Grundlagen der Informatik 1
WS 2008/09

Prof. Mühlhäuser, Dr. Rößling, Melanie Hartmann, Daniel Schreiber
http://proffs.tk.informatik.tu-darmstadt.de/gdi1

Übung 8 Version: 1.0 8.12.2008

---

1 Mini Quiz

1. ☐ Scheme ist näher an der Funktionsweise eines Rechners orientiert als Java.
2. ☐ '−' kann als unäre Operation verwendet werden.
3. ☐ $a >> b$ verschiebt die Bitrepräsnetation von a um b Stellen nach rechts.
4. ☐ $<$ hat eine niedrigere Piorität als &&.

2 Fragen

1. Was ist der Hauptunterschied zwischen Java und Scheme?
2. Erklären Sie, was Operator-Priorität ist und bringen Sie folgende Operatoren in die richtige Reihenfolge: $*$, $=$, &&, $<=$, $++$, $-$.
3. Warum spielte in Scheme die Operator-Priorität keine Rolle?
4. Erklären Sie, was Operator-Assoziativität ist. Welche Assoziativität haben die oben in 2.2 genannten Operatoren?

3 Schleifen

Gegeben sei folgende `while` Schleife in Java:

```
1 int x = 2;
2 int number = 10;
3 int count = 1;
4 while (Math.pow(x, count) <= number){
5   count++;
6 }
```

Hinweis: `Math.pow(a,b)` berechnet a^b.

1. Was macht die Funktion? Kommt sie Ihnen bekannt vor?
2. Ersetzen Sie die `while` Schleife durch eine `for` Schleife.
3. Ersetzen Sie die `while` Schleife durch eine `do while` Schleife.

4 Scheme in Java

1. **RGB Werte** In Übung 3 Aufgabe 3 haben Sie eine Scheme Prozedur geschrieben, die aus einem gegebenen Farbwert in RGB den zugehörigen Farbwert (Hue-Wert) ermittelt. Schreiben Sie nun die entsprechende Methode **float** getHue(**int** red, **int** green, **int** blue) in Java. Zur Erinnerung, die Berechnung erfolgt wie folgt:

$$r = red/255, g = green/255, b = blue/255$$

$$M = max(r, g, b), m = min(r, g, b)$$

$$H = \begin{cases} 0 & \text{wenn } M = m \\ 60(g-b)/(M-m) & \text{wenn } r = M \\ 120 + 60(b-r)/(M-m) & \text{wenn } g = M \\ 240 + 60(r-g)/(M-m) & \text{wenn } b = M \end{cases}$$

2. **Skalarprodukt** In Übung 2 Aufgabe 6.1 haben Sie das Skalarprodukt zweier Vektoren berechnet. Schreiben Sie diese Methode in Java (**int** computeScalar(**int**[] a, **int**[] b)). Vektoren sollen dabei als ein Array von Zahlen repräsentiert werden. Zur Erinnerung: Das Skalarprodukt zweier Vektoren ist die Summe der Produkte der Komponenten der Vektoren:
`scalarProduct(new int[]{3, 1, 0}, new int[]{1, 0, 2})` = `3*1 + 1*0 + 0*2`.

3. **Bubblesort** In Übung 5 Aufgabe 6 haben Sie Bubblesort implementiert. Zur Erinnerung: Beim Durchlaufen des Arrays (in Scheme: der Liste) vertauscht man zwei benachbarte Elemente genau dann, wenn sie nicht in der richtigen Reihenfolge stehen. Dadurch "bubblet" das größte Element automatisch an die höchste Position des Arrays. Daher verringert sich die Länge der zu sortierenden Arrayteilfolge mit jeder Iteration um eins, womit sichergestellt ist, dass die Prozedur terminiert.

   Wenn Sie am Rechner arbeiten, schreiben Sie zur Kontrolle der Sortierung eine weitere Methode **void** printArray(**int**[] array), die ein Array auf der Konsole ausgibt. Verwenden Sie dazu die Methode System.out.println(String output) und den Operator "+", um zwei Strings zu verketten, z.B. System.out.println("Der Wert der Variable x ist "+x); (x muss dabei kein String sein, es wird dann automatisch die Stringrepräsentation von x gewählt).

   Implementieren Sie nun Bubblesort in Java (**void** bubblesort(**int**[] array)). Verwenden Sie `printArray` wenn Sie am Rechner arbeiten, um sich die Schritte der Sortierung anzeigen zu lassen.

Hausübung

Die Vorlagen für die Bearbeitung werden im GdI1-Portal bereitgestellt. Kommentieren Sie Ihren selbst erstellten Code. Die Hausübung muss bis zum Abgabedatum im GdI1-Portal abgegeben werden. Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Mit der Abgabe Ihrer Hausübung bestätigen Sie, dass Sie bzw. Ihre Gruppe alleiniger Autor des gesamten Materials sind. Falls Ihnen die Verwendung von Fremdmaterial

gestattet war, so müssen Sie dessen Quellen deutlich zitieren.

