Prof. Dr. Arnd Poetzsch-Heffter
Dr.-Ing. Jan Schäfer

Technische Universität Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Übungsblatt 9: Übersetzer und sprachverarbeitende Werkzeuge (SS 2011)

Hand Out: 29. Juni 2011
Hand In: 6. Juli 2011

Aufgabe 1 Prüfungsfragen zu den Vorlesungen

Überlegen Sie sich mindestens zwei Prüfungsfragen und tragen Sie diese unter `http://bit.ly/kJORSi` ein.

Aufgabe 2 Codegenerierung für Methoden

Erweitern Sie ihren Compiler um die Fähigkeit Code für beliebige statische Methoden und deren Aufrufe zu erzeugen.

a) Legen Sie Ihr Layout eines Stackframes, der Methodenpräambel und des -epilogs fest, und dokumentieren Sie es kurz schriftlich.

b) Implementieren Sie die oben beschriebene Erweiterung Ihres Compilers.

Aufgabe 3 Codegenerierung für globale Variablen

Erweitern Sie Ihren Compiler um die Möglichkeit Speicherplatz für statische Attribute von Klassen anzulegen und diese in Ausdrücken zu lesen und zu schreiben.

Aufgabe 4 IO-Bibliothek

Erweitern Sie Ihren Compiler um eine minimale Standardbibliothek. Diese enthält eine Klasse `IO`, s. unten. Diese Klasse wird implizit jedem Program zur Verfügung gestellt. Die Methoden lassen sich nicht direkt in MiniJava implementieren, deshalb wird `IO` fest in den Compiler eingebaut.

```
class IO {
  static void write(int i) { ... } // int in Terminal ausgeben
  static int read() { ... } // int vom Terminal einlesen
}
```

In der Vorlesung wurden einige Systemcalls vorgestellt, u.a. zum Ausgeben eines Integers. Der Systemcall zum Lesen eines Integers ist dort nicht dokumentiert. Er hat den System-Call-Code 5 und schreibt sein Ergebnis in `$v0`.

Hinweis: Um die Integration der Klasse `IO` möglichst einfach zu gestalten ist es möglich einen abstrakten Syntaxbaum für `IO` nach dem Parsen an den Parseroutput anzuhängen. Dabei repräsentiert man die Methodenrümpfe, die nicht in MiniJava programmierbar sind, durch einen Dummy. Beispielsweise können Sie die Deklaration von `MethodDecl` durch folgende Deklarationen ersetzen. Dann steht `InternalMethodBody()` für einen Methodenrumpf, der in den Compiler fest eingebaut ist.

```
MethodDecl ( Integer line, Integer column, Modifiers modifiers, ReturnType retType,
          Identifier ident, FormalParameters params, MethodBlock body )
MethodBlock = Block
          | InternalMethodBody ()
```

Die Namens- und Typanalyse kann dann weitgehend unverändert auf dem erweiterten Programm durchgeführt werden. Die Codegenerierung benötigt dann eine Sonderbehandlung für `InternalMethodBody()`.