern Sie den Algorithmus durch Hinzufügen zusätzlicher Schritte.
   • Testen Sie wieder die obigen Bedingungen.

2. (Algorithmen, Struktogramme II) Erstellen Sie zwei Algorithmen zum Finden einer geeigneten Freundin bzw. Ehefrau oder eines geeigneten Freundes bzw. Ehemanns. Die Schritte in Aufgabe 1, sollten für beide Algorithmen erneut durchgeführt werden. Algorithums 1 sollte Ihrer Meinung nach einen sehr guten Weg zum Erlangen des Ziels darstellen, und Algorithums 2 einen völlig ungeeigneten.

3. (Klassen, Objekte) Erstellen Sie eine Klasse 'Kaninchen'. Es sollten mindestens 5 Attribute und 3 Methoden in der Klasse enthalten sein. Die Methoden und Attribute sollten einen kurzen Namen (max. 15 Buchstaben) haben, der ihre Aufgabe möglichst gut repräsentiert. Es sollte eine kurze Beschreibung der Aufgabe jeder Methode angegeben werden.

4. (Klassen, Objekte II) Zur Klasse 'Kaninchen' sollte eine Oberklasse 'Tier' erstellt werden und zu 'Tier' eine Oberklasse 'Lebewesen'. Verschieben Sie alle Attribute und Methoden der Klasse 'Kaninchen', die nicht nur zum Kaninchen, sondern auch zu anderen Tieren bzw. Lebewesen gehören, jeweils in die Klasse 'Tiere' bzw. in die Klasse 'Lebewesen'. Alle drei Klassen sollten nachher so ergänzt werden, dass sie jeweils mindestens 3 Attribute und 3 Methoden besitzen und alle 9 Attribute bzw. Methoden verschieden sind.

5. (Klassen, Objekte III) Die Klasse 'Mensch' sollte in die vorherige Struktur richtig eingeordnet werden, mit 3 verschienen neuen Attributen und Methoden.

6. (Java Grundlagen, Erste Schritte) Implementieren Sie eine Klasse 'Bsp06', die die 'Ausführungsmethode' `main` enthält. Die `main`-Methode sollte Ihren Namen, Ihre Matrikelnummer und Studienkennzahl in drei Zeilen auf dem Standardoutput ausgeben. Wenn Sie mit der Implementation fertig sind, übersetzen Sie Ihr Programm und testen Sie, ob es auch richtig funktioniert. Nun lesen Sie nochmal Ihren Sourcecode und

versuchen Sie jetzt ohne Benutzung von Hilfsmitteln wie Mitschrift oder Skriptum das ganze Programm noch einmal zu schreiben.

7. (Java Grundlagen, Erste Schritte II) Erstellen Sie eine Klasse 'Bsp07', die eine `main` Methode besitzt (ohne Mitschrift oder Skriptum). Kopieren Sie die auf meiner Homepage zum Download freigegebene Class-Datei `Read.class` in dasselbe Verzeichnis, wo sich die von Ihnen geschriebene Bsp07.java Datei befindet.
   Nun können Sie in der `main` Methode von Bsp07 die Befehle:
   • `int i=Read.integer()` und `int i=Read.integer('...')` für das Einlesen eines Integers,
   • `double d=Read.number()` und `double d=Read.number('...')` für das Einlesen einer Double-Zahl,
   • `String str=Read.text()` und `String str=Read.text('...')` für das Einlesen eines Strings verwenden. Der Ausdruck `'...'` repräsentiert immer eine String Konstate.
   Verwenden Sie die aufgelisteten Befehle, um einen Namen, eine Matrikelnummer, eine Studienkennzahl und eine beliebige Zahl zwischen 0 und 1 zu erfragen. Speichern Sie die Antworten in Variablen `name`, `skz`, `mn` und `rat`. Geben Sie schließlich folgenden Satz: 'Der Student (`name`) hat die Studienkennzahl (`skz`) und die Matrikelnummer (`mn`), und hat die Zahl (`rat`) geraten.

