Grundlagen der Informatik 1

Sommersemester 2011

Dr. Guido Rößling
https://moodle.informatik.tu-darmstadt.de/

Übung 8 Version: 1.0 06. 06. 2011

---

1 Mini Quiz

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

2 Fragen

1. Was ist der Hauptunterschied zwischen Java und Racket?
2. Erklären Sie, was Operator-Priorität ist und bringen Sie folgende Operatoren in die richtige Reihenfolge: $*$, $=$, &&, $<=$, $++$, $-$.
3. Warum spielte in Racket 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 Racket in Java

4.1 RGB Werte

In Übung 3.3 haben Sie eine Racket-Prozedur geschrieben, die aus einem gegebenen Farbwert in RGB den zugehörigen Farbwert (Hue-Wert) ermittelt. Schreiben Sie nun die entsprechende Methode **double** getHue(**int** red, **int** green, **int** blue) in Java. Zur Erinnerung, die Berechnung ist wie folgt:

$$\begin{aligned} r &= R/255 \\ g &= G/255 \\ b &= B/255 \\ M &= \max(r,g,b) \\ m &= \min(r,g,b) \end{aligned}$$

$$H = \begin{cases} 0^\circ & \text{für } M = m, \\ 60^\circ \times \frac{g-b}{M-m} & \text{für } r = M, \\ 120^\circ + 60^\circ \times \frac{b-r}{M-m} & \text{für } g = M, \\ 240^\circ + 60^\circ \times \frac{r-g}{M-m} & \text{für } b = M \end{cases}$$

Hinweis: Das Maximum zweier Zahlen *x, y* kann in Java berechnet werden durch Math.max(x, y). Allerdings funktioniert dies nur für *zwei* Zahlen.

4.2 Skalarprodukt

In Übung 2.6.1 haben Sie das Skalarprodukt zweier Vektoren berechnet. Schreiben Sie diese Methode in Java als **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, 4, 5}, **new int**[]{1, 0, 2}) = 3*1+4*0+5*2.

4.3 Selection Sort

Bei dem Sortierverfahren *Selection Sort* wird beim Durchlaufen des Arrays der Größe n im Durchlauf $i\ (= 0, \ldots, n-1)$ das kleinste Element ab Position $i+1$ bestimmt und mit dem Element an Position i vertauscht. 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 Methoden aus *ACM JTF* und den in Foliensatz T11.57 vorgestellten *StringBuffer*.
Implementieren Sie nun Selection Sort in Java (**void** selectionSort(**int**[] array)). Nutzen Sie printArray wenn Sie am Rechner arbeiten, um sich die Schritte der Sortierung anzeigen zu lassen.

Hausübung

Die Vorlagen für die Bearbeitung werden im Lernportal Informatik bereitgestellt. Kommentieren Sie Ihren selbst erstellten Code. Die Hausübung muss bis zum Abgabedatum im Lernportal Informatik 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.
Falls Sie die Hausübung in einer Lerngruppe bearbeitet haben, geben Sie dies bitte deutlich bei der Abgabe an. Alle anderen Mitglieder der Lerngruppe müssen als Abgabe einen Verweis auf die gemeinsame Bearbeitung einreichen, damit die Abgabe im Lernportal Informatik auch für sie bewertet werden kann. Beachten Sie dazu die Hinweise bei der Aufgabenabgabe im Lernportal Informatik!

Abgabedatum: Freitag, 17. 06. 2011, 16:00 Uhr

Denken Sie bitte daran, Ihren Code hinreichend gemäß den Vorgaben zu kommentieren (Racket:

Vertrag, Beschreibung und Beispiel sowie zwei Testfälle pro Funktion; Vertrag, Beschreibung und Beispiel für jede local definierte Funktion; Vertrag (ohne Namen) und kurze Beschreibung für jeden lambda-Ausdruck; Java: JavaDoc). Zerlegen Sie Ihren Code sinnvoll und versuchen Sie, wo es möglich ist, bestehende Funktionen wiederzuverwenden. Wählen Sie sinnvolle Namen für Hilfsfunktionen und Parameter.

