Kurzdokumentation zum Projekt „Compilerbau“

COMPILERBAU PROJEKT GRUPPE 2

Projektthema	Erstellen eines Compilers
Projektgruppenname	Gruppe 2
Kurs	TINF 2018
Projektmitglieder	Blessing, Corinne (Projektleitung) Dietrich, Benita (Testing) Finkbeiner, Paul (Typisierung) Götz, Tobias (Scanner/Parser) Herter, John (Abstrakte Syntax) Schwarz, Jonas (Codegenerierung)
Projektbetreuer	Prof. Dr. Martin Plümicke
Abgabetermin	30.06.2020

Inhaltsverzeichnis

Dokumentation Projektteam Gruppe 2

Nachfolgend werden die Projektdaten genannt sowie die Projektverantwortlichkeiten geklärt. Es wird der Projektverlauf nach Eintritt in die Praxisphase geschildert. Eine kurze Funktionserklärung erläutert das Projekt genauer. Das Dokument schließt mit der Erläuterung zur Benutzung der Anwendung. Das Klassendiagramm, das den Aufbau der abstrakten Syntax zeigt, befindet sich im Anhang.
Der Abschluss des Projektes findet in Form einer Präsentation sowie der Demonstration des Compilers zu einem gesonderten Termin statt.

Projektverantwortlichkeiten

Die im Projekt definierten Rollen wurden weitestgehend eingehalten. Als Projektleiterin plante **Corinne Blessing** die wöchentlichen Treffen und trug die Verantwortung für die termingerechte Fertigstellung des Projektes. Bei Problemen mit dem Versionsverwaltungstool Git war sie als Ansprechpartnerin zur Klärung der Probleme beauftragt. Sie erstellte die Kurzdokumentation, die Schnittstellen und das Klassendiagramm zur abstrakten Syntax.
Benita Dietrich war als Verantwortliche für Tests dafür zuständig, die Testsuiten zu implementieren. So generierte sie 23 Testsuiten, die sich im Ressourcenordner befinden. Im eigentlichen Testordner (der sich in vier weitere schematisch unterteile Unterordner gliedert), sind insgesamt 46 weitere Testklassen aufgeführt. In den Tests sind Klassen in verschiedenen Komplexitätsgraden aufgeführt, um nicht nur einfache Fälle abzuprüfen. Zur besseren Übersichtlichkeit wurde das Test Package anders strukturiert als im Java Package üblich.
Die Erstellung des Scanners und Parsers wurde von **Tobias Götz** realisiert. Semantisch wurden zwei Ordnerstrukturen eingerichtet, die eine beherbergt die Adapter-, die andere die Converter-Klassen. Tobias arbeitete eng mit **John Herter** zusammen, der für die abstrakte Syntax zuständig war. Die Entwicklung der beiden Teile ging Hand in Hand und wurde durch regelmäßige Pair-Programming-Einheiten unterstützt.
Paul Finkbeiner war für die Entwicklung der Typisierung verantwortlich. Semantisch ist der Ordner zusätzlich in die Teilbereiche „expression“, „statement“ und „stmtExpr“ unterteilt. In den wöchentlichen Gesprächen erfolgte die Abstimmung, wie weit Tobias und John waren und welchen Teil Paul als nächstes beginnen kann.
Jonas Schwarz war für die Generierung des Codes verantwortlich. Seine Arbeit baut auf dem Rest der Gruppe auf und sorgt dafür, die Klasse zu generieren.

Projektverlauf

Eine pandemiebedingte Verlagerung der Projektdauer führte zu einer knapp vierwöchigen Implementierungsphase außerhalb des Theoriesemesters.
Während den vier Wochen fand eine wöchentliche Abstimmung im Videochat statt. Darin wurden Probleme thematisiert und das Vorgehen der folgenden Woche geplant. Jedes Teammitglied hatte die Gelegenheit, seine Arbeit zu präsentieren und die sich mit den anderen Teammitgliedern über kritische Stellen auszutauschen.
Eine besonders enge Zusammenarbeit fand zwischen den Gruppenmeetings zwischen dem Scanner und Parser und der Abstrakten Syntax statt. Die enge Absprache begann schon während dem laufenden Theoriesemester und hat sich bewährt, da eine Strukturänderung in der Grammatik zum Zusammenbruch der nachfolgenden Strukturen führte. Mit Problemen wurde dynamisch umgegangen. Wurde ein Teammitglied aufgrund eines Problems in seiner Weiterarbeit behindert, fand in der Regel noch am selben Tag ein Termin zwischen den Beteiligten statt. Im Anschluss wurde das Ergebnis gruppenintern kommuniziert. Kontinuierliches Testen diente zur Kontrolle des Codes sowie zur Qualitätskontrolle.

Funktion des Compilers

Der Einstieg erfolgt über die Klasse „JavaCompiler.java“. Hier ist die Logik für die Übergabe der zu kompilierenden Datei implementiert. Über die Kommandozeile wird die gewünschte Datei eingelesen. Die in der Datei befindlichen Informationen werden über die abstrakte Syntax mittels des ANTLR-Tools in einen Parse-Tree übersetzt. Die Adapterklassen erstellen aus den jeweiligen Objekten des Parse Trees die korrespondierenden Objekte der abstrakten Syntax und speichern diese in einer ArrayList. Im Anschluss wird der TypeCheck aufgerufen. Diese Funktionalität führt zum Schreiben der Typen in die einzelnen Bestandteile. Mit den Typen kann der Codegenerierer aufgerufen werden und die Klasse aufbauen.

Benutzung des Compilers

Über die Kommandozeile wird mit dem Befehl

```
javac /Pfad/zu/JavaCompiler.java 'zu kompilierende Klasse'
```

die zu kompilierende Datei eingelesen und die .class-Datei direkt im Ordner gespeichert.