8. (Java-Grundlagen, Operatoren) Berechnen Sie am Papier (Schreibtischtest) die Werte der Variablen `i`, `j`, `k`, `zahl`, `test1` und `bitv` anhand des Codes:

```
int i=1;
int j=(int)8.9;
boolean test1= (--i<=1) || (j != 8.9);
boolean test2= !test1;
int k=j<<i;
i=k++;
j=--k*i;
double zahl=(test1 || (j>-5)) ? 5.99:-88;
test2=test1 && (zahl != -88);
int bitw=(--i<=6) ? 32:15;
bitw%=2;
bitw=~bitw;
```

   und implementieren Sie ein Programm (Bsp08), das die obige Operationen ausführt und die gesuchten Werte ausgibt. Vergleichen Sie Ihre Ergebnisse und die Resultate, die vom Programm erzeugt wurden. Falls diese nicht übereinstimmen, suchen Sie so lange nach Fehlern in ihrem Schreibtischtest, bis die Ergebnisse identisch sind.

9. (Java Grundlagen, Operatoren II) Erweitern Sie das vorherige Beispiel so, dass die Zahl `i` und die Zahl `j` mittels `Read.class` eingelesen und alle Resultate anhand der eingelesenen Werte berechnet werden.

10. (Java Grundlagen, Bedingugen) Schreiben Sie 'Bsp10', wo mittels `Read.class` Name, Geschlecht, Anzahl der Kinder eingelesen werden, und erzeugen Sie die Ausgabe:
    ('Herr'/'Frau') (`name`) ' hat ' ('keine'/'einen'/'mehrere' /'einen Haufen'/'eine Horde von' ...) ' Kind(er(n)).'.
    Wie Sie Name, Geschlecht und Anzahl der Kinder intern speichern, ist Ihnen überlassen, sowie die Grenze, ab der Sie Anzahl der Kinder als 'mehrere', 'einen Haufen', etc.

klassifizeren. Verwenden Sie auch die Mehrfach-Bedingung (`switch`), nicht nur einfache Bedingungen.

11. (Java Grundlagen, Schleifen) Das Programm 'Bsp11' sollte zuerst nach einer ganzen Zahl `n` fragen. Wird diese eingegeben, sollten genau so viele neue ganze Zahlen eingelesen werden. Nach jeder Eingabe sollten so viele Sternchen ausgegeben werden, wie die Größe des Absolutbetrages der neu eingelesene Zahl. Wenn alle `n` Zahlen gelesen wurden, sollten Sie sich höflich beim Anwender bedanken, dass er Ihnen seine kostbare Zeit gewidmet hat.

12. (Java Grundlagen, Arrays) Modifizieren Sie 'Bsp11' so, dass zuerst alle Zahlen eingelesen werden (in ein Array) und danach alle Sternchen ausgegeben werden.
(*) Rotieren Sie die so entstandene Figur um 90 Grad (links oder rechts) und geben Sie auch die rotierte Version aus. Dazu verwenden Sie am besten ein zweidimensionales `char` Array, mit Einträgen '*' oder ' '.

13. (Java Grundlagen, Bedingugen, Schleifen, Zufallszahlen) 'Bsp13' : verwenden Sie den Befehl `int zzahl= (int)((max-min+1)*Math.random()+min)`, um 20 Zufallszaheln im Intervall [`max`, `min`] zu erzeugen. Speichern Sie diese in ein geeigentes Array. Vergessen Sie nicht das Package '`Math.*`' zu importieren. Geben Sie jetzt alle Zahlen aus, die kleiner als der Mittelpunkt des verwendeten Intervalles [`max`, `min`] sind.

14. (Java Grundlagen, Sortieren) Kopieren Sie 'Bsp11' in 'Bsp12' und sortieren Sie das Array der Zufallszahlen nach dem Betrag durch geschicktes Vertauschen der Elemente, und geben Sie die sortierten Zahlen aus.