5 Hello Java (0 Punkte)

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. Die aktuelle Fassung von Eclipse kann mit `eclipse` auf den Poolrechnern aufgerufen werden—zum Zeitpunkt des Schreibens war dies *Eclipse „Indigo" 3.7.1*. Zusätzlich gibt es noch die ältere Version 3.5.2, die gestartet wird über den Befehl `eclipse-3.5.2`. Im Lernportal Informatik finden Sie unter *Java Materialien - bitte unbedingt lesen!* neben dem Downloadlink von *Java* und *Eclipse* auch einen Link zu den „Java-Tutorials". Außerdem stellen wir einige Tutorial-Videos zur Nutzung von Eclipse bereit, auf die Sie auch zugreifen können.
Wenn Sie Eclipse benutzen möchten, empfehlen wir Ihnen, das *„Create a Hello World application"* Tutorial durchzuarbeiten, das Sie in Eclipse über `Help> Welcome> Tutorials` starten können. Hierzu gibt es auch ein Video von uns.
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 Chaos' Code (4 Punkte)

Die Firma „Schematic" hat beschlossen, einen Teil ihrer Softwareentwicklung fortan in Java zu machen. Der Mitarbeiter Herr Chaos hat dabei wohl seinen Namen ein wenig zu wörtlich genommen und überreicht Ihnen den folgenden ausgedruckten Code (den Sie nicht abtippen müssen, sondern im Lernportal herunterladen können)—von dem er stolz sagt, dass er „im Wesentlichen" schon funktioniert und sogar auf „einheitliche Zeilenlänge zur optimierten Anzeige und Papiereinsparung beim Ausdrucken" formatiert wurde:

```
public class ArrayTransformer{public void f1(){int x,
y;boolean z=true;for (x=ar.mySize;z&&x>-1;x--){for(y
=1,z=false;y<x;y++)/*Teste jetzt alle Elemente*/if (
ar[y-1]>ar[y])/*falsch angeordnet?*/{f2(ar,y-1,y);/*
dann vertausche!*/z=true;/*merke 1 Tausch*/}}}public
void f2(int[] a,int x,int y){int z;/*Check if a OK &
access OK??*/if (a!=null&&x>=0&&a.length>x/*x valid?
*/&&y>=0&&a.length>y)/*y valid?*/{/*Do it*/z=a[x];a[
x]=a[y];a[y]=z;}}public static void main(String[] b)
{int[] pk={15,7,3,12,4,9,2,6,1,3,1};ArrayTransformer
a=new ArrayTransformer();long takeTimeOfNow=System.
nanoTime();a.ar=pk;a.print();a.f1();System.out.println
("Program execution time in ns: "+(System.nanoTime()
-takeTimeOfNow));a.print();}public void print(){String
s="";for(int x=0;x<ar.length;x++){s=s+ar[x]+" ";}
System.out.println(s);}public int[]ar;}
```

Damit wir erahnen können, was hier passiert, sollten zunächst einige Schritte zur Verschönerung und Bereinigung des Codes vorgenommen werden. Wir empfehlen hierzu Eclipse zu verwenden, damit 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 (0.5 Punkte).

2. Der Code lässt sich so wie er ist nicht kompilieren, da ein Zugriff auf `mySize` verwendet wird, welches nicht existiert. Der Entwickler wollte hier wohl die Länge des Arrays herausfinden. Wie heißt das Element richtig? Wenn Sie Eclipse verwenden, gehen Sie dazu mit dem Cursor hinter `ar.` und drücken Strg+Leertaste. Eclipse zeigt Ihnen dann an, welche Attribute und Funktionen für Arrays zur Verfügung stehen. Wählen Sie das Passende aus (0.5 Punkte).

3. Als nächstes benennen wir einige Variablen um, da `ar`, `x` und `y` nicht besonders aussagekräftige Namen sind. Dies kann mit Eclipse wie folgt gemacht werden: selektieren Sie die entsprechende Variable (Cursor auf den Namen setzen), wählen Sie `Refactor->Rename`, geben Sie den neuen Variablennamen ein und bestätigen Sie den neuen Namen mit Return (0.5 Punkte).

4. Geben Sie nun auch den Methoden `f1` und `f2` sowie der Klasse `ArrayTransformer` aussagekräftigere Namen. Gehen Sie dabei mit Eclipse wie bei der Umbenennung der Variablen vor (0.5 Punkte).

5. Die Firmenpolitik untersagt eigentlich, dass Attribute (hier `ar`) in der `main`-Methode direkt gesetzt werden dürfen. Schreiben Sie daher eine Methode `getArray`, die das Feldattribut zurückgibt und eine Methode `setArray`, die ein `int[]` entgegennimmt und in dem Attribut speichert. In Eclipse können Sie dazu die entsprechende Variable anwählen und über `Source ->Generate Getters and Setters...` die zu erzeugenden Methoden auswählen (0.5 Punkte).

