Westfälische Hochschule
Gelsenkirchen Bocholt Recklinghausen
University of Applied Sciences

Fachbereich Informatik und Kommunikation
Prof. Dr. Ulrike Griefahn

MiniJava-Projekt (ÜSB)

Getting Started

25. 09. 2022

In diesem Projekt, das aus vier aufeinander aufbauenden Aufgaben besteht, realisieren Sie mit JavaCC einen Top-Down-Übersetzer, mit dem Sie MiniJava-Programme kompilieren können. Sie dürfen sehr gerne im Team (max. 3 Personen) arbeiten.

1 Die Sprache MiniJava

MiniJava[1] ist eine Teilmenge von Java. Die Semantik eines MiniJava-Programms wird durch die Semantik von Java-Programmen definiert.

Überladung von Methoden ist in MiniJava nicht erlaubt. Die MiniJava-Anweisung `System.out.println(...)` kann nur auf Integerzahlen angewendet werden. Der Zugriff mit `.length` kann nur auf Ausdrücke des Typs **`int`**`[]` angewendet werden.

1.1 Lexikalische Struktur

Bezeichner: Ein Bezeichner ist eine Folge von Buchstaben, Ziffern und Unterstrichen. Die Folge muss mit einem Buchstaben beginnen. Groß- und Kleinbuchstaben werden unterschieden, in der BNF für MiniJava als `IDENTIFIER` bezeichnet.

Integer-Literale: Eine Folge von Ziffern ist eine Integer-Konstante, die die entsprechende Integerzahl darstellt, in der BNF für MiniJava als `INTEGER_LITERAL` bezeichnet.

Operatoren: Binäre Operatoren sind `&&`, `<`, `+`, `-`, `*`, `.`, `[]`, `()`. Sie sind alle linksassoziativ. Unäre Operatoren sind: `!` und **`new`**. Beide sind rechtsassoziativ.

Kommentare: Ein Kommentar kann zwischen zwei beliebigen Token stehen. Es gibt zwei Arten von Kommentaren: Einzeilige Kommentare, die mit `//` beginnen und bis zum Ende der Zeile gehen, und mehrzeilige Kommentare, die mit `/*` beginnen und mit `*/` enden.

1.2 BNF für MiniJava

Im Folgenden finden Sie die BNF für MiniJava. Zur besseren Unterscheidung von Symbolen, die sowohl in der Sprache MiniJava als auch in der BNF vorkommen, wie die Klammersymbole und der $*$, sind die Terminalzeichen der Sprache in blau

[1] Die Definition von MiniJava wurde dem Buch „Modern Compiler Implementation in Java“ von Andrew W. Appel (Cambridge University Press, October 2002) entnommen.

und der Schriftart Courier gesetzt, während die Variablen der Grammatik und die Symbole der BNF in schwarz und der Schriftart Times gesetzt sind.

Die Notation N* für eine Variable N hat die Bedeutung, dass N beliebig häufig wiederholt aber auch ganz weggelassen werden kann. Die Notation N? gibt an, dass N optional ist.

Program	::=	MainClass ClassDeclaration* <EOF>
MainClass	::=	**class** Id { **public static void** `main ( String [ ]` Id) { Statement } }
ClassDeclaration	::=	**class** Id (**extends** Id)? { VarDeclaration* MethodDeclaration* }
VarDeclaration	::=	Type Id **;**
MethodDeclaration	::=	**public** Type Id ((Type Id (**,** Type Id)*)?) { VarDeclaration* Statement* **return** Expression **;** }
Type	::=	**int** [ ] | **boolean** | **int** | Id
Statement	::=	{ Statement* } | **if** (Expression) Statement **else** Statement | **while** (Expression) Statement | `System.out.println` (Expression) **;** | Id = Expression **;** | Id [Expression] = Expression **;**
Expression	::=	Expression (&& | < | + | - | *) Expression | Expression [Expression] | Expression . `length` | Expression . Id ((Expression (**,** Expression)*)?) | `<INTEGER_LITERAL>` | **true** | **false** | Id | **this** | **new int** [Expression] | **new** Id ( ) | ! Expression | (Expression)
Id	::=	`<IDENTIFIER>`

Im Projekt, das Ihnen zur Verfügung gestellt wird (s. u.), finden Sie acht korrekte MiniJava-Programme.

