Übungsblatt 1 (Block 1)

Prof. Dr. Olaf Hellwich und Mitarbeiter

Java-Wiederholung und Normalformen der booleschen Algebra

Verfügbar ab:	**23.04.18**
Abgabe bis:	**30.04.-04.05.18**

Organisatorisches zum Modul

- Organisatorische Einzelheiten (Prüfungsanmeldung, Termine, Kontakt-E-Mail-Adresse, aktuelle Ankündigungen ...) sind auf **ISIS** zu finden.

Bewertung der Hausaufgaben

- Es soll pro Gruppe **ein** Archiv im `zip`-Format bei ISIS hochgeladen werden. Dafür gelten folgende Kriterien:
  - Andere Dateiformate als **`.zip`** werden nicht akzeptiert.
  - Die Abgabe besteht aus einer `zip`-Datei mit dem Namen **TxxGyy.zip** (xx ist die Tutoriumsnr. und yy die Gruppennr. – z.B. T02G04.zip)
  - Programmierteil: Die `zip`-Datei enthält die **Java-Dateien**, welche zur Lösung der Aufgaben erstellt wurden sowie die **vorgegebenen** Dateien.
  - Nichtprogrammierteil: Weiterhin werden Nichtprogrammieraufgaben (z.B. Handsimulationen) ausschließlich im **`pdf`**-Format abgegeben. Andere Formate werden nicht akzeptiert, außer es ist explizit anders angegeben.
- Die Online-Abgabe muss **vor** Beginn des eigenen offiziellen Tutoriums (in Moses) erfolgen. Zu späte Abgaben werden mit 0 Punkten bewertet.
- **Handschriftliche** Abgaben von Quellcode im Tutorium werden mit 0 Punkten bewertet.
- **Mehrdeutige Lösungen** werden mit 0 Punkten bewertet.
- **Nicht kompilierbare** Programme werden mit 0 Punkten bewertet.
- **Plagiate**, welche wir mit einer darauf spezialisierten Software finden, gelten als Betrugsversuch und werden mit „**nicht bestanden**“ für das Hausaufgabenkriterium und mit einer einsemestrigen Sperre für die Klausur geahndet.
  - Als Plagiat zählt beispielsweise die Abgabe einer Musterlösung aus einem früheren Semester (bzw. Teilen davon) oder die Abgabe von Lösungen (bzw. Teilen davon) von anderen Gruppen
- Es können Punkte für die äußere Form abgezogen werden.
- Java Programmcode muss sinnvoll (knapp aber ausreichend) kommentiert sein, sonst werden Punkte abgezogen.
- Java-Bibliotheken (wie z.B. Math.sqrt()) dürfen nur genutzt werden, wenn in der Aufgabenstellung darauf hingewiesen wird. Ansonsten können Punkte abgezogen werden.
- Die oben genannten Vorgaben gelten für **alle** (auch die nachfolgenden) Übungsblätter.

Bewertung der Tests

- Es soll pro Student pro Block online ein Test abgegeben werden.
- Die Frist für jeden der drei Tests wird rechtzeitig angekündigt.
- Angefangene Versuche werden bei Fristende automatisch abgegeben.
- Der letzte Versuch wird bewertet.
- Ein Test gilt als bestanden, wenn mindestens 50% erreicht wurden.