6. Ändern Sie die `print`-Methode so ab, dass sie statt einem String und dem „+“-Operator einen *StringBuffer* (siehe T11.57) verwendet, um das **int**-Array auszugeben.

   Passen Sie dann Ihr Programm so an, dass es ein *ACM JTF DialogProgram* ist und verwenden Sie *println*, um den im vorherigen Schritt erzeugten String auszugeben (0.5 Punkte).

7. Fügen Sie zuletzt auch noch angemessene Kommentare im JavaDoc-Format hinzu (1 Punkt).

7 Fußballweltmeisterschaft (6 Punkte)

In dieser Aufgabe setzen wir den Sportanteil der GdI 1 fort. Nachdem Sie bereits die Erstellung einer Tabelle bei Gruppenspielen—aber in Racket—modelliert haben, implementieren Sie nun in Java die Modellierung einer KO-Runde bei einem Fußballwettbewerb. Mit leichten Anpassungen könnte die Modellierung auch auf andere Sportarten passen.
Zu den Spielregeln: ein Spiel dauert 90 Minuten. Hat eine Mannschaft am Ende dieser *regulären Spielzeit* mehr Tore erzielt, hat sie gewonnen. Andernfalls gibt es 2×15 Minuten *Verlängerung* (im Englischen „extra time“). Hier werden nur die in der Verlängerung erzielten Tore betrachtet. Besteht auch danach noch ein Gleichstand, gibt es ein Elfmeterschießen[1]. Hier gibt es am Ende in jedem Fall einen Sieger.
Der Sieger einer Begegnung tritt in einer späteren Begegnung erneut an, bis es nur noch eine Mannschaft gibt: diese ist der Turniersieger. Der Verlierer einer „KO“-Partei scheidet direkt aus und spielt nicht mehr (es gibt hier also kein „Spiel um Platz 3“).
Von Ihnen zu implementieren sind dabei die folgenden Klassen: *SoccerTeam, SoccerMatch, SoccerTournament* sowie *SoccerSimulator*.

[1] Die Details der Regelung des Elfmeterschießens sind für diese Aufgabe unerheblich und werden daher vereinfacht.

Zu beachten: bitte verwalten Sie die aktuell noch spielenden Mannschaften in einem Array (siehe weiter unten). In einer Runde spielen immer die „benachbarten“ Mannschaften gegeneinander: die Mannschaft an Position 0 spielt als „Heimmannschaft“ gegen die Mannschaft an Position 1; die von Position 2 als „Heimmannschaft“ gegen Position 3, etc.
Hinweis: Es gibt zu dieser Aufgabe bewusst nur eine Beschreibung und *keine* Vorlagen, damit Sie das Schreiben von Klassen trainieren können! Eine *einfache* Tests stellen wir bereit; die *echten Tests sollten Sie aber selbst implementieren*. Unsere eigenen Tests sind—wie üblich—umfangreicher als die Ihnen bereitgestellten Tests.
Zu beachten: Sie müssen sich *generell selbst überlegen*, welche Attribute Sie (und ggf. wo) deklarieren wollen, um die Aufgabe zu lösen! Beachten Sie dabei die Faustregel „erst planen, dann implementieren“.
Zu beachten: Wir stellen einige (einfache) Tests für Sie bereit. Diese können Sie in Eclipse importieren. Um die Tests zu starten, gibt es mindestens drei Wege:

- Die Testklasse wird als aktuelle Klasse im Texteditor angezeigt. In diesem Fall können Sie im Hauptmenü den Eintrag Run > Run As... > JUnit Test auswählen, um den Testvorgang zu starten.
- Alternativ können Sie mit der im Menü angezeigten Tastenkombination den Test starten. Dazu müssen Sie entweder im Texteditor die Testklasse angezeigt bekommen oder im „Package Explorer“ die Testklasse markiert (angeklickt) haben.
- Alternativ können Sie im „Package Explorer“ mit der *rechten* Maustaste auf die Testklasse klicken und nun Run As > JUnit Test auswählen. Das funktioniert unabhängig davon, ob Sie die Klasse geöffnet haben oder nicht.

