Technische Universität Darmstadt
Fachbereich Informatik
Prof. Dr. Andreas Koch

Allgemeine Informatik 1 im WS 2007/08

Übungsblatt 12 (Wiederholung, Anwendung)

Bearbeitungszeit: ab 28.01.2008

Hinweis:
Diese recht umfangreiche Übung soll auch zur Klausurvorbereitung dienen. Wir erwarten nicht, dass Sie innerhalb einer Woche fertig werden. Die Musterlösung erscheint rechtzeitig vor der Klausur.

Klausur

Die Klausur Allgemeine Informatik 1 im Wintersemester 2007/2008 findet am Donnerstag, den 28. Februar 2008 zwischen 9:00 und 11:00 Uhr statt. Unabhängig von einer Anmeldung im Prüfungssekretariat müssen Sie sich auch bei uns über das WebReg-System anmelden, einen Link zur Anmeldung finden Sie auf unserer Webseite. Die Saalverteilung der Klausur wird dann auf der Webseite bekanntgegeben.

Aufgabe 1: `for`-Schleife

Neben der **`while`**-Schleife gibt es in Java noch die **`for`**-Schleife. Sie hat folgende Form:

```
for (Initialisierung; Bedingung; Weiterschaltung) {
  // ...
}
```

Diese **`for`**-Schleife:

```
for (int i = 0; i < 5; i++) { // Init, Bedingung, Weiterschaltung
  System.out.println(i);      // Anweisung
}
```

hat den gleichen Effekt wie diese **`while`**-Schleife:

```
int i = 0;                  // Initialisierung
while (i < 5) {             // Bedingung
  System.out.println(i); // Anweisung
  i++;                      // Weiterschaltung
}
```

Mit beiden Schleifen werden untereinander die Zahlen 0, 1, 2, 3 und 4 auf dem Bildschirm ausgegeben.

Eine **`for`**-Schleife lässt sich also immer in eine gleichbedeutende **`while`**-Schleife übersetzen, ist allerdings etwas kompakter zu schreiben.

Erzeugen Sie in einem **task** einen **Robot karel** bei (1, 1) mit 999 Beepern und Blickrichtung Osten. **karel** bleibt in der ersten Street und soll mittels einer **for**-Schleife in der ersten Avenue zwei Beeper legen, in der zweiten Avenue vier, in der dritten Avenue sechs und so weiter, bis zu 18 Beepern in der neunten Avenue. Sie können zum eigentlichen Ablegen der Beeper innerhalb der **for**-Schleife eine **loop**-Schleife oder gerne auch eine zweite **for**-Schleife verwenden.

Aufgabe 2: Arrays

Arrays (Felder) sind vergleichbar mit mathematischen Vektoren. Sie bieten die Möglichkeit, mehrere Variablen unter einem gemeinsamen Namen in Verbindung mit einem Index anzusprechen.

Ist z.B. mathematisch $a = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$, so ist $a_1 = 2$, $a_2 = 3$ und $a_3 = 5$.

In Java fängt der Index allerdings nicht bei 1 an, sondern bei 0, und er steht in eckigen Klammern hinter dem Namen des Arrays: **a[0] = 2** weist dem ersten Array-Element den Wert 2 zu.

Die Deklaration eines Arrays sieht ähnlich aus wie die einer normalen Variablen:
datentyp[] arrayname;
Also wird ein Array **a**, das **int**-Werte aufnimmt, mit **int[] a;** deklariert, und ein Array, das Roboter beinhaltet, z.B. mit **Robot[] roboter;**.

Die Initialisierung eines Arrays weist Parallelen zur Erzeugung eines Roboters auf: sie geschieht mit **new**. Dabei wird in die eckigen Klammern die Anzahl der Elemente des Arrays geschrieben:
a = new int[3]; erzeugt ein Array mit den drei Elementen **a[0]**, **a[1]** und **a[2]**. Diese Elemente sind allerdings noch nicht initialisiert und müssen wie jede andere Variable auch erst initialisiert werden. Mit **a[0] = 2; a[1] = 3; a[2] = 5;** entsteht z.B. ein Array, das dem Vektor aus dem obigen Beispiel entspricht.

