Technische Universität Berlin **Computer Vision & Remote Sensing**

Übungsblatt 1 (Block 1)

Prof. Dr. Olaf Hellwich und Mitarbeiter

Java-Wiederholung und Normalformen der Booleschen Algebra

Verfügbar ab:	**22.04.16**
Abgabe bis:	**02.-04.05.16**

Organisatorisches

- Zum Bestehen der Übung (= Klausurvoraussetzung) muss:
  - Mindestens die Hälfte der Summe der Punkte der Übungsblätter in jedem Block erreicht werden.
  - UND zusätzlich müssen mindestens zwei von drei der Onlinetests auf ISIS bestanden werden.
- Die Übungsblätter werden in Gruppen von 3 bis 4 Studenten bearbeitet.
- Programmiercode soll in **einem** `.zip` Archiv pro Gruppe bei ISIS hochgeladen werden, dafür gelten folgende Kriterien
  - Die Abgabe besteht aus einer zip-Datei mit dem Namen TxxGyy.zip (xx ist die Tutoriumsnr. und yy die Gruppennr. – z.B. T02G04.zip)
  - Die zip-Datei enthält die Java-Dateien, welche zur Lösung der Aufgaben erstellt wurden sowie die vorgegebenen Dateien.
  - Andere Datei-Formate als zip werden nicht akzeptiert.
- Die Online-Abgabe muss **vor** Beginn des jeweiligen Tutoriums erfolgen.
- Lösungen zu nicht-Programmieraufgaben können auch im Tutorium abgegeben werden (mit Gruppenkennung)
- Für den Programmier Code gilt:
  - Plagiate, welche wir mit einer darauf spezialisierten Software finden, gelten als Betrugsversuch und werden geahndet. **Bei einem Plagiat wird das gesamte Blatt mit 0 Punkten bewertet! Als Plagiat zählt jede Abgabe einer Musterlösung aus einem früheren Semester (bzw. Teilen davon) und die Abgabe von Lösungen (bzw. Teilen davon) von anderen Gruppen.**
  - Mit 0 Punkte bewertet werden:
    - Handschriftliche Abgaben von Java-Quellcode.
    - Mehrdeutige Lösungen (mehrere .zip-Dateien)
    - Nicht kompilierbare Java Programme
  - Punkte können abgezogen werden für:
    - Die äußere Form.
    - Fehlende Kommentare im Java-Code.
- Die oben genannten Vorgaben gelten für alle (auch die nachfolgenden) Übungsblätter.

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

- Legen Sie eine von außen sichtbare Klasse `Integers` an, die ein `int`-Array `werte` und ein `int groesstes` als von außen nicht sichtbare Attribute hat.
- Legen Sie in `Integers` einen von außen sichtbaren parametrisierten Konstruktor an, der das Attribut `werte` mit dem übergebenen `int`-Array initialisiert.
- Implementieren Sie in der Klasse `Integers` eine Methode `int groessere(int x, int y)`, die von x und y den Wert zurückgibt, der den größeren Absolutwert (Betrag) hat. Verwenden Sie die in Java verfügbare Methode `int Math.abs(int z)` zur Berechnung des Absolutwertes.
- Implementieren Sie nun in der Klasse `Integers` eine Methode `void setGroesstes()`, die dem Attribut `groesstes` den Eintrag in dem Array `werte` mit dem größten Absolutwert zuweist. Verwenden Sie hierzu die Methode `groessere`.
- 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.
- Legen Sie eine Test-Klasse `TestIntegers` an, in deren `main`-Methode Sie:
  - Einen Array definieren, welchen Sie mit 10 Objekten vom Typ `Integers` befüllen.
  - Verwenden Sie hierfür eine Schleife ($i = 0, 1, \cdots, 9$) und übergeben Sie bei der Instanziierung der `Integers`-Objekte jeweils ein drei-elementiges `int`-Array gemäß der Bildungsregel ($1 + i * 3,\ -1 - i * 2,\ 10 - i$).
  - Von dem `Integers`-Objekt im Array mit dem größten Index die Methode `ausgabeRekursiv` aufrufen.

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

Gegeben sei folgendes Interface:

```
1 public interface Transportmittel {
2     void beschleunigen(double geschwindigkeit);
3 }
```

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

Die Klasse `Auto` soll ein privates Attribut `double geschwindigkeit` besitzen, für welches Sie einen parametrisierten Konstruktor schreiben sollen, welcher das Attribut initialisiert. Die Geschwindigkeit ist in km/h angegeben. Der Betrag der Geschwindigkeit, die dem Konstruktor übergeben wird, darf dabei nicht größer als 50 km/h sein. Ist die übergebene Geschwindigkeit größer als 50 km/h bzw. kleiner als -50 km/h, dann 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 50 km/h fahren (also $-50 \leq$ `geschwindigkeit` ≤ 50). Falls die Summe der übergebenen Geschwindigkeitsdifferenz und der momemanten Geschwindigkeit größer als 50 km/h ist, soll eine Warnung ausgegeben werden mit dem Text: *Das Auto darf nicht schneller als 50 km/h fahren.* Ist die Geschwindigkeit also beispielsweise 48 km/h und soll um 5 km/h erhöht werden, dann soll die Geschwindigkeit auf 48 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 5 km/h erhöhen.

- nach jeder Geschwindigkeitserhöhung die Geschwindigkeit auf der Konsole ausgeben.

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.

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

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

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		$\overline{a + b} = \overline{a} \cdot \overline{b}$