Dr. Annette Bieniusa
Mathias Weber, M. Sc.
Peter Zeller, M. Sc.

**TU Kaiserslautern**
**Fachbereich Informatik**
**AG Softwaretechnik**

# Übungsblatt 3: Software-Entwicklung 1 (WS 2017/18)

Ausgabe: 06.11.17
Abgabe: 13.11.17

Auf diesem Blatt werden Sie Ihre ersten Programmieraufgaben für SE1 lösen. Falls Sie bisher noch keine Erfahrung im Programmieren haben, ist es besonders wichtig, die Aufgaben selbst zu bearbeiten um die nötige Übung zu bekommen. Die folgenden Blätter bauen auf den Grundlagen auf, die auf diesem Blatt geübt werden. Beachten Sie auch, dass es auf Programme, die nicht kompilieren keine Punkte gibt.

## Aufgabe 1 Sprachen und Grammatiken (12 Punkte)

In dieser Aufgabe verwenden wir eine Mengennotation für Sprachen. Dabei steht $a^n$ für das Wort in dem $a$ $n$ mal wiederholt wird. Wörter lassen sich auch hintereinander schreiben, so steht $a^n b^m$ für das das Wort in dem zuerst $n$ mal $a$ und dann $m$ mal $b$ steht. Außerdem werden Klammern benutzt um ganze Gruppen zu wiederholen, zum Beispiel steht $(a^2 b)^3$ für das Wort *aabaabaab*. Mit diesen Abkürzungen für Wörter können wir leicht Mengen von Wörtern definieren. Zum Beispiel steht die Menge $\{a^n b^{2m} a^n \mid n \geq 0 \textit{ und } m \geq 1\}$ für die Sprache $\{bb, abba, aabbaa, \ldots, bbbb, abbbba, \ldots\}$. Für alle Wörter der Sprache gilt also, dass vorne genau so viele $a$ stehen wie hinten, und dass in der Mitte eine gerade Anzahl von $b$ steht. Außerdem muss es mindestens zwei $b$ in der Mitte geben, weil $m \geq 1$ gilt.

a) Geben Sie jeweils eine Grammatik für die folgenden Sprachen an:

$$
\begin{aligned}
L_0 &= \{a^n b^m \mid n \geq 1 \textit{ und } m \geq 2\} \\
L_1 &= \{a^n b^m \mid n \geq 1 \textit{ und } m \geq n + 2\} \\
L_2 &= \{(ab)^n \mid n \geq 0\} \\
L_3 &= \{a^n b^{2m} a^{3n} \mid n \geq 0 \textit{ und } m \geq 1\}
\end{aligned}
$$

b) Geben Sie die Sprachen für folgende Grammatiken in Mengen-Schreibweise an:

$$
\begin{aligned}
\Gamma_3 &= (N, T, \Pi, S) \quad \textit{wobei:} \quad N = \{S\},\ T = \{a\},\ \Pi = \{S \to aS \mid a\} \\
\Gamma_4 &= (N, T, \Pi, S) \quad \textit{wobei:} \quad N = \{S\},\ T = \{a, b\},\ \Pi = \{S \to aSa \mid b\} \\
\Gamma_5 &= (N, T, \Pi, S) \quad \textit{wobei:} \quad N = \{S, X\},\ T = \{a, b, c\},\ \Pi = \left\{ \begin{array}{lcl} S & \to & aXa \\ X & \to & bSb \mid c \end{array} \right\}
\end{aligned}
$$