Aufgabe 1: Java-Wiederholung: Objektorientierung I — 5 Punkte

- Legen Sie eine von außen sichtbare Klasse `Floats` an, die ein **`float`**-Array `werte` und ein **`float`** `kleinstes` als von außen nicht sichtbare Attribute hat.
- Legen Sie in `Floats` einen von außen sichtbaren parametrisierten Konstruktor an, der das Attribut `werte` mit dem übergebenen `float`-Array initialisiert.
- Implementieren Sie in der Klasse `Floats` eine Methode **`float`** `kleinere(`**`float`** `x, `**`float`** `y)`, die von x und y den Wert zurückgibt, der den kleineren Absolutwert (Betrag) hat. Verwenden Sie die in Java verfügbare Methode **`float`** `Math.abs(`**`float`** `z)` zur Berechnung des Absolutwertes.
- Implementieren Sie nun in der Klasse `Floats` eine Methode **`void`** `setKleinstes()`, die dem Attribut `kleinstes` den Eintrag in dem Array `werte` mit dem kleinsten Absolutwert zuweist. Verwenden Sie hierzu die Methode `kleinere`.
- Implementieren Sie weiter eine Methode `ausgabeRekursiv`, die *rekursiv* die Werte aus dem Array `werte`, beginnend mit dem als Parameter übergebenen Index bis zum letzten Element auf dem Bildschirm ausgibt. Sie können davon ausgehen, dass nur gültige Indizes übergeben werden. Weiterhin soll das kleinste Element `kleinstes` zuerst gesetzt und dann ausgegeben werden.
- Legen Sie eine Test-Klasse `TestFloats` an:
  - In der `main`-Methode sollen Sie ein Array definieren, welches Sie mit 10 Objekten vom Typ `Floats` befüllen.
  - Verwenden Sie hierfür eine Schleife ($i = 0, 1, ..., 9$) und übergeben Sie bei der Instanziierung der `Float`-Objekte jeweils ein drei-elementiges **`float`**-Array gemäß der Bildungsregel (`1.1f + i` , `- i*9.5f` , `10 - i + 0.1f*i*i`), wobei in Java `(`**`float`**`)2.2` und `2.2f` äquivalent sind und eine double-Variable zu einer float-Variable casten.
  - Von dem `Floats`-Objekt im Array mit dem größten Index soll die Methode `ausgabeRekursiv` aufgerufen werden, sodass alle Elemente und das kleinste Element auf dem Bildschirm ausgegeben werden.

Musterlösung:

```
1  public class Floats {
2     // Attribute
3     private float[] werte;
4     private float kleinstes;
5
6     // Konstruktor
7     public Floats(float[] werte) {
8        this.werte = werte;
9     }
10
11    // gibt den Wert mit dem kleineren Betrag zurueck
12    public float kleinere(float x, float y) {
13       if (Math.abs(x) < Math.abs(y)) {
14          return x;
15       } else {
16          return y;
17       }
18    }
19
20    // setzt im Attribut kleinstes den kleinsten Wert aus dem Werte-Array
```

```
21     public void setKleinstes() {
22        kleinstes = werte[0]; // Anfangswert
23        for (int i = 0; i < werte.length ; i++){
24           kleinstes = kleinere(werte[i], kleinstes);
25        }
26     }
27
28     // gibt alle Werte vom uebergebenem Index bis zum Ende des Werte-
          Arrays und kleinstes Element aus
29     public void ausgabeRekursiv(int index) {
30        System.out.println("Ausgabe der Werte ab Index " + index + ":");
31        if (index == werte.length-1) { // Rekursionsabbruch
32           System.out.println(werte[index]);
33        } else { // rekursiver Aufruf
34           System.out.println(werte[index]);
35           ausgabeRekursiv(index + 1);
36        }
37
38        if(index==werte.length-1){ // da letztes Element durch
             aufsteigende Ausgabe immer ausgegeben wird
39           System.out.println("Ausgabe des kleinsten Elements:");
40           setKleinstes();
41           System.out.println(kleinstes);
42        }
43     }
44  }
```

```
1  public class TestFloats {
2     public static void main(String[] args) {
3        Floats[] values = new Floats[10];
4        for (int i = 0; i < values.length; i++) {
5           float[] threeFloats = {1.1f + i, -i*9.5f, 10 - i + 0.1f*i*i };
6           values[i] = new Floats(threeFloats);
7        }
8        values[values.length - 1].ausgabeRekursiv(0);
9     }
10 }
```

Aufgabe 2: Java-Wiederholung: Objektorientierung II — 3 Punkte

Gegeben sei das Interface in der Datei `Transportmittel.java`.

Implementieren Sie eine nicht-abstrakte, von außen sichtbare Klasse `Auto`, die das Interface `Transportmittel` implementiert.

Die Klasse `Auto` soll ein privates Attribut `float geschwindigkeit` besitzen, für welches Sie einen parametrisierten Konstruktor schreiben sollen, welcher das Attribut initialisiert. Die Geschwindigkeit ist in km/h angegeben. Die Geschwindigkeit vorwärts (rückwärts), die dem Konstruktor übergeben wird, darf dabei nicht größer als 100 km/h (50 km/h) sein. Ist die übergebene Geschwindigkeit größer als 100 km/h bzw. kleiner als -50 km/h, soll die Geschwindigkeit mit 0 initialisiert werden.