7.1 Klasse SoccerTeam (1 Punkt)

Diese Klasse modelliert eine Mannschaft, etwa die Nationalmannschaft von Deutschland. Sie soll die folgenden Methoden anbieten:

- **public** SoccerTeam(String) ist der Konstruktor der Klasse. Er erhält den Namen der Mannschaft als Argument.
- **public** String getName() liefert den Namen der Mannschaft.
- **public void** addMatchResult(**int**, **int**) fügt ein Spielergebnis hinzu. Die beiden **int**-Argumente stehen dabei für die in der regulären und ggf. der Nachspielzeit erzielten bzw. erhaltenen Tore. Die Tore des denkbaren Elfmeterschießens werden hier hingegen *nicht* erfasst!
- **public** String toString() repräsentiert die Leistungen der Mannschaft im Turnier als String. Die Ausgabe hat *exakt* die folgende Form, wenn Name der Name der Mannschaft, og und fg die erzielten bzw. erhaltenen Tore und x die Anzahl Spiele der Mannschaft ist:

```
Name og:fg in x matches
Chile 3:2 in 1 matches
Argentina 4:5 in 3 matches
```

Zu beachten: Sie müssen sich selbst „passende“ Attribute überlegen, um die Funktionalität korrekt umzusetzen!

7.2 Klasse SoccerMatch (1 Punkt)

Diese Klasse modelliert ein Spiel zweier Mannschaften. Sie soll die folgenden Methoden anbieten:

- **public** SoccerMatch(**int**, SoccerTeam, SoccerTeam) ist der Konstruktor der Klasse SoccerMatch. Er erhält die laufende Nummer des Spiels (beginnend bei 1, siehe unten) und zwei Mannschaften.
- **public void** addResultRegular(**int**, **int**) erhält das Ergebnis eines Spiels, das bereits nach der regulären Spielzeit beendet wurde. Hier und in den folgenden Methoden steht die erste Zahl immer für die Anzahl Tore der „Heimmannschaft", die zweite für die der „Gastmannschaft".
- **public void** addResultExtraTime(**int**, **int**, **int**, **int**) erhält das Ergebnis eines Spiels, das erst nach der Verlängerung entschieden war. Die Parameter stehen zunächst für die Tore der regulären Spielzeit (Argument 1 und 2) und dann für die der Nachspielzeit (Argument 3 und 4).
- **public void** addResultPenaltyShootout(**int**, **int**, **int**, **int**, **int**, **int**) erhält das Ergebnis eines Spiels, das erst nach Verlängerung und Elfmeterschießen entschieden war. Die Parameter stehen für die Tore der regulären Spielzeit (Argument 1 und 2), der Nachspielzeit (Argument 3 und 4) sowie des Elfmeterschießens (Argument 5 und 6).
- **public** SoccerTeam getWinner() bestimmt den Sieger der aktuellen Partie gemäß den obenstehenden Spielregeln.
- **public** String toString() gibt einen String im folgenden Format zurück. Wir liefern hier extra mehrere Beispiele, um alle Fälle abzudecken:

```
1 Match 1: USA – Ghana: 0:3 Winner: Ghana
2 Match 2: Netherlands – Slovakia: 5:5 (1:0 a.e.t.) Winner: Netherlands
3 Match 3: Paraquay – Japan: 3:3 (1:1 a.e.t.) [4:0 PSO] Winner: Paraquay
```

  Dabei stehen „a.e.t." und „PSO" für die englischen Begriffe „after extra time" (*nach der Nachspielzeit*) bzw. „Penalty Shoot-Out" (*Elfmeterschießen*). Jeder Eintrag gibt also die Spielnummer, die Mannschaften—durch ein Leerzeichen, ein Minuszeichen und erneut ein Leerzeichen getrennt—und das Ergebnis sowie den Sieger an.

7.3 Klasse SoccerTournament (3 Punkte)

Diese Klasse repräsentiert ein Fußballturnier, etwa die Frauen-WM 2011. Diese Klasse soll die folgenden Methoden haben:

- **public** SoccerTournament(String[]) legt ein neues Turnier an und merkt sich die Namen der teilnehmenden Mannschaften. Für jede Mannschaft—hier als String übergeben—ist ein Objekt der Klasse SoccerTeam anzulegen und intern zu speichern.
- **public** SoccerTeam[] playRound(SoccerTeam[], **int**) erhält das Feld der in der aktuellen Runde (noch) mitspielenden Mannschaften sowie die laufende Nummer des Spiels im Gesamtturnier. Das erste Spiel der ersten Turnier-KO-Runde hat dabei die Nummer 1; das erste Spiel der folgenden Runde hat eine Nummer, die *um 1 größer ist als die höchste bisherige Nummer*.

