Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences
Fachbereich Informatik
Prof. Dr. Rudolf Berrendorf

Einführung in die Programmierung
WS 19/20
Ausgabe : 14.10.2019
Bearbeiten bis : 20.10.2019

Übung 3

Aufgabe 1 :
Formulieren Sie für die Java-Operationen `&` (Logisches Und) und `|` (Logisches Oder) das Kommutativgesetz, das Assoziativgesetz und das Distributivgesetz. Wie lauten die de Morganschen Regeln?

Aufgabe 2 :
Vereinfachen Sie folgende logische Ausdrücke:

1. `(x < 100) | !(x < 200)`
2. `x & !(y & !(z | y))`

Nutzen Sie ausschließlich Java-Syntax für logische Ausdrücke, wie dies in den vorgegeben Ausdrücken auch schon geschehen ist. Eine Umformung eines Ausdrucks in einen anderen äquivalenten Ausruck können Sie durch `==` kennzeichnen. Beispiel dazu: `a & b == b & a`.

Aufgabe 3 :
Für welche ganzzahligen Wertkombinationen der vorkommenden Variablen sind folgenden logische Ausdrücke wahr?

a) $(x < y)$ & $(y < z)$ & $(x > 100)$ & $(z < 200)$
b) $(x > -2)$ & $(x < 8)$ & $(x$ ist gerade $)$

Aufgabe 4 :
Geben Sie einen logischen Ausdruck an, der anhand einer Jahresangabe $j > 0$ feststellt, ob das Jahr ein Schaltjahr ist/war. Beachten Sie dabei den Unterschied zwischen Julianischem und Gregorianischem Kalender ab 1582.

Aufgabe 5 :
Rechnen Sie zuerst auf dem Papier folgende logische Ausdrücke per Hand aus und implementieren diese anschließend in Java. `wert1` und `wert2` seien logische Variablen mit den Wahrheitswerten `wert1=true` beziehungsweise `wert2=false`.

1. $(wert1$ & $wert2)$ | $wert1$
2. $(wert1$ & $wert2)$ & $!wert1$
3. $!((!wert1)$ & $wert2)$
4. $!(((5 < 7)$ & $wert1)$ || $wert2)$

Aufgabe 6 (1 Punkt) :
Dies ist eine Praktomataufgabe!

Schreiben Sie ein Java-Programm mit dem Namen `BoolExpression`, das zwei Variablen `wert1` und `wert2` des Typs `boolean` enthält. `wert1` soll zu Beginn den Wert `false` enthalten, `wert2` soll den Wert `true` enthalten.

Berechnen Sie in Java dann das Ergebnis des logischen Ausdrucks `!wert1 & (wert1 | wert2)` und geben Sie das Resultat dieses Ausdrucks auf dem Bildschirm aus. Die Ausgabe dazu wird also aus genau einer Zeile bestehen, in der entweder `true` oder `false` steht (und sonst nichts).

Ändern Sie anschließend im Programm die Variable `wert1` so, dass sie den Wert `true` enthält. Berechnen Sie den obigen Ausdruck erneut in ihrem Programm und geben wiederum den Ergebniswert des Ausrucks wie oben auf dem Bildschirm aus.

Die Ausgabe des Programms besteht also aus genau zwei Zeilen, die jeweils entweder `true` oder `false` enthalten.

Hinweise:

- Sie können das Ergebnis jeweils entweder in einer weiteren Variablen zwischenspeichern oder den o.g. logischen Ausdruck direkt in `System.out.println` verwenden.
- Sie sollen *ein Programm* entwickeln, das nacheinander die Variablen deklariert, den Ausdruck auswertet und das Resultat ausgibt, dann den Wert der Variablen `wert1` verändert und dann diesen Ausdruck auswertet und das Ergebnis dazu ausgibt.

Aufgabe 7 :
Geben Sie Java-Formulierungen für folgende logische Bedingungen an (i,j, und k stehen für int-Variablen beliebigen Inhalts)

1. i, j und k sind alle echt kleiner als 50
2. die Summe von i,j und k ist nicht durch 7 teilbar
3. j ist ungerade und ist entweder im Intervall $[3,21]$ oder im Intervall $[112,157]$
4. k ist weder durch 3 noch durch 7 teilbar, aber gerade

Aufgabe 8 :
Überlegen Sie sich jeweils zuerst, wie Sie die Umrechnung möglichst einfach vollziehen können.