Schreiben Sie außerdem eine Getter-Methode `getGeschwindigkeit` für das Attribut.

Die Methode `beschleunigen` soll die momentane Geschwindigkeit um die übergebene Geschwindigkeitsdifferenz verändern. Das Auto darf nicht schneller als 100 km/h (vorwärts) bzw. schneller als 50 km/h (rückwärts) fahren (siehe Konstruktor). Falls die Summe der übergebenen Geschwindigkeitsdifferenz und der momemanten Geschwindigkeit nicht im gefordertem Intervall liegt, soll eine Warnung ausgegeben werden mit dem Text: *Das*

Auto darf vorwaerts nicht schneller als 100 km/h und rueckwaerts nicht schneller als 50 km/h fahren. Ist die Geschwindigkeit also beispielsweise 98 km/h und soll um 5 km/h erhöht werden, dann soll die Geschwindigkeit auf 98 km/h bleiben und die Warnung ausgegeben werden, da die Geschwindigkeit ansonsten den zulässigen Bereich überschreiten würde.

Testen Sie Ihre Implementierung, indem Sie eine Testklasse `TestAuto` schreiben, in welcher Sie

- ein `Auto`-Objekt `auto` mit der Geschwindigkeit 0 km/h erzeugen.
- in einer for-Schleife 15 Mal die Geschwindigkeit um 8 km/h erhöhen.
- nach jeder Geschwindigkeitserhöhung die Geschwindigkeit auf der Konsole ausgeben.

Musterlösung:

```java
public class Auto implements Transportmittel {
   // Attribut
   private float geschwindigkeit; // in km/h

   // Konstruktor
   public Auto(float g) {
      if ( (g >= -50) && (g <= 100) ) {
         geschwindigkeit = g;
      } else {
         geschwindigkeit = 0;
      }
   }

   // Getter
   public float getGeschwindigkeit() {
      return geschwindigkeit;
   }

   // Methode vom Interface, die ueberschrieben wird
   public void beschleunigen(float geschwindigkeit) {
      float g = this.geschwindigkeit + geschwindigkeit;
      if ( (g >= -50) && (g <= 100) ){ // Geschwindigkeit pruefen
         this.geschwindigkeit += geschwindigkeit; // Geschwindigkeit
            setzen
      } else {
         System.out.println("Das Auto darf vorwaerts nicht schneller als
             100 km/h und rueckwaerts nicht schneller als 50 km/h fahren
            .");
      }
   }
}
```

```java
public class TestAuto {
   public static void main(String[] args) {
      Auto auto = new Auto(0);
      for(int i=0; i<=14; i++){
         auto.beschleunigen(8);
         System.out.println("Geschwindigkeit: "+auto.getGeschwindigkeit
            ()+" km/h");
      }
   }
```

```
9 }
```

Aufgabe 3: Normalformen von booleschen Ausdrücken **2 Punkte**

Formen Sie den nachfolgenden – von drei Variablen $x, y, z \in \{0, 1\}$ abhängigen – booleschen Ausdruck $f(x, y, z)$ in eine ausgezeichnete disjunktive Normalform (aDNF) um. Geben Sie in jedem Schritt die verwendeten Axiome und Eigenschaften der booleschen Algebra an, welche sie im Anhang finden.

$$f(x, y, z) = \overline{\overline{x} + y} \cdot z + x \cdot z + \overline{(x \cdot y)} \cdot z$$

Geben Sie außerdem an, ob die von Ihnen erhaltene aDNF auch eine minimale DNF ist (mit Begründung).

Musterlösung:

Zur Lösung der Aufgabe teilen wir den gesamten Ausdruck in drei Terme auf.
(Hinweis: Man kann die Umformung auch mit dem gesamten Ausdruck machen.)