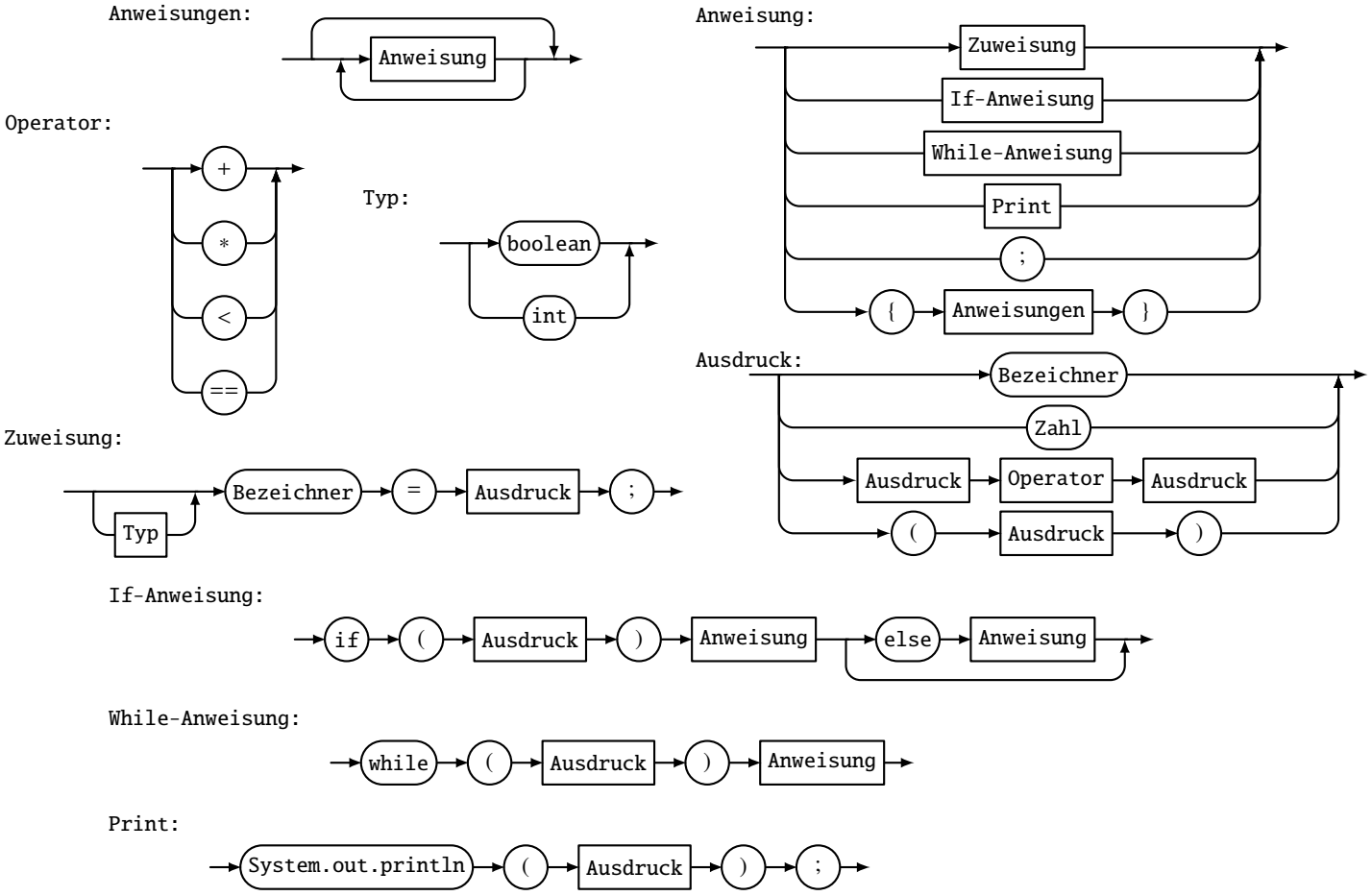
c) Geben Sie zur folgenden Beschreibung eine Grammatik und eine Definition in Mengennotation an:

Jedes Wort der Sprache ist eine mindestens einelementige Folge von $a$ gefolgt von einer möglicherweise leeren Folge von $x$, die zur linken und rechten durch ein einzelnes $l$ bzw. $r$ begrenzt ist.

**Beispiel**: Die Worte $alr, aaalxr, alxxxxr$ sind in der Sprache und die Worte $lar, arl, xrl, lxxr, allrr$ sind **nicht** in der Sprache enthalten.

## Aufgabe 2 Java Grammatik (10 Punkte)

Das folgende Syntaxdiagramm beschreibt die Syntax von Anweisungen, für eine Teilsprache von Java.




a) Entscheiden Sie für die folgenden 10 Eingaben, ob sie zu der Sprache gehören, die durch das gegebene Syntax-Diagramm mit Startsymbol "Anweisungen" definiert wird. Dabei steht der Terminalknoten "Zahl" für alle Zahlen bestehend aus Ziffern 0-9 und der Terminalknoten "Bezeichner" für Zeichenreihen aus den Buchstaben a-z, die nicht andersweitig als Terminalknoten verwendet werden. Leerzeichen und Zeilenumbrüche werden ignoriert.

Erklären Sie bei Eingaben, die nicht zur Sprache gehören, was der Fehler ist und wie dieser korrigiert werden könnte.

1.
```
System.out.println(2 == 3);
```

2.
```
x = 42;
```

3.
```
int x = 3;
x == 3;
```

4.
```
int i = 40;
int f = 2;
int if = i + f;
```

5.
```
int x = 1;
if (x = 1)
  x = 2;
else {
  x = 3;
}
```

6.
```
{int x=1
;}
{boolean x = 1 == 2
;}
```

7.
```
if (x < 0);
    b = 1;
else
    b = 2;
```

8.
```
{{{;}{{{};}}{};};}
```

9.
```
int x = 3;
if (x < 3)
  if (x < 10)
    x = 1;
else
  x = 2;
System.out.println(x);
```

10.
```
int x = 1;
while (x < 100)
    if (x < 90)
      x = x * 2;
    else
      x = x + 1;
System.out.println(x);
```

b) Testen Sie die Anweisungen aus Teil a) in einem Java-Programm. Verwenden Sie hierzu das Programm "Hello.java" von Aufgabenblatt 1 und ersetzen Sie die Anweisung `System.out.println("Hello World!")` durch die Anweisungen aus Teil a). Vergleichen Sie die Fehlermeldungen des Java-Übersetzers mit Ihren Ergebnissen aus Teil a). Für diese Teilaufgabe muss keine Lösung hochgeladen werden.

c) Wandeln Sie das Syntaxdiagramm in eine kontextfreie Grammatik um, welche die gleiche Sprache beschreibt.