Mit dieser **for**-Schleife können wir die Elemente des Arrays **a** untereinander auf dem Bildschirm ausgeben:

```
for (int i = 0; i < 3; i++) {
  System.out.println(a[i]);
}
```

Da wir nicht immer wissen, wie viele Elemente das Array hat, benutzen wir im Allgemeinen in der Bedingung der **for**-Schleife nicht einen festen Wert, sondern **arrayname.length** (im obigen Beispiel **a.length**) – diese Variable enthält immer die Anzahl der Elemente des Arrays. Die Elemente eines Arrays a sind also von **0** bis **a.length – 1** nummeriert.

Erweitern Sie nun die auf der Webseite vorgegebene Datei **uebung12-2.task**. In ihr wird ein Array **a** einer zufälligen Länge erzeugt und mit zufälligen Werten gefüllt.

Benutzen Sie im markierten Bereich eine einzige **for**-Schleife, um folgendes zu tun:

- die einzelnen Arrayelemente untereinander auf dem Bildschirm auszugeben
- die Summe der Werte in der Variablen **summe** zu berechnen
- die Position des maximalen Wertes zu finden und in der Variablen **maxi** zu speichern (dazu müssen Sie den aktuellen Wert mit dem an der Position des bisher maximalen Wertes vergleichen).

Sie helfen sich, wenn Sie zuerst versuchen, die Schleife auf dem Papier zu entwerfen und nachzuvollziehen – in der Klausur haben Sie auch keinen Rechner!

Die Ausgabe der errechneten Werte auf dem Bildschirm ist schon in der Vorgabe enthalten.

Aufgabe 3: Variablen oder Arrayelemente vertauschen

In vielen Fällen ist es nötig, den Inhalt von Variablen vertauschen zu können.

Angenommen, die Variablen a und b wurden folgendermaßen initialisiert:

```
int a = 42;
int b = 23;
```

Das Vertauschen funktioniert dann nicht einfach mittels

```
a = b;
b = a;
```

Dann hätten nämlich beide Variablen den Wert 23. Sie benötigen eine dritte Variable als Hilfe!

Angenommen, wir wollen in einem Array die Elemtene **`a[i]`** und **`a[j]`** vertauschen, wenn **`a[i]`** kleiner als **`a[j]`** ist. Wie sieht der entsprechende Code aus?

```
if (a[i] < a[j]) {
    _____________________________
    _____________________________
    _____________________________
}
```

Aufgabe 4: Sortieren mit SelectionSort

Gegeben ist eine Datei **uebung12-4.task** (auf unserer Webseite), die Sie erweitern sollen **(also keine eigene Datei anfangen!)**.

Diese Aufgabe verbindet die beiden letzten Aufgaben zu einer sinnvollen Anwendung: einem Sortieralgorithmus, genauer den Algorithmus **SelectionSort**. Die Informatik kennt noch viele weitere, oft bedeutend effizientere (aber auch kompliziertere) Sortieralgorithmen – auf www.sortieralgorithmen.de können Sie mehr darüber erfahren.

In der vorgegebenen Datei wird wieder ein Array **`a`** zufällig erzeugt und auf dem Bildschirm ausgegeben. Ihre Aufgabe ist es nun, dieses Array aufsteigend zu sortieren. Aufsteigend bedeutet, dass der kleinste Wert des Arrays nach dem Ordnen in **`a[0]`** steht und der größte in **`a[a.length - 1]`**.

a) Implementieren Sie die Methode **`print`**. Diese soll das Array in einer Zeile und nicht untereinander ausgeben. Dazu geben Sie in einer **`for`**-Schleife jedes Element mit **`System.out.print()`** aus, gefolgt von einem Leerzeichen. Nach der Schleife rufen Sie noch einmal **`System.out.println();`** auf, um in die nächste Zeile zu springen.

Unser Sortieralgorithmus funktioniert folgendermaßen:

1. Am Anfang ist das Array komplett von Index **`0`** bis Index **`a.length - 1`** unsortiert.
2. Wir suchen das größte Element und vertauschen es mit dem letzten Element. Dadurch geht der unsortierte Teil nur noch von **`0`** bis **`a.length – 2`**, das letzte Element ist jetzt sortiert.
3. Nun suchen wir im unsortierten Teil das größte Element und vertauschen es mit dem letzten Element im unsortierten Teil des Arrays. Der unsortierte Teil geht nur noch bis **`a.length – 3`**, die letzten beiden Elemente sind jetzt sortiert.
4. Da in jedem Schritt der sortierte Teil des Arrays um eins größer wird, reichen **`a.length – 1`** Schritte, um das Array komplett zu sortieren.