1. Geben Sie die folgenden natürlichen Zahlen jeweils in Dual-, Dezimal-, Oktal- und Hexadezimaldarstellung an:
   (a) 110001101011_2
   (b) 3712_8
   (c) 9675_{10}
   (d) FAB_{16}
2. Ermitteln Sie für die folgenden ganzen Zahlen jeweils die (kürzestmögliche) Zweikomplementdarstellung:
   (a) -5_{10}
   (b) -300_{10}

   Was kennzeichnet die Bitfolge einer kürzestmöglichen Darstellung?
3. Ermitteln Sie für die folgenden ganzen Zahlen (in Zweikomplementdarstellung) jeweils die Dezimaldarstellung mit Vorzeichen (der hochgestellte * soll kennzeichnen, dass dies eine Zweierkomplementdarstellung ist):
   (a) 1000001100^*_2
   (b) 1111111111110010011^*_2

Aufgabe 9 (1 Punkt) :
Dies ist eine Praktomataufgabe!

Bei der menschlichen Eingabe von Zahlen kann es zu Falscheingaben kommen, zum Beispiel durch einen Tippfehler oder einen Zahlendreher. Einige Verfahren zur automatischen Erkennung solcher Fehler beruhen darauf, dass man zur eigentlichen Nutzinformation eine zusätzliche Prüfziffer anhängt, die sich aus den vorhergehenden Zahlen ergibt. Nachfolgend wird ein sehr einfaches Verfahren dazu beschrieben (Paritätsbit).

Schreiben Sie ein Java-Programm **`ParityBit`**, das Folgendes leistet. In einer int Variablen **`wert`** soll ein (beliebiger) Wert enthalten sein, der aus genau 4 Bit Nutzinformation in den Bits 1-4 besteht und einem zusätzlichen Bit als Prüfziffer im niederwertigsten Bit 0. Der Wert des Prüfbits muss die Quersumme der Bits der Nutzinformation modulo 2 sein, damit die Nutzinformationen als korrekt gewertet wird. Oder anders ausgedrückt: ist die Quersumme der Bits der Nutzinformation eine gerade Zahl, so muss das Prüfbit 0 sein, ansonsten 1.

Beispiele:

1. 11000_2 ist ein korrekter Wert, weil Summe 1+1+0+0=2 ergibt und dies demzufolge eine gerade Zahl ist, was dem Prüfbit 0 entspricht.
2. 11101_2 ist ein korrekter Wert, weil Summe 1+1+1+0=3 ergibt und dies demzufolge eine ungerade Zahl ist, was dem Prüfbit 1 entspricht.
3. 11111_2 ist ein inkorrekter Wert, weil Summe 1+1+1+1=4 ergibt und dies demzufolge eine gerade Zahl ist, aber das Prüfbit ist 1.

Schreiben Sie ihr Java-Programm basierend auf folgender Deklaration: **`int wert = 0x17;`** (wir haben derzeit noch keine Möglichkeiten zur Dateneingabe) und geben Sie folgendes aus, jeweils in einer Zeile und in der angegeben Reihenfolge:

- das Resultat der Überprüfung der Nutzinformation (Summe der 4 Bits) mit der Prüfziffer als Wahrheitswert (also entweder **`true`** oder **`false`**, je nachdem ob die Prüfziffer korrekt war oder nicht).
- die berechnete Summe (eine Zahl größer gleich 0)
- der Wert des Paritätsbits (0 oder 1).

Beispiel: Für den Wert $11000_2 = 24_{10}$ wäre die Ausgabe:

```
true
2
0
```

Ihr Programm soll natürlich nicht nur diesen Testwert bearbeiten können, sondern auch jeden anderen zulässigen Wert!

Hinweise:

- Nutzen Sie u.a. geeignete Bitoperationen.
- Wie kann man überprüfen, ob das 0. Bit eines int-Wertes 1 ist oder 0?
- Wenn Sie den Wert eines beliebigen Boolschen Ausdrucks ausgeben, so wird dieser Wert berechnet und entweder **`true`** oder **`false`** ausgegeben, je nachdem, ob der Wert des Ausdrucks wahr oder falsch war. Zum Beispiel produziert **`System.out.println(i > 5);`** die Ausgabe **`true`** oder **`false`**, abhängig vom Wert der Variablen **`i`**.