2 Benötigte Software

Das Projekt, das Sie für die Aufgaben als Gerüst vorgegeben bekommen, wurde mit IntelliJ IDEA entwickelt. Die folgende Beschreibung kann sich daher an manchen Stellen auf IntelliJ beziehen. Da Sie mit dem Projekt jedoch nur den Quellcode erhalten, können Sie auch jede andere geeignete IDE verwenden.

JavaCC : Laden Sie sich die JavaCC-Bibliothek über `http://javacc.org/` herunter.

Jasmin : Laden Sie sich die Jasmin-Bibliothek über `http://jasmin.sourceforge.net/` herunter.

JavaCC-Plugin : Für IntelliJ IDEA gibt es ein JavaCC-Plugin, das ein Syntax-Highlighting für die Parser-Spezifikation zur Verfügung stellt. Ebenfalls werden die first-Mengen der Variablen der Grammatik dargestellt.

3 MiniJava-Projekt `minijava`

Als Zusatzmaterial für das Projekt erhalten Sie Quellen, die Ihnen den Einstieg in JavaCC und das Testen erleichtern sollen. Die Verzeichnisstruktur des Ordners ist wie folgt aufgebaut:

- `MiniJavaC/`: Projektordner
  - `tools/`: Entpacken Sie hier die in Abschnitt 2 herunter geladenen Bibliotheken und legen Sie die jar-Dateien ohne Versionsnummer in diesem Verzeichnis ab.
    - `jasmin.jar`: Jasmin-Compiler (benötigt für Blatt 4)
    - `javacc.jar`: JavaCC-Compiler
  - `samples/`
    - `src/`: Acht (korrekte) MiniJava-Programme (Endung `.mjava`)
    - `out/`: Zielverzeichnis für alle im Übersetzungsprozess erzeugten Dateien
  - `out/` Alle im Übersetzungsprozess erzeugten Dateien werden hier abgelegt.
  - `src/`:
    - `parser/`:
      - `MiniJavaParser.jj`: Die Datei enthält die Spezifikation für den MiniJava-Parser. Sie ist bereits zu einem Teil vorgegeben. Sie müssen sie ergänzen. (Aufgaben 1 bis 2).
      - Die Java-Programme, die von JavaCC erzeugt werden, werden ebenfalls in diesem Verzeichnis erzeugt.
    - `syntaxtree/`: Die Klassen für den abstrakten Syntaxbaum (benötigt ab Blatt 2)
    - `visitor/`: Visitorklassen, mit denen der Syntaxbaum durchlaufen werden kann, um diesen in eine Datei auszugeben (benötigt für Blatt 2), die Symboltabelle zu füllen und die Typprüfung durchzuführen (benötigt für Blatt 3) und Jasmin-Code zu erzeugen (benötigt für Blatt 4).
  - `README.md`: Enthält alle Schritte, mit der der Parser erzeugt wird, und die Schritte für die Übersetzung und Ausführung von MiniJava-Programmen.

4 Interne Prozessschritte des MiniJava-Compilers

Die JavaCC-Spezifikation für den MiniJava-Compiler erstellen Sie in den Aufgaben 1 und 2 (Datei `MiniJavaParser.jj`). Aus der JavaCC-Spezifikation wird dann mit JavaCC Java-Quellcode für den lexikalischen Scanner und den Parser erzeugt. Dieser Parser kann MiniJava-Programme parsen und sie in einen abstrakten Syntaxbaum überführen. In Aufgabe 3 realisieren Sie über das Visitor-Pattern die Typprüfung für MiniJava-Programme (Datei `TypeVisitor.java`). Die Erzeugung

der Symboltabelle ist bereits vorgegeben. Aufgabe 4 hat die Erzeugung von Jasmin-Code zum Ziel (Datei `JasminVisitor.java`). Der Jasmin-Code kann schließlich mit dem Jasmin-Assembler (`http://jasmin.sourceforge.net/`) in `class`-Dateien übersetzt und ausgeführt werden.