

Tutoriumsvorbereitung 1

Prof. Dr. Olaf Hellwich und Mitarbeiter

24.04.2017 - 26.04.2017

Inhaltsverzeichnis

0 Organisatorisches	2
1 Wiederholung - Java	3
2 Wiederholung - Rekursion	3
3 Wiederholung - Boolesche Operatoren	4
4 Längere boolesche Ausdrücke	5
5 Normalformen	5
5.1 Übung: Erkennen von Normalformen	6
5.2 Herleitung der ausgezeichneten Normalformen aus einer Wertetabelle	6
5.3 Übung: aDNF und aKNF	6

cd desktop ↗ 4

cd java2 ↗ 4 ,

↳ java floats.java TestFloats.java

↳ java TestFloats
↳ main方法的執行

0 Organisatorisches

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 gehandelt.
 - 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.

1 Wiederholung - Java

Betrachten Sie das Interface Comparable, welches die Vergleichbarkeit zweier Objekte der implementierenden Klassen mit Hilfe einer Methode gewährleistet.

Listing 1: Comparable.java

```
1 public interface Comparable {
2     int compareTo(Comparable c);
3 }
```

- Schreiben Sie eine Klasse Rechteck.java, die das Interface Comparable implementiert.
- Ein Rechteck hat zwei nicht-öffentliche Attribute, Länge und Breite. Wählen Sie einen geeigneten Typ.
- Stellen Sie durch einen parametrisierten Konstruktor sicher, dass beide Attribute per Parameterübergabe gesetzt werden können.
- Berechnen Sie den Flächeninhalt des Rechtecks mit Hilfe einer Methode getFlaecheninhalt.
- Implementieren Sie die Methode compareTo des Interfaces. Der Vergleich soll auf Basis des Flächeninhalts geschehen.
- Dabei soll die Methode -1 zurückliefern, falls das Objekt, auf dem die Methode aufgerufen wird, einen kleineren Flächeninhalt aufweist, als die übergebene Instanz. Im umgekehrten Fall lautet die Rückgabe 1.
- Für den Fall, dass beide Objekte den gleichen Flächeninhalt besitzen, gibt die Methode 0 zurück.

Ergänzen Sie nun die Klasse TestComparable wie folgt:

implement Comparable 3, 也就是说

Listing 2: TestComparable.java

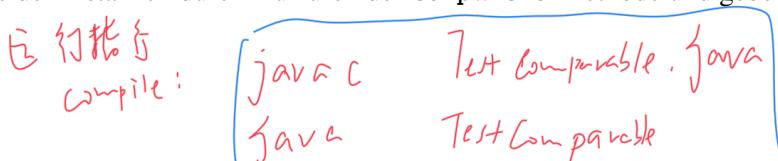
```
1 public class TestComparable {
2
3     public static void main(String[] args) {
4         Comparable c; // Comparable c;
5         Rechteck r; // Rechteck r;
6         Object o; // Object o;
7         // Erzeugen Sie für c, r und o je ein Rechteck, wo dies möglich ist.
8         // Vergleichen Sie die Flächeninhalte der Instanzen durch Aufrufen der
9         // compareTo-Methode und geben Sie das Ergebnis auf der Konsole aus.
10
11    // 和本阶段的 cast 在 Comparable
12
13 }
14 }
```

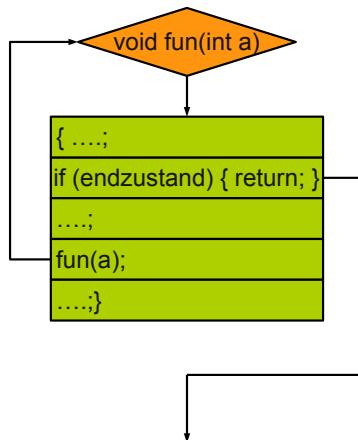
(To O 是 cast: ((Rechteck) o), 不建议调用 Comparable 的 getFlaecheninhalt 方法，因为这违反了接口设计的原则)

- Erzeugen Sie für c, r und o je ein Rechteck, wo dies möglich ist.
- Vergleichen Sie die Flächeninhalte der Instanzen durch Aufrufen der compareTo-Methode und geben Sie das Ergebnis auf der Konsole aus.

2 Wiederholung - Rekursion

- Rekursiver Aufruf einer Methode bedeutet: Sie ruft sich selbst wieder auf.
- Rekursive Methodenaufrufe benötigen:
 1. Einen Grundzustand (eine Abbruchbedingung), bei dem (der) sie aufhören.
 2. Einen rekursiven Funktionsaufruf
- das Ablaufdiagramm sieht folgendermaßen aus:





- Es gibt auch die indirekte Rekursion. Beispielsweise wenn in einer Methode `fun1(...)` eine Methode `fun2(...)` aufruft, in der wiederum die Methode `fun1(...)` aufgerufen wird.
- **Aufgabe:** Schreiben Sie eine rekursive Methode, die für ein `char[]`-Array testet, ob zwei aufeinanderfolgende Buchstaben gleich sind und genau dann `true` zurück liefert, wenn es zwei aufeinanderfolgende Buchstaben gibt, die gleich sind. Diese soll mit der folgenden Testklasse aufgerufen werden können.

```

1 public class TestReihen {
2     public static void main(String[] args) {
3         char[] word = {'H', 'a', 'l', 'l', 'o'};
4         System.out.println(Reihe.istdoppelt(word, 0));
5     }
6 }

```

→ 来自于 object 在 class 中存在，即为

- Hinweis: Jeder Funktionsaufruf benutzt eigene lokale Variablen! 从类名直接在 Test.java 中取
- Neben linearen Rekursionen gibt es weitere Kategorien von Rekursionen¹. Baumartige/kaskadenförmige Rekursionen haben mehrere rekursive Aufrufe nebeneinander stehen. Ein Beispiel ist die Fibonacci-Folge, deren Folgeglieder rekursiv über `fib(n-1)+fib(n-2)` berechnet werden.