15. (Objekte, Tierfreunde) Das Kaninchen ist nun zu implementieren! Bauen Sie die Klassenstruktur Lebewesen, Tier, Kaninchen, Mensch. Alle Attribute, Eingabetypen und Rückgabetypen sollten entweder primitiv sein, oder benutzen Sie die Java API, um eine vom Namen her passende Klasse zu finden (die Funktionalität muss nicht beachtet werden). Eine Alternative für die diversen Farben wäre z.B. die Klasse 'Color'. Alle Klassen und Methoden sollten 'abstract' sein. Das heißt, sie sollten keinen Code enthalten. Die ganze Struktur sollte ohne Fehler zu Compilieren sein.
(*) Sie können auch neue Klassen entwickeln, die als Typ für die Attribute oder Typ der Ein- bzw. Rückgabewerte diverser Methoden dienen.

16. (Objekte, Kaninchen II) Die Kaninchen sollten nun zum Leben erweckt werden.
• Die Klasse Kaninchen sollte keine abstrakte Methoden mehr enthalten. Jede Methode sollte implementiert werden, auch die von den Oberklassen geerbten Methoden sollten entweder in der Oberklasse implementiert werden, oder in der Oberklasse abstract gelassen und dafür in Kaninchen implemetiert werden.
• Wenn es Methoden gibt, für deren Implementierung neue Atrribute bzw. Methoden notwendig sind, fügen Sie sie ein.
• Zu komplexe (für Sie nicht implementierbare) Methoden können weggelassen werden.
• Die Methode

```
public String toString()
```

sollte alle relevanten Attribute des Kaninchens in einem String (text) zurückliefern.
• Ein (oder mehrere) Konstruktoren des Kaninchens sollten geschrieben werden.

• Das Programm 'KaninchenTest' sollte nun ein Kaninchen erzeugen, seine Beschreibung mittels `toString()` ausgeben, und sämtliche verfügbaren Methoden testen.
(*) Versuchen Sie, komplexe und vielfältige Methoden zu implemenieren, wie zB.

```
public int[] bewege (int richtung, int distanz) bzw.
public Point bewege (int richtung, int distanz).
```

17. (Objekte, Mehr Kaninchen) Überprüfen Sie die Kaninchen nochmal:
• Sind alle Attribute bzw. Methoden in die Klassenstruktur auch richtig verteilt?
• Ist die Kapselung richtig? Sind die nach aussen nicht relevanten Attribute und Methoden wirklich `private`?
• Haben Sie an die Vortplanzung gedacht? Kaninchen sollten sich auch vermehren können:

```
public Kaninchen paare (Kaninchen opfer)
```

Welche Eigenschaften der Nachkömmling hat, kann von den Eigeschaften der Eltern abgeleitet werden, kann aber auch wilkürlich sein. Testen Sie auch das Geschlecht. Wenn sich zwei männliche bzw. zwei weibliche Kaninchen zu paaren versuchen, wird's ja nichts... also in diesen Fall liefern Sie eine `null` Referenz zurück.

18. (Objekte, Kaninchen Überschuss) Mehr Lebensraum für Kaninchen! Wir wollen mehr Platz, Bio-Futter, Schutz vor Wölfen ...! — protestieren Ihre neu erzeugte Kaninchen. Nun ist es höchste Zeit eine Kaninchen–Farm einzurichten. Die Klasse 'KaninchenFarm' sollte maximal 1000 Kaninchen beherbergen können (in einem Kaninchen Array) und mindestens die folgenden Methoden enthalten:

```
public KaninchenFarm()  //Default Konstruktor
public String toString() //Text representation des auktuellen Zustands
                           ihrer Farm
public void add(Kaninchen neukoemmling) //Ein Kaninchen zur Farm
                                          hinzufügen (wenn noch Platz ist).
public Kaninchen kill(int num)            //Kaninchen nummer 'num' wird
                                          von der Farm "entfernt".
public void vermehre()   //Alle Kaninchen suchen sich wilkürlich einen
                           Partner (aber nur einen) und paaren sich.
```