Abgabe: Bis spätestens Fr, 09.01.2009, 16:00

5 Hello Java (0P)

Bevor Sie mit der eigentlichen Hausübung beginnen, sollten Sie sich zuerst mit den Sprachelementen von Java und Ihrer Entwicklungsumgebung vertraut machen. Wir empfehlen Ihnen *Eclipse* als Entwicklungsumgebung. Dies kann mit `eclipse` auf den Poolrechnern aufgerufen werden. Auf der GdI 1 Veranstaltungsseite finden Sie unter *Java-Add-Ons* neben dem Downloadlink von *Java* und *Eclipse* auch einen Link zum 'Java-Tutorial von Sun'.
Wenn Sie Eclipse benutzen möchten, empfehlen wir Ihnen, das *„Create a Hello World application"* Tutorial durchzuarbeiten, das Sie in Eclipse über `Help> Welcome> Tutorials` finden können.
Programme werden in Eclipse über `Run> Run As> Java Application` ausgeführt; eventuelle Ausgaben sehen Sie in der *Console*. Sie können zum Testen den folgenden Programmcode nutzen:

```
1  class HelloJava {
2    public static void main(String[] args) {
3      System.out.println("Hello Java!");
4    }
5  }
```

6 Code Chaos (4P)

Die aus Übungsblatt 3 bekannte Firma "Schematics" hat beschlossen, einen Teil ihrer Softwareentwicklung fortan in Java zu machen. Da die Mitarbeiter aber über keinerlei Erfahrung in Java verfügen, kommt dabei folgender Code zustande (den Sie auch auf der Vorlesungsseite herunterladen können):

```
1  public class Nonsense {
2    static int[] a;
3
4  public static double getD(){
5    double result = 1.0; for (int i=0; i<a.length; i++)     result+=a[i];
6    return result/a.length;
7  }
8
9  public static int getE(){
10   if (a.length<1) return -1;
11   double b = getD();       double d = Math.absolut(b-a[0]);int c = a[0];
       for (int i=1; i<a.length; i++){
12   if (Math.absolut(b-a[i])< d){d = Math.absolut(b-a[i]);c = a[i]; }
13   }return c;
14 }
15
16 public static void main(String[] args){
17   a = new int[]{2,5,1,4,9,7};
18   getE();
19 }
20 }
```

Damit wir erst mal erahnen können, was hier passiert, sollen einige Codeverschönerungs und -bereinigungsoperationen vorgenommen werden. Wir empfehlen hierzu Eclipse zu verwenden, da-

mit Sie Eclipse genauer kennen lernen (die in Eclipse benötigten Operationen werden im Folgenden angegeben), aber Sie können das auch mit jedem anderen Tool machen.

1. Formatieren Sie den Code, so dass nur noch ein Befehl in einer Zeile steht. In Eclipse können Sie dazu `Source->Format` verwenden.

2. Der Code lässt sich so wie er ist nicht kompilieren, da eine Methode `Math.absolut` verwendet wird, die nicht existiert. Der Entwickler wollte hier wohl den absoluten Wert einer Zahl berechnen. Wie heißt die Methode richtig? Wenn Sie Eclipse verwenden, gehen Sie dazu mit dem Cursor hinter `Math.` und drücken Strg+Leertaste. Eclipse zeigt Ihnen dann an, welche Funktionen in der Klasse `Math` zur Verfügung stehen. Wählen Sie die passende aus.

3. Als nächstes benennen wir einige Variablen um, da `a`, `b`, `c` und `d` nicht besonders aussagekräftige Namen sind. `a` soll `array` heißen, `b` `average`, `c` `bestElement` und `d` `minDistance`. Dies kann mit Eclipse wie folgt gemacht werden: wählen Sie die entsprechende Variable an (Cursor auf ihren Namen setzen), wählen Sie `Refactor->Rename` und geben Sie den neuen Variablennamen ein.

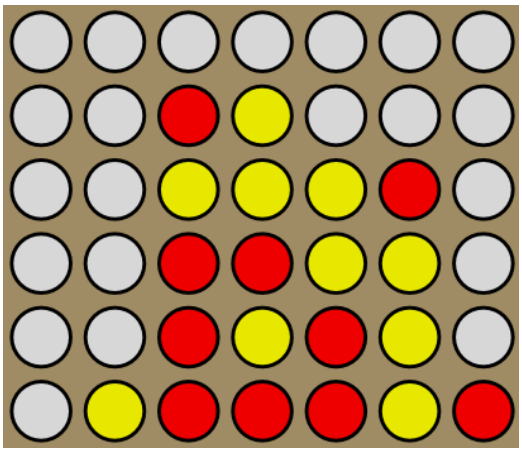
4. Geben Sie nun auch den Methoden (`getF`, `getE`) und der Klasse `Nonsense` aussagekräftigere Namen. Gehen Sie dabei mit Eclipse wie bei der Umbenennung der Variablen vor.