3 Wiederholung - Boolesche Operatoren

x	y	AND x·y	OR x+y	XOR x⊕y	NOT \bar{x}
		$e_1 \quad \boxed{\&} \quad e_2 \quad a$	$e_1 \quad \boxed{\geq 1} \quad e_2 \quad a$	$e_1 \quad \boxed{= 1} \quad e_2 \quad a$	$e \quad \boxed{1} \quad a$
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Operatorrangfolge: \bar{x} vor $x \cdot y$ vor $x + y$ (Vergleichbar mit Punkt- vor Strichrechnung)

Andere Schreibweisen:

$$\begin{aligned}
 x \cdot y &\triangleq x \wedge y && (\text{Konjunktion}) \\
 x+y &\triangleq x \vee y && (\text{Disjunktion}) \\
 \bar{x} &\triangleq \neg x && (\text{Negation})
 \end{aligned}$$

Um die logischen Operatoren auf der Hardware abilden zu können, existieren entsprechende logische Gatter.

¹https://de.wikipedia.org/wiki/Rekursion#Formen_der_Rekursion

4 Längere boolesche Ausdrücke

Aus InfTech1 bekannte Fragestellung:

lesen Sie aus der gegebenen Wahrheitstabelle den entsprechenden booleschen Ausdruck ab.

x	y	a
0	0	0
0	1	1
1	0	0
1	1	1

$$a = (\bar{x} \cdot y) + (x \cdot y)$$

Anscheinend ist der Ausdruck nur von y (da y=a) abhängig. Es müsste also gelten:

$$a = (\bar{x} \cdot y) + (x \cdot y) = y$$

- Die Wertetabelle liefert offenbar nicht immer die minimale Form.
- Für die praktische Verwendung sollte ein Ausdruck aber minimal sein.
- Bald lernen wir aber eine Methode zum Minimieren kennen.

5 Normalformen

- Es gibt unterschiedliche äquivalente Darstellungsmöglichkeiten.
- Normalformen sind eine Variante (disjunktiv und konjunktiv).
- Des Weiteren gibt es die ausgezeichnete KNF und DNF (aKNF und aDNF).

DNF: Disjunktion von Konjunktionstermen

$$\text{Beispiel: } (x \cdot y) + (\bar{x} \cdot \bar{z}) + (x \cdot \bar{y} \cdot \bar{z})$$

KNF: Konjunktion von Disjunktionstermen

$$\text{Beispiel: } (x + y) \cdot (\bar{x} + \bar{z}) \cdot (x + \bar{y} + \bar{z})$$

Für die Fälle, wenn alle Min-/Max-Terme nur aus jeweils einer Variable bestehen, kann eine Normalform sowohl DNF als auch KNF sein. Auch eine alleinstehende Variable ist sowohl eine DNF, als auch eine KNF. (Max-Terme sind die verundeten Terme einer KNF. Mindestens ein Literal muss zu 1 ausgewertet werden, damit ein Term erfüllt ist. Min-Terme sind die veroderten Terme einer DNF. Alle Literale müssen zu 1 ausgewertet werden, damit ein Term erfüllt ist.)

Sind bei einer DNF (bzw. KNF) alle Variablen in jedem Konjunktionsterm (bzw. Disjunktionsterm) vorhanden, und unterscheiden sich alle Konjunktionsterme (bzw. Disjunktionsterme) so spricht man von einer aDNF (bzw. aKNF). In der Fachliteratur werden diese ausgezeichneten Normalformen auch als kanonische Normalformen (kurz KDNF und KKKNF) bezeichnet.

Neben der DNF, KNF (und ihren Spezialfällen aDNF und aKNF) existieren weitere Normalformen, die je nach Anwendungsvorhaben Vorteile bieten können.

5.1 Übung: Erkennen von Normalformen

In welcher Form liegen folgende boolesche Ausdrücke vor?

1. $(x + y) \cdot (x + z)$
2. $x + y + (x \cdot z)$
- ~~3. $\overline{(x + y)}$~~ *归到与或之和，2.3 表达式为与或表达式*
4. $(x + y + z) \cdot (\bar{x} + \bar{y} + \bar{z})$
5. $x + y$
- ~~6. $x + (y \cdot (z + x))$~~
7. $(z \cdot x \cdot y) + (\bar{y} \cdot x \cdot z)$

5.2 Herleitung der ausgezeichneten Normalformen aus einer Wertetabelle

aDNF:

- für jede 1-Zeile einen Konjunktionsterm mit allen Eingangsvariablen erstellen (\cdot)
- einzelne Terme anschließend disjunkt vereinigen ($+$)
- die einzelnen Konjunktionsterme nennt man Minterm (alle Variablen enthalten)

aKNF:

- für jede 0-Zeile einen Disjunktionsterm mit allen negierten Eingangsvariablen erstellen ($+$)
- einzelne Terme anschließend konjunktiv vereinigen (\cdot)
- die einzelnen Disjunktionsterme nennt man Maxterm (alle Variablen enthalten)

~~5.3 Übung: aDNF und aKNF~~

Leiten Sie die aDNF und die aKNF aus folgender Wertetabelle her:

x	y	z	a
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0