'Willkürliche Partnersuche' heisst, man sucht zu jedem Kaninchen einen zufälligen Partner (siehe die Aufgabe über Zufallszahlen) und verwendet die Methode paare auf die beide Exemplare; wenn die nicht gleichgeschlechtlich waren, entsteht ein neues Kaninchen.
Das Programm 'KaninchenFarmTest' sollte eine Farm mit 5 weiblichen und 5 männlichen Kaninchen besiedeln, die sich 100mal paaren lassen, und beobachten Sie wie sich die Farm entwickelt.
(*) Die harte Realität: jede Farm besitzt eine gewisse Menge an Futter. Jedes Mal, wenn sich zwei Kaninchen paaren, egal ob erfolgreich oder nicht, essen sie eine Portion Futter. Wenn die Paarung abgeschlossen ist, wächst das Futter wieder nach, in Abhängigkeit der Futterreserven. Wenn ein Kaninchen kein Futter bekommt, stirbt es. Allerdings wird sich auch die Rate, wie sich die Futterreserven nach der nächste Paarung erholen, um einen geringen Prozentsatz erhöht (Düngung).

• Die Futterreserven sollten nach unten und nach oben begrenzt werden.
• Der durch Kaninchenexitus auftretender 'Futterwachstum–Bonus' sollte in eine Variable gespeichert, jedes Jahr verwendet, gelöscht und neu berechnet werden.
• Testen Sie die Ergebnisse bzw. verfeinern Sie das Modell.

19. (Strings, Sortieren II) Lesen Sie 5 Sätze (Strings) ein, speichern Sie sie in ein Array, dann geben sie folgende Informationen über jeden Satz aus:
• Länge des Satzes,
• Anzahl der Grossbuchstaben,
• Anzahl der Kleinbuchstaben,
Ersetzen sie nun alle Sätze durch die kleingeschriebe Version (API), und verwenden Sie die Klasse `StringTokenizer`, um jeden Satz in einzelne Wörter zu zerlegen. Die so erhaltenen Worte sollen in ein neues Array gespeichert werden. Verwenden Sie nun die `compareTo` Methode der Klasse String, um dieses Array zu sortieren. Das sortierte Array soll schließlich ausgegeben werden.

20. (Strings, Applets) Implementieren Sie die vorherige Aufgabe in einen `JApplet`:
• Ein Text beliebige Länge soll in ein `JTextArea` eingelesen werden und mittels `String Tokenizer` in einzelne Sätze zerlegt werden.
• Wenn der `JButton` 'Analysiere' gedrückt wird, sollten alle eingegebenen Sätze wie im Bsp 19 analysiert werden.
• Die Ergebnissen, sollten in einem deaktivierten `JTextField` mit roter Frabe ausgegeben werden.
• Einr weitere `JButton` 'Löschen' sollte den bisher eingegebenen Text von der `JTextArea` löschen.
• Schreiben Sie eine HTML-Seite, die Ihr `JApplet` auch in einen Web-Browser ausführbar macht.
(*) Versuchen Sie die Ergebnisse in ein `JTextPane` auszugeben, wobei die Texte schwarz und die Zahlen rot ausgegeben werden sollten. Als Anleitung siehe die Algodat II WS 2005/06 'Hilfsinformationen' auf mein Homepage.

21. (Strings, Applets II) Bsp 20 sollte weiter ergänzt werden. Das `JApplet` sollte nun auch als Applikation ausführbar sein, also implementieren Sie die entsprechende `main` Methode. Als Layout verwenden Sie ein`BorderLayout`. Das JTextField für das Ergebnis und die `JButtons` setzen Sie nach Norden, die JTextArea ins Zentrum und im Süden sollte ein neues leeres `JLabel` eingefügt werden. Verwenden Sie `MouseMotionListener`, um die aktuelle Mousekoordinaten auf dieses neue JLabel auszugeben.