Index	***0***	***1***	***2***	***3***	***4***	***5***
Anfang	42	23	**66**	7	11	**13**
nach Schritt 1	**42**	23	13	7	**11**	66
nach Schritt 2	11	**23**	13	**7**	42	66
nach Schritt 3	11	7	**13**	23	42	66
nach Schritt 4	**11**	**7**	13	23	42	66
nach Schritt 5	7	11	13	23	42	66

weiß: unsortiert
grau: sortiert
unterstrichen: diese Elemente werden vertauscht

b) Implementieren Sie nun die Methode **`sort`**. Sie benötigen dazu zwei verschachtelte **`for`**-Schleifen. Der Index der äußeren Schleife entspricht dem Ende des unsortierten Teils des Arrays und fällt von **`a.length – 1`** auf **`1`**. In dieser äußeren **`for`**-Schleife benutzen Sie wie in Aufgabe 2 eine weitere **`for`**-Schleife, um den Index des größten Elements zu finden und vertauschen dieses dann wie in Aufgabe 3 mit dem letzten Element des ungeordneten Teils.

Aufgabe 5: Osterdatum

Bis Ostern dauert es zwar noch, aber wie lange genau dauert es eigentlich noch? Und wann ist nächstes oder übernächstes Jahr Ostern? Das lässt sich herausfinden…

Implementieren Sie in der vorgegebenen Datei **uebung12-5.task** den Algorithmus, der zu einem gegebenen Jahr das Datum des Ostersonntags im Format „Ostersonntag: Tag.Monat.Jahr“ auf dem Bildschirm ausgibt.

Den Algorithmus sowie weitere Informationen und Links finden Sie im Internet auf http://de.wikipedia.org/w/index.php?title=Osterdatum&oldid=25787488.

Die beiden Zeilen für den Julianischen Kalender müssen Sie natürlich weglassen. Achten Sie darauf, dass alle Zahlen vom Typ **`int`** sind. Der Modulo-Operator wird in Java durch das Prozentzeichen dargestellt: „a mod b“ entspricht in Java **`a % b`**.

Aufgabe 6: Scope von Variablen

Betrachten Sie das folgende Programm genau. Nehmen sie an, dass mit **`Tester test = new Tester (42, 23);`** eine Instanz dieser Klasse angelegt wird und anschließend mit **`test.m();`** aufgerufen wird. Welchen Wert haben die einzelnen Attribute von **`test`** nach dem Ende des Aufrufs?

<u>Achtung:</u> Diese Aufgabe mit Hilfe der KarelJIDE zu lösen ist nicht gestattet! In Ihrem eigenen Interesse - halten Sie sich bitte daran. In der Klausur hilft Ihnen auch kein PC!
Sie können allerdings Ihre auf dem Papier entstandene Lösung mit der KarelJIDE verifizieren.

```
class Tester {

  int x, y, z = 0;

  public Tester(int a, int z) {
    x = a;
    y = z;
    z = x + y;
  }

  public void m() {
    int v, w, x = 19;
    v = f(3) * z;
    w = y / x – f(f(z + x));
    x = x + f(y);
    y = x – v;
    z = f(v) – f(w) + f(z);
  }

  public int f(int x) {
    return z / y + x;
  }

}
```

Aufgabe 7: Restklassenverfahren

Wie Sie wissen, gibt es nicht nur das Dezimalsystem, sondern auch andere Zahlensysteme. So ist die Zahl 255_{10} im Dezimalsystem äquivalent zu 11111111_2 im Binär-, 100110_3 im Ternär-, 377_8 im Oktal- und FF_{16} im Hexadezimalsystem.