5. Fügen Sie in die Methode, die bis eben `getE` hieß, vor der Rückgabe eine Ausgabe ein, die angibt, was die Methode gerade berechnet hat. Geben Sie dazu einen Satz auf der Konsole aus, in dem `average`, `bestElement` und `minDistance` vorkommen.

6. Die Firmenpolitik untersagt eigentlich, dass globale Variablen (hier `array`) in der main Methode direkt gesetzt werden dürfen. Schreiben Sie daher eine Methode `getArray`, die `array` zurückgibt und eine Methode `setArray`, die ein Array aus ints entgegennimmt und in der Variablen `array` speichert. In Eclipse können Sie dazu die entsprechende Variable anwählen und über `Source->Generate Getters and Setters...` die zu erzeugenden Methoden auswählen.

7 4 gewinnt (9P)

In dieser Hausübung sollen Sie das Spiel "4 gewinnt" implementieren. Die Spielregeln sind wie folgt: Das aufrecht stehende Spielbrett besteht aus sieben Spalten und sechs Reihen (s. Abbildung) in das zwei Spieler abwechselnd ihre Spielsteine fallen lassen. Jeder Spieler besitzt 21 gleichfarbige Spielsteine. Gewinner ist der Spieler, der es als erster schafft, vier seiner Spielsteine waagerecht, senkrecht oder diagonal in eine Linie zu plazieren.



Im GDI-Portal können Sie ein Gerüst für die zu implementierende Klasse `ConnectFour` herunterladen. `ConnectFour` erweitert die aus der Vorlesung bekannte Klasse `DialogProgram`, damit stehen alle Funktionen aus `DialogProgram` (wie z.B. `println` oder `readInt`) in ihrer Klasse zur Verfügung (s. T11.16ff). Dazu müssen Sie die zugehörige Bibliothek (`acm.jar`) herunterladen und einbinden, wie im Portal unter *Java Add-Ons* beschrieben. Sie können statt `DialogProgram` auch `ConsoleProgram` verwenden, dann hat das Spiel eine Konsolenausgabe. Der Hauptteil der Programms befindet sich in der Methode `run`, die das Spielfeld darstellt und abwechselnd Züge der zwei Spielern einliest.
Das Spielbrett wird als ein Array aus `chars` repräsentiert, das die Breite `width` und Höhe `height` hat. Jeder Spielstein eines Benutzers wird durch einen Character dargestellt (entweder 'x' oder 'o'), der über `getPlayerChar(int playerNo)` abgefragt werden kann. Die Nummer des Spielers ist entweder 1 oder 2.
Hinweis: Vergessen Sie nicht, für jede Methode mit Hilfe von JavaDoc Vertrag (`@param`, `@return`) und Beschreibung anzugeben.

1. Implementieren Sie die Methoden promptPlayerMessage, rowFullMessage, winMessage, drawMessage und errorMessage so, dass sinnvolle Meldungen an den Spieler ausgegeben werden.

2. Implementieren Sie die Methode **void** init (), die das gesamte Spielfeld „leer“ (mit '_') initiiert.

3. Implementieren Sie die Methode **int** putInColumn(**int** colNo, **int** playerNo), die eine Zeilennummer und die Spielernummer entgegennimmt und einen Spielstein des entsprechenden Spielers in die entsprechende Zeile einfügt. `putInColumn` soll die y-Koordinate zurück geben, an die der Spielstein eingefügt wurde. Ist die Spalte bereits voll, soll die Methode -1 zurückgeben. Die `run` Methode ruft die Funktion nach einer Benutzereingabe auf. Gibt die Methode -1 zurück, wird ein Fehler ausgegeben und der aktuelle Spieler wird um eine neue Eingabe gebeten.

   Vorsicht: der Benutzer gibt eine Zeilennummer zwischen 1 und `width` ein. An die Methode wird aber die Zeilennumer - 1 übergeben, da die Zeilennnummerierung eines Arrays bei 0 und nicht bei 1 beginnt.

4. Die Methode **boolean** hasWon(**int** x, **int** y) gibt zurück, ob ein neu hinzugefügter Spielstein in der Zeile x auf der Höhe y eine Viererreihe bildet und der aktuelle Spieler somit gewonnen hat. Sie greift auf die Methoden **boolean** hasFourVertically (), **boolean** hasFourHorizontally() und **boolean** hasFourDiagonally() zurück, die als nächstes implementiert werden sollen.

   - Implementieren Sie die Methode **boolean** hasFourVertically(**int** x, **int** y). Diese Methode nehmen die Koordinaten des eingesetzten Spielsteins entgegen (x, y) und geben zurück, ob sich eine vertikale Viererkette gebildet hat.
   - Implementieren Sie die Methode **boolean** hasFourHorizontally(**int** x, **int** y). Diese Methode überprüft wie oben auf horizontale Viererketten.
   - Implementieren Sie die Methode **boolean** hasFourDiagonally(**int** x, **int** y), die überprüft, ob sich eine diagonale Viererkette durch den neuen Spielstein bei (x, y) gebildet hat.