22. (Applets, Layout, Buttons) Ein `JApplet` mit einem `BorderLayout` sollte die folgende Komponente enthalten:
• Ein `JPanel` im Norden mit einen `GridbagLayout`. Darauf sollten folgende Komponenten geklebt und schön geordnet werden:
-ein `JLabel` 'Zeilenzahl:',
-ein `JTextField` für die Eingabe der Zeilenzahl (`m`),
-ein `JLabel` 'Spaltenzahl:',
-ein weiteres `JTextField` für die Eingabe der Spaltenzahl (`n`),
-ein `JButton` 'erzeugen'.
• Wenn der `JButton` 'erzeugen' gedrückt wird, sollte im Zentrum ein `JPanel` mit einem `m` mal `n` `GridLayout` erzeugt werden. Auf diesem `JPanel` sollten `m*n` durchnumerierte `JButtons` plaziert sein.
-Drückt man einen `JButton` mit gerader Nummer, werden dieser und alle benachbarten

Buttons deaktiviert.
-Drückt man einen `JButton` mit ungerader Nummer wird die Farbe seiner Zahl auf rot geändert und alle benachbarten Buttons aktiviert.
-Drückt man einen `JButton` mit ungerader Nummer und roter Farbe werden alle deaktivierte Buttons aktiviert und alle aktivierte Buttons deaktiviert. Weiters setzt man alle Farben wieder auf schwarz.
• Im Süden soll ein `JButton` sein mit einem beliebigen Bild darauf. Dazu erzeugen Sie einen `ImageIcon` mit dem Konstruktor `(String filename)` und dann benutzen Sie dieses `ImageIcon` im Konstruktor `(String text, Icon icon)` vom `JButton`, um den gewünschten Button zu erhalten.

23. (Zeichnen, Frames) Entwickeln Sie einen `JFrame` mit einen `BorderLayout`. Im Norden sollte eine Gruppe von `RadioButtons` eine Selektion diverser Farben ermöglichen. Weiterhin sollte ein `JMenu` mit mindestens 3 zeichenbaren Formen wie Kreis, Rechteck, Linie usw. erzeugt werden. Im Süden sollte die Mouseposition auf einem `JPanel` angezeigt werden. Wenn der `JButton` 'zeichen' gedrückt wird, sollte die im `JMenu` ausgesuchte Form in die selektierte Farbe auf einen im Zentum eingefügten `JPanel` gezeichnet werden. Testen Sie Ihren neues Frame in einer Testklasse.

24. (Zeichnen, Frames, Schon wieder diese Kaninchen!) Verwenden Sie einen Zeichenprogramm, um zwei kleine Kaninchen zu zeichen. Kaninchen 1 sollte blau sein und Kaninchen 2 rosa. Erzeugen Sie eine von `JFrame` abgeleitete Klasse namens 'KaninchenSpielplatzJFR':
• Der Konstruktor sollte die gewünschte Anzahl der blauen (`m`) und die Anzahl der grünen Kaninchen (`n`) bekommen und in Klassenvariablen speichern.
• Legen Sie ein Array `pos` und ein Array `dest` der Länge `n+m` vom Typ `Point` an (siehe die API für die Beschreibung der Klasse `Point`). Das Array `pos` repräseniert die Koordinaten diverser Kaninchen und das Array `dest` die Zielkoordinate, wohin die Kaninchen sich später bewegen sollten. Am Anfang initialisieren Sie alle Points mit (0,0).
• In zwei Klassenvariablen `final hpath1` und `final hpath2` sollten die Pfade zu den von Ihnen gezeichneten Kaninchenbilder gespeichert werden. Beachten Sie dabei, dass in einem String das Backslash-Zeichen als doppelter Backslash geschrieben werden muss.
• Die Methode `setPos(int k, Point p)` soll die Position des $k$ten Kaninchen zu p, und die Methode `setDest(int k, Point p)` sol die Zielkoordinate des $k$ten Kaninchen zu p setzen.
• Die `paint`-Methode sollte nun auf die ersten `m` in `pos` gespeicherte Koodinaten die blauen und auf die letzten `n` Koodinaten die rosa Kaninchen zeichen. Zum Zeichen verwenden Sie die Methode `drawimage` der Klasse `Graphics`. Um ein Image für beide Kaninchen zu bekommen, erzeugen Sie zwei Klassenvariablen `Image himage1,himage2` und im Konstruktor initializieren Sie mittels:
`himage1=(new ImageIcon(hpath1)).getImage();` bzw.
`himage2=(new ImageIcon(hpath2)).getImage();`
• Zum Testen schreiben Sie die ausführbare Klasse 'KaninchenSpielplatzTest'. Erzeugen Sie einen 'KaninchenSpielplatzJFR' mit einem blauen und zwei rosa Kaninchen, setzen Sie die Positionen und lassen diese zeichen.