  Zu beachten: Beachten Sie, dass Felder in Java immer bei 0 beginnen, so dass die *Spielnummer* und die *Position im Feld* nicht identisch sind!
- **public** String playTournament() führt ein gesamtes Turnier durch. In jeder Runde halbiert sich die Anzahl Mannschaften, da die Verlierer ausscheiden. Sie können davon ausgehen, dass die Anzahl Mannschaften immer eine Zweierpotenz ist, so dass die Halbierung immer exakt aufgeht.

Ein Turnier wird durchgeführt, indem beginnend mit allen Mannschaften eine Runde gespielt wird. Die verbliebenen (halb so vielen) Mannschaften spielen dann die nächste Runde. Dieser Prozess setzt sich fort, bis nur noch eine Mannschaft—der Turniersieger—übrig ist.

Das Ergebnis der Methode ist ein String. Dieser enthält die Ausgabe aller Spielergebnisse, wie bei der Klasse *SoccerMatch* definiert, sowie am Ende des Turniers den Sieger wie folgt:

```
Champion: Germany
```

- **public** SoccerMatch[] getMatches() liefert ein Feld mit den bislang stattgefundenen Spielen. Das Feld sollte genau die korrekte Größe haben, um alle Spiele eines Turniers abzuspeichern. Während des Turniers—also bevor alle Runden beendet wurden—sollten die entsprechenden „hinteren“ Einträge **null** sein. Am Ende des Turniers muss jede Position passend belegt sein, wobei das erste Match (ID = 1) an Position 0 steht.

  Hinweis: Überlegen Sie sich auf Papier, wie viele Spiele ein Turnier mit 2, 4, 8, 16. ... Mannschaften hat und legen Sie das Feld dann „passend groß“ an!

- **public** String getTeamStatistics() liefert die Gesamtstatistik über alle Teams gemäß der Spielergebnisse. Das Ausgabeformat ist bereits bei der Klasse *SoccerMatch* beschrieben. Die Anzeige wird eingeleitet durch den Text Team statistics:. Dieser Text sowie die Angabe zu jeder Mannschaft endet mit einem Zeilenvorschub, codierbar als String oder **char** in der Form "\n" bzw. '\n'.

7.4 Klasse SoccerSimulator (1 Punkt)

Diese Klasse hat eine Methode, die Sie ohne Erzeugung eines konkreten Objekts verwenden können: **public static** SoccerMatch getResultFor(**int**, SoccerTeam, SoccerTeam). Der Aufruf der Methode erfolgt über SoccerSimulator.getResultFor(...). Diese Methode erhält die ID des Spiels sowie die beiden Mannschaften und liefert eine Instanz von SoccerMatch, die auch das Ergebnis enthält.
Gehen Sie zur Bestimmung des Ergebnisses wie folgt vor:

- Binden Sie am Anfang der Datei—noch vor **public class**...—via **import** java.util.Random; die Klasse Random ein, die Zufallszahlen erstellen kann. Für uns relevant sind nur die beiden Methoden **int** nextInt(**int**) sowie **boolean** nextBoolean(). Die erste Methode erzeugt eine **int** zwischen 0 und dem übergebenen Wert (ausschließlich)—also [0, n)—, die zweite erzeugt zufällig entweder **true** oder **false**. Um die Methoden nutzen zu können, müssen Sie erst ein Objekt vom Typ Random erzeugen.

- Bestimmen Sie das Ergebnis der regulären Spielzeit durch die genannten Methoden. Die Anzahl Tore sollte auf 6 pro Team (einschließlich) begrenzt werden.

- Nur bei Gleichstand sind entsprechend die in der Verlängerung erzielten Tore (maximal je 2) zu bestimmen.

- Herrscht auch hier noch Gleichstand, verfahren Sie analog für die im Elfmeterschießen gefallenen Tore (maximal je 5).

- Sollte es auch hier unentschieden stehen, nutzen Sie die Methode nextBoolean, um zu bestimmen, ob die erste Mannschaft gewonnen hat oder nicht. „Schenken“ Sie der so bestimmten Siegermannschaft einfach ein Elfmetertor.

- Nachdem der Sieger feststeht, ist die entsprechende Methode der Klasse *SoccerMatch* zur Meldung des Ergebnisses zu nutzen.