Karlsruher Institut für Technologie
**Lehrstuhl für Programmierparadigmen**
Sprachtechnologie und Compiler, Sommersemester 2014
Dozent: Prof. Dr.-Ing. G. Snelting
Übungsleiter: Sebastian Buchwald `Sebastian.Buchwald@kit.edu`

Übungsblatt 6 | Ausgabe: 27.5.2014 | Besprechung: 2.6.2014

**Aufgabe 1:** *Auswertungsreihenfolgen*

Geben Sie eine attributierte Grammatiken an, die:

1. Nur synthetisierte Attribute enthält.
2. Nur ererbte Attribute enthält.
3. Ein Attribut enthält das sowohl synthetisiert als auch ererbt wird.
4. Stets mit einem Baumdurchlauf berechenbar ist.
5. Mehr als einen Baumdurchlauf für ihre Berechnung benötigen.
6. Nur für manche Bäume berechenbar ist. Es sollte sowohl berechenbare als auch nicht berechenbar Bäume geben, in denen alle Grammatikproduktionen vorkommen.

Geben Sie zur Verdeutlichung ggf. einen passenden Syntaxbaum an.

**Aufgabe 2:** *Namensanalyse*

Die meisten Programmiersprachen besitzen geschachtelte Namensräume. Oft finden sich auch voneinander unabhängige Namensräume für verschiedene Programmierkonstrukte. Betrachten Sie die Abbildungen 1, 2 und 3.

- Auf welche Definitionen beziehen sich die markierten (Bezeichner mit Subskript $x_1$) Referenzen? (Die Subskripte selbst sind natürlich nicht Teil des Programms, sondern hier nur zur Verdeutlichung der Aufgabenstellung).
- Beschreiben Sie wo in den Beispiele neue Namensraumschachteln entstehen.
- Gibt es unabhängig Namensräume?

```
struct x { int x; };
x1 k;
int x;
void f(void) { x2 = 20; k3 . x4 = 42; }
```

Abbildung 1: Beispiel 1 (C++ Code)

```
extern "C" int rand(void);

int main(void) {
    int foo;

 foo :
    if (rand()) {
        float foo ;

        for (float foo = 0; foo < 42; ++ foo1 ) {
            if (rand() < 13)
                goto foo2 ;
        }
         foo3 = 42;
    }
}
```

Abbildung 2: Beispiel 2 (C++ Code)

```
class Base {
    public int X , Y;
    public int f () {
        return X1 ;
    }
}
class Derived extends Base2 {
    public class Inner extends Base {
        public void func() {
            f3 ();
            X4 = 20;
            Y5 = 10;
        }
        private int Y ;
    }
    public int f () {
        return X6 ;
    }
    public void call_f(Base b ) {
         b7 . f8 ();
        ((Derived) b). f9 ();
    }
    public int X , Y;
}
```

Abbildung 3: Beispiel 3 (Java Code)

**Aufgabe 3:** *Praxis: Namensanalyse*

Unter `http://pp.info.uni-karlsruhe.de/lehre/SS2014/compiler/uebung/nameana.zip` befindet sich Java Sourcecode mit einem Parser und Interpreter. Die implementierte Sprache besitzt eine Zuweisungs- und einer Ausgabeoperation. Außerdem können mit { und } Namensbereiche geschachtelt werden. Die Verwaltung der Symboltabelle in SymbolTable.java wurde entfernt.

- Implementieren Sie die fehlende Funktionalität in SymbolTable.java

Beispiel für eine Eingabe:

```
{
    foo = "bar";
    print(foo);

    {
        foo = "bar2";
        print(foo);
    }

    print(foo);
}
```

Korrekte Ausgabe:

```
foo is bar
foo is bar2
foo is bar
```