25. (Zeichnen, Frames, Kaninchen Romanze!) Entwickeln Sie Ihren 'KaninchenSpielplatzJFR' weiter:
• Zeichen Sie ein rotes Herzchen, und erzeugen Sie eine neue Klassenvariable für sein

`Image` genau so wie im vorigen Beispiel, das für die Kaninchenbilder gemacht wurde.
• Die Methode `moveAll()` sollte alle Kaninchen von ihrer akutuellen Position zu ihrer Zielposition bewegen... die neue Position sollte jetzt die Zielposition werden. Die Bewegung kann aus einen einzigen 'Sprung' bestehen oder (*) kontinuierlich sein.
• Definieren Sie eine neue Klassenvariable `distanz`, die durch den Konstruktor übergeben wird.
• Die Methode `moveAll()` sollte nachdem sich alle Kaninchen bewegt haben, die Entfernung aller blauen und rosa Kaninchen testen: wenn ein blauer und ein rosa Kaninchen eine Entfernung kleiner als die Variable `distanz` hat, sollten Sie das von Ihnen erzeugte Herz zwischen diese zwei Kaninchen zeichnen. Eine zusätzliche Bedingung ist dass jedes rosa bzw. blaues Kaninchen nur einen einzigen 'Liebling' pro Bewegung haben kann. Die Anzahl der gezeichneten Herzchen soll vermerkt und von der Methode zurückgeliefert werden.
• Testen Sie ein paar 'Sprünge' Ihrer blauen und Ihrer zwei rosa Kaninchen so, dass Sie am Anfang zufällige start- und Zielkoordinaten setzen und nach jede bewegung für alle Kaninchen wieder neue Zielkoordinaten festlegen. Testen Sie, ob die Herzchen richtig gezeichet werden und die Methode `moveAll()` auch die genaue Anzahl der Herzen zurückliefert.

26. (**)(Kaninchenfarm-Simulation) Vereinen Sie Aufgabe 18 und 25! Ihre Kaninchenfarm sollte nun in einem JApplet mit einem `BorderLayout` dargestellt werden:
-Sie können 'KaninchenSpielplatzJFR' im Zentrum platzieren.
-Die weiblichen Kaninchen werden durch die rosa Kaninchenbilder repräsentiert und die männlichen durch die blauen.
-Neue Methoden vom 'KaninchenSpielplatzJFR' wie `addKaninchen(int blau, int rosa)` und `removeKaninchen(int nummer)` müssen entwickelt werden, um das Populationswachstum zu simulieren.
-Im Norden und Süden können Buttons wie z.B. 'Nächste Phase' oder Textfelder plaziert werden, wo die aktuellen Anzahl der weiblichen und männlichen Kaninchen bzw. die aktuelle Futterreserven ausgegeben werden.
-Lassen Sie Ihre Phantasie spielen und verfeinern Sie die graphische Darstellung und die Simulation so weit, wie sie es wollen.
-Eine geistreiche und korrekte Lösung dieser Aufgabe garantiert eine sehr gute Übungsnote.