## Aufgabe 3 Aufsteigend sortierte Zahlen (3 Punkte)

Schreiben Sie ein Java-Programm "IsStronglyMonotonicIncreasing3", welches drei double-Werte $x, y$, und $z$ nimmt und "true" ausgibt, wenn die drei Zahlen von links nach rechts immer größer werden (also $x < y < z$ gilt).

Beispiel:

```
> java IsStronglyMonotonicIncreasing3 1 2 3
true
> java IsStronglyMonotonicIncreasing3 4.1 4.2 4.2
false
```

*Zur Erinnerung: Die Funktion* `Double.parseDouble(String s)` *wandelt einen* `String` *in einen* **`double`** *Wert um.*

## Aufgabe 4 Wochentage (5 Punkte)

Schreiben Sie ein Programm "Wochentag", welches den Wochentag für ein bestimmtes Datum ausgibt. Dabei soll das Datum über 3 Programmparameter angegeben werden:

- Parameter 1 ist der Tag des Monats (1-31)
- Parameter 2 ist der Monat (1 für Januar bis 12 für Dezember)
- Parameter 3 ist das Jahr (positive Zahl)

Der entsprechende Wochentag soll ausgeben werden (deutsche Sprache, erster Buchstabe groß geschrieben), zum Beispiel:

```
> java Wochentag 13 11 2017
Montag
> java Wochentag 15 11 2017
Dienstag
```

Verwenden Sie zur Berechnung keine Datums-Funktionen der Standardbibliothek, sondern statt dessen die folgende Formel (Zellers Kongruenz):

$$h = \left( q + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \left\lfloor \frac{K}{4} \right\rfloor + \left\lfloor \frac{J}{4} \right\rfloor - 2J \right) \bmod 7$$

Hierbei ist:

- $h$ ist der Wochentag (0 = Samstag, 1 = Sonntag, 2 = Montag, ..., 6 = Freitag)
- $q$ ist der Tag des Monats (1-31)
- $m$ ist der Monat (März bis Dezember wie üblich von 3 bis 12; Januar und Februar werden als 13. und 14. Monat des vorherigen Jahres betrachtet)
- $K$ ist das Jahr im Jahrhundert ($year$ mod 100).
- $J$ ist das Jahrhundert ($\lfloor year/100 \rfloor$).

- Der Ausdruck "$x$ mod $y$" ist der Rest der Division von $x$ geteilt durch $y$, entspricht also für positive Zahlen dem Java-Ausdruck `x % y`.

  *Hinweis:* Um auch negative `x`-Werte korrekt zu behandeln kann statt dessen der Ausdruck `Math.floorMod(x, y)` verwendet werden.

- Der Ausdruck $\left\lfloor \frac{x}{y} \right\rfloor$ ist die Division mit abrunden auf die nächst kleinere ganze Zahl, entspricht also für positive Zahlen dem Java-Ausdruck `x / y`.

## Aufgabe 5 Logarithmus berechnen (4 Punkte)

Schreiben Sie ein Java-Programm "Logarithmus", welches zwei **`int`** Zahlen $x$ und $b$ als Programm-Parameter nimmt und die größte **`int`** Zahl $n$ berechnet, so dass $b^n \leq x$ gilt. Geben Sie $n$ auf der Konsole aus. Sie können davon ausgehen, dass $b > 1$ und $x \geq 1$ gilt. Verwenden Sie für diese Aufgabe keine mathematischen Funktionen aus der Java Standard-Bibliothek.

## Aufgabe 6 Programm-Parameter und Terminierung (6 Punkte)

*Für diese Aufgabe ist nur Teil e) abzugeben. Die anderen Teilaufgaben sollten Sie in der Abnahme erklären können.*

Gegeben ist das folgende Java-Programm:

```
public class Range {
    public static void main(String[] args) {
        int start = Integer.parseInt(args[0]);
        int end   = Integer.parseInt(args[1]);
        int step  = Integer.parseInt(args[2]);

        int i = start;
        while (i != end) {
            System.out.println(i);
            i = i + step;
        }
    }
}
```

a) Speichern Sie das obige Programm in einer Datei namens "Range.java" und übersetzen Sie es mit dem Java-Übersetzer.

b) Führen Sie das Programm mit dem Befehl `java Range 0 100 5` aus. Was ist die Ausgabe?

c) Was passiert, wenn Sie andere Argumente an das Programm übergeben? Zum Beispiel nur eine Zahl, vier Zahlen, oder Buchstaben? Wenn Fehler auftreten, wo steht die Zeilen-Nummer, in der der Fehler auftrat?

d) Für welche Eingaben terminiert das Programm, wenn man annimmt, dass es keine Überläufe gibt? Begründen Sie jeweils, warum das Programm terminiert. Wie sieht es mit Überläufen aus?

*Hinweis*: Was ist die Ausgabe von `java Range 0 25 4`?

Wenn Sie ein Programm von der Shell aus starten, dann können Sie es mit der Tastenkombination `Strg+C` abbrechen.

e) Verbessern Sie das Programm so, dass es immer terminiert und es für alle bereits terminierenden Eingaben im Verhalten unverändert bleibt.