Aufgabe1

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

Kommutativgesetz: x|y = y|x, x&y = y&x
Assoziativgesetz: (x&y)&z = x&(y&z), (x|y)|z = x|(y|z)
Distributivgesetz: (x&y)|z = (x|z)&(y|z), (x|y)&z = (x&z)|(y&z)
de Morgansche Gesetze: !(x|y) = !x & !y, !(x&y) = !x | !y

Aufgabe2

Vereinfachen Sie folgende logische Ausdrücke:

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

1.(x < 100) | !(x < 200)=(x < 100) | (x > 200)

2. x & !(y & !(z | y))= x & (!y | (z | y))
=x & (!y | z | y)
= x & (!y | y | z)
= x & (1 | z)
= x &1
= x

Aufgabe 3 :

Für welche ganzzahligen Wertkombinationen der vorkommenden Variablen sind folgende logische Ausdrücke wahr?

1. (x >= y) & (x < 20) & (z >= 8) & (y > z)
alle werten in [8 20[so dass z<y<=x

2. (-x <= 5) & (x ist ungerade) & (x < 4)=(x>-5) & (x ist ungerade) & (x < 4)
=(x>-5) & (x =2k+1) & (x < 4)
=[-3,-1,1,3]

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.

Ein Jahr ist ein Schaltjahr, wenn die Jahreszahl durch 4 teilbar ist. Ist sie zusätzlich durch 100 teilbar, so ist das Jahr doch kein Schaltjahr. Ist die Jahreszahl aber zusätzlich durch 400 teilbar, so ist das Jahr doch ein Schaltjahr.

istSchaltjahr = (((jahr % 4) == 0) && (!((jahr % 100) == 0)))|| ((jahr % 400) == 0);

Übung 2

Aufgabe1

Schauen Sie sich an, welche Arten von Infomationen Sie zu Java auf der Webseite finden:
http://download.oracle.com/javase/8/docs/

ich habe alle API von java da gesehen.

Aufgabe 2 :

1.
SchauenSiesichdieGrammatikregelnzuHexadezimalkonstanteninJavaan(Nichtterminal symbol<HexNumeral>). Sie finden die Grammatikregeln dazu in Kapitel 3.10.1 des Javasprachstandards (siehe Link auf unserer Web-Seite zu Java).
Hexadezimalkonstanten sind eine andere Darstellungsmöglichkeit für Zahlen, die wir später noch intensi- ver betrachten werden. Geben Sie mit eigenen Worten an, was diese Definitionen beschreiben.
2. Geben Sie Ableitungen an von <HexNumeral>
(a) zu einem möglichst kurzen Wort nur aus Terminalsymbolen
(b) zu einem Wort nur aus Terminalsymbolen, in dessen Ableitung alle Regeln angewandt werden

1. Wir sehen Literals. Als Literal (lateinisch littera ‚Buchstabe') bezeichnet man in Programmiersprachen eine Zeichenfolge, die zur direkten Darstellung der Werte von Basistypen (z. B. Ganzzahlen, Gleitkommazahlen, Zeichenketten) definiert bzw. zulässig ist.

Man unterscheidet logische (wahr, nicht wahr), numerische und Zeichenliterale. Je nach Programmiersprache gibt es weitere und detailliertere Kategorisierungen für Literale. Damit Literale vom Compiler identifiziert werden können, müssen sie bestimmten syntaktischen Regeln genügen, z. B. (sprachenabhängig und in bestimmten Fällen) in

Anführungszeichen eingeschlossen sein.

Literale als Teil von Befehlen werden auch als literale Konstanten oder nicht benannte Konstanten bezeichnet, da sowohl Literale als auch Konstanten zur Laufzeit des Programms unveränderlich sind. Literale dürfen in Zuweisungsoperatoren nur als Sende-Ausdruck (i. d. R. rechtsseitig; 'Zeilen = 60'), als Argument einer Funktion oder als der Wert einer Konstante codiert werden.

In der funktionalen Programmierung können auch Funktionen als Literale geschrieben werden. Diese werden als anonyme Funktionen oder Lambda-Funktionen bezeichnet.

Ähnlich dem Literalbegriff kennen manche Programmiersprachen sog. „figurative Konstanten“. Dies sind (z. B. in Cobol) gem.[1] „Cobol-Worte [in unterschiedlichen Schreibweisen, z. B. in Pluralform] für die vom Compiler bestimmte Werte erzeugt werden“: Zero, Space, High-Value und Low-Value, Quote und 'ALL Literal'.

Als Aspekt des Programmierstils wird zum Teil empfohlen, im Befehlsteil des Programmcodes möglichst keine Literale, sondern ersatzweise Konstanten zu verwenden, die im Quelltext beliebig oft angesprochen werden können – anstatt immer das gleiche Literal zu verwenden. Diese Vorschrift soll zu höherer Transparenz und Wartungsfreundlichkeit führen.

2-a (a) zu einem möglichst kurzen Wort nur aus Terminalsymbolen
HexNumeral|----0 x HexDigits
|----0 xHexDigit [HexDigitsAndUnderscores] HexDigit
|----0 xHexDigit [HexDigitsAndUnderscores] HexDigit
|----0 xHexDigit HexDigitsAndUnderscores HexDigit
|----0 xHexDigit HexDigitsAndUnderscores 0
|----0 xHexDigit-0
|----0 x 1-0

2-(b) zu einem Wort nur aus Terminalsymbolen, in dessen Ableitung alle Regeln angewandt werden

Aufgabe 3 :