Karlsruher Institut für Technologie
Lehrstuhl für Programmierparadigmen
Compilerpraktikum WS 2012/13
Dozent: Prof. Dr.-Ing. G. Snelting
Betreuer: Andreas Zwinkau zwinkau@kit.edu

Übungsblatt 4 | Ausgabe: 07.11.2012 | Besprechung: 14.11.2012

Aufgabe 1: *Abstrakter Syntaxbaum*

1.1 *Modellierung als abstrakte Algebra*

Überlegen Sie sich aus welchen Elementen der abstrakte Syntaxbaum bestehen sollte und erstellen Sie ein kleines Dokument in dem jedes Element in 2-3 Sätzen beschrieben wird. Formulieren Sie eine entsprechende abstrakte Algebra, wie sie in der Vorlesung Compiler 1 vorgestellt wurde (siehe `http://pp.info.uni-karlsruhe.de/lehre/SS2012/compiler/intern/03-syntaktischeanalyse.pdf`).

1.2 *Planung*

Zeigen Sie an folgendem Beispielprogramm wie Ihr geplanter AST aussehen sollte. Was sind die Unterschiede zum entsprechenden Parsebaum?

```
1  class A {
2
3  public int x;
4  public int y;
5
6  public int calc(int x, int y) {
7     int sum;
8
9     if (x > y) {
10       sum = x + y;
11    } else {
12       sum = x * y;
13    }
14
15    return sum;
16 }
17 }
```

1.3 *Modellierung als attributierte Grammatik*

Stellen sie eine attributierte Grammatik für den Aufbau des abstrakten Syntaxbaums auf (siehe `http://pp.info.uni-karlsruhe.de/lehre/SS2012/compiler/intern/04-attributiertegrammatiken.pdf`) und geben Sie diese ab.

1.4 *Implementierung*

Erweitern Sie ihren Parser anhand der attributierten Grammatik, so dass ein abstrakter Syntaxbaum erzeugt wird.

Aufgabe 2: *Pretty Printer*

Eine gute Art und Weise seinen Parser zu testen. Ist die Implementierung eines „Pretty-Printers“. Also ein Programm dass einen geparsten AST möglichst schön wieder ausgibt. Ein Pretty-Printer arbeitet korrekt, falls es kein semantischer Unterschied zwischen dem ausgegebenen und eingelesenen Programm besteht.

Um die Übersetzer im Praktikum vergleichbar zu machen beachten Sie bitte folgende Regeln:

- Eingerückt wird mit einem Tabulator Zeichen
- Ausdrücke werden stets vollständig geklammert ausgegeben. Ausnahmen hiervon sind die Literale (**`INTEGER_LITERAL`**, **`true`**, **`false`**, **`null`**, **`this`**, **`IDENTIFIER`**). Klammern aus dem Sourcecode werden nicht ausgegeben (sofern sie im AST vorhanden sind). Der Lesbarkeit zuliebe [1] werden die äußeren Ausdrücke in Expression-Statements, in Array-Zugriffen und Argumente beim Aufruf von Funktionen nicht geklammert.
- Binäre Operatoren werden mit Leerzeichen umgeben. Unäre Operatoren ohne Leerzeichen ausgegeben.
- Innerhalb einer Klasse werden zunächst alle Methoden, dann alle Felder ausgegeben.
- Methoden und Klassen werden alphabetisch sortiert ausgegeben.
- Im Zweifel sollte man die Art der Formatierung am Beispiel unten ablesen können, bei weiteren Fragen schreiben Sie uns E-Mails.
- Leere Block-Statements werden als `{ }` ausgegeben (keine Zeilenumbrüche)
- Falls der else-Teil eines if-Statements ein Empty-Statement enthält so wird der komplette else-Teil nicht ausgegeben.

Beispiel Eingabe:

```
class HelloWorld
{
    public int c;
    public boolean[] array;
    public static /* blabla */ void main(String[] args)
    { System.out.println( (43110 + 0) );
    boolean b = true && (!false);
    if (23+19 == (42+0)*1)
        b = (0 < 1);
        else if (!array[2+2]) {
            int x = 0;;
            x = x+1;
        } else {
            new HelloWorld().bar(42+0*1, -1);
        }
    }
    public int bar(int a, int b) { return c = (a+b); }
}
```

Ausgabe:

```
class HelloWorld {
    public int bar(int a, int b) {
        return c = (a + b);
    }
    public static void main(String[] args) {
        (System.out).println(43110 + 0);
        boolean b = true && (!false);
        if ((23 + 19) == ((42 + 0) * 1))
```

[1] und weil hier direkt auf den Parsebaum geschlossen werden kann

```
            b = (0 < 1);
        else if (!(array[2 + 2])) {
            int x = 0;
            x = (x + 1);
        } else {
            (new HelloWorld()).bar(42 + (0 * 1), -1);
        }
    }
    public boolean[] array;
    public int c;
}
```

Ändern Sie ihren Compiler so dass bei Aufruf mit dem Parameter `--print-ast` der AST in obigem Format ausgegeben wird.