1. Term:

$$\begin{aligned} \overline{\overline{x} + y} \cdot z &\overset{E22}{=} \overline{\overline{x}} \cdot \overline{y} \cdot z \\ &\overset{E20}{=} x \cdot \overline{y} \cdot z \end{aligned}$$

2. Term:

$$\begin{aligned} x \cdot z &\overset{A10}{=} x \cdot (1) \cdot z \\ &\overset{A13}{=} x \cdot (y + \overline{y}) \cdot z \\ &\overset{A11}{=} x \cdot y \cdot z + x \cdot \overline{y} \cdot z \end{aligned}$$

3. Term:

$$\begin{aligned} (\overline{x \cdot y}) \cdot z &\overset{E21}{=} (\overline{x} + \overline{y}) \cdot z \\ &\overset{A11}{=} \overline{x} \cdot z + \overline{y} \cdot z \\ &\overset{A10}{=} \overline{x} \cdot (1) \cdot z + (1) \cdot \overline{y} \cdot z \\ &\overset{A13}{=} \overline{x} \cdot (y + \overline{y}) \cdot z + (x + \overline{x}) \cdot \overline{y} \cdot z \\ &\overset{A11}{=} \overline{x} \cdot y \cdot z + \overline{x} \cdot \overline{y} \cdot z + x \cdot \overline{y} \cdot z + \overline{x} \cdot \overline{y} \cdot z \end{aligned}$$

Damit ergibt sich die folgende DNF für den Ausdruck:

$$f(x, y, z) = x \cdot \overline{y} \cdot z + x \cdot y \cdot z + x \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z + \overline{x} \cdot \overline{y} \cdot z + x \cdot \overline{y} \cdot z + \overline{x} \cdot \overline{y} \cdot z$$

Wenden wir nun mehrmals A4 (Kommutativgesetz) an und eliminieren dann die doppelten Terme mit dem Idempotenzgesetz E18, erhalten wir die aDNF:

$$f(x, y, z) = x \cdot y \cdot z + x \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z + \overline{x} \cdot \overline{y} \cdot z$$

Diese aDNF ist keine minimale DNF. Es gibt viele Möglichkeiten dies zu zeigen. Eine ist, eine DNF mit weniger Termen herzuleiten. Das könnte man zum Beispiel wie folgt machen:

$$\begin{aligned}
x \cdot y \cdot z + x \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z + \overline{x} \cdot \overline{y} \cdot z &\overset{A11}{=} (x \cdot y + x \cdot \overline{y}) \cdot z + (\overline{x} \cdot y + \overline{x} \cdot \overline{y}) \cdot z \\
&\overset{A11}{=} x \cdot (y + \overline{y}) \cdot z + \overline{x} \cdot (y + \overline{y}) \cdot z \\
&\overset{A13}{=} x \cdot (1) \cdot z + \overline{x} \cdot (1) \cdot z \\
&\overset{A10}{=} x \cdot z + \overline{x} \cdot z \\
&\overset{A11}{=} (x + \overline{x}) \cdot z \\
&\overset{A13}{=} (1) \cdot z \\
&\overset{A10}{=} z
\end{aligned}$$

Ersetzt man den umgeformten Term, kommt man auf folgende DNF:

$$f(x, y, z) = z$$

Diese DNF besitzt weniger Terme und damit kann die aDNF nicht minimal sein.

Anhang
Axiome der booleschen Algebra

Für alle $a, b, c \in \{0, 1\}$ gilt

Nr	Bezeichnung	Axiom
A1	Assoziativgesetze	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$
A2		$a + (b + c) = (a + b) + c$
A3	Kommutativgesetze	$a \cdot b = b \cdot a$
A4		$a + b = b + a$
A5	Absorptionsgesetze	$a + (a \cdot b) = a$
A6		$a \cdot (a + b) = a$
A7	Existenz der Null und Eins	$a + 0 = a$
A8		$a \cdot 0 = 0$
A9		$a + 1 = 1$
A10		$a \cdot 1 = a$
A11	Distributivgesetze	$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
A12		$a + (b \cdot c) = (a + b) \cdot (a + c)$
A13	Existenz des Komplements	$a + \overline{a} = 1$
A14		$a \cdot \overline{a} = 0$

Eigenschaften der booleschen Algebra

Nr	Bezeichnung	Gesetz
E15	Negation / Komplement	$\overline{a} = 1 \Leftrightarrow a = 0$
E16	Konjunktion/ Durchschnitt	$a \cdot b = 1 \Leftrightarrow a = 1$ und $b = 1$
E17	Disjunktion / Vereinigung	$a + b = 1 \Leftrightarrow a = 1$ oder $b = 1$
E18	Idempotenz	$a + a = a$
E19		$a \cdot a = a$
E20	Involution	$\overline{\overline{a}} = a$
E21	De Morgan'sche Gesetze	$\overline{a \cdot b} = \overline{a} + \overline{b}$
E22		$\overline{a + b} = \overline{a} \cdot \overline{b}$