Ihre Aufgabe ist es nun, gemäß folgenden Vorgaben ein Programm zu schreiben, das Zahlen aus dem Dezimalsystem in ein beliebiges System zur Basis 2 bis 36 umwandeln und auf dem Bildschirm ausgeben kann. Lesen Sie sich die komplette Aufgabe durch, bevor Sie anfangen!

a) Erstellen Sie eine Roboterklasse **`Converter`** mit zwei Attributen **`int base`** und **`String digits`**.

b) Schreiben Sie einen Konstruktor, der nur einen Parameter **`int base`** erwartet.
Erzeugen Sie zuerst mittels **`super`** einen beliebigen Roboter erzeugt werden und füllen Sie **`digits`** mit **`„0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ“`**.
Ihr Programm soll dann überprüfen, ob der Parameter im erlaubten Bereich [2, 36] liegt.
Falls nein, soll auf dem Bildschirm eine entsprechende Fehlermeldung ausgegeben werden und das Klassenattribut **`base`** mit 10 initialisiert werden.
Falls ja, soll das Attribut mit dem übergebenen Parameter initialisiert werden.

Hinweis: *auf das Klassenattribut müssen Sie mit* ***`this.base`*** *zugreifen, sonst greifen Sie auf den Parameter zu!*

c) Schreiben Sie eine Methode **`convert`**, die einen Parameter **`int number`** erwartet und einen **`String`** zurückgibt. Erzeugen Sie zuerst einen leeren String. Mit dem im der Vorlesung (und im Skript „Zahlen, Zeichen und Texte“) kennen gelernten Restklassenverfahren berechnen Sie Ziffer um Ziffer die Zahlendarstellung zur Basis **`base`**.
Zu jeder neu gewonnenen Ziffer brauchen Sie die korrekte Darstellung im neuen Zahlensystem, also das entsprechende Zeichen in **`digits`**. Auf das Zeichen Nummer **`i`** in diesem String können Sie mit **`digits.charAt(i)`** zugreifen. Diesen Ausdruck können Sie in einer Variable vom Typ **`char`** zwischenspeichern oder direkt weiterverwenden.
Jede neu gewonnene Ziffer hängen Sie nun **vorne** an den String an. Um Strings oder andere Variablen an Strings zu hängen, verbinden Sie die Variablen in der richtigen Reihenfolge mit dem Operator **`+`**.
Zuletzt muss der String, in dem nun (hoffentlich) das richtige Ergebnis steht, mittels **`return`** zurückgegeben werden.

d) Schreiben Sie eine Methode **`void out`**, die einen Parameter **`int number`** erwartet. Diese soll einen Text im folgenden Format auf dem Bildschirm ausgeben:

```
38 dezimal = 123 zur Basis 5
```

Die fett gedruckten Zahlen müssen Sie natürlich durch Variablen bzw. einen Methodenaufruf ersetzen. Vergessen Sie die entsprechenden Leerzeichen nicht!

e) Testen Sie Ihre Methode, z.B. mit den in dieser Aufgabe angegebenen Zahlen.

Aufgabe 8: Array umkehren

Fügen Sie der Klasse **`Ascending`** aus Aufgabe 4 (Sie müssen dazu die Aufgabe nicht vollständig gelöst haben, die Datei **uebung12-4.task** von der Webseite reicht, ergänzt um die Implementierung der Methode **`print()`**, aus) eine Methode **`void reverse()`** hinzu.

Diese soll das Array umkehren, also den ersten Wert mit dem letzten vertauschen, den zweiten mit dem vorletzten etc.

Testen Sie die Methode, indem Sie dem **`task`** einen entsprechenden Funktionsaufruf hinzufügen.

Aufgabe 9: Rekursion

Platzieren Sie in einem **`task`** eine senkrechte Wand, die über die erste Straße geht, bei einer beliebigen Avenue > 1. Schreiben Sie nun eine Roboterklasse, die eine Methode **`int distanceToWall()`** enthält. Diese Methode soll rekursiv(!!!) herausfinden und zurückgeben, wie viele Schritte es bis zur Wand sind. Der Roboter soll nach dem Aufruf der Methode wieder an der Ausgangsposition stehen! Sie können davon ausgehen, dass er westlich der Wand steht und nach Osten schaut.
Zum Test erzeugen Sie im **`task`** dann einen Roboter dieser Klasse auf der ersten Straße in einer beliebigen Avenue westlich der Wand mit 9999 Beepern und Blickrichtung Osten. Legen Sie mit Hilfe der Methode an der Ausgangsposition des Roboters so viele Beeper, wie es Schritte bis zur Wand sind. Sorgen Sie dann dafür, dass der Roboter vom Beeperstapel verschwindet (**`turnLeft(); move();`**).

Viel Spaß bei den Aufgaben und viel Erfolg in der Klausur!