Compilerbau
– Prüfungsleistung –

Dozent: Julian Schmidt, Andeeas Stadelmeier, Martin Plümicke

SS 2025

Spezifikation

Deklarationen:
- Σ: Eingabe–Alphabet
- JC: Menge aller syntaktisch korrekten Java–Klassen mit folgenden Einschränkungen:
  - keine generischen Klassen
  - keine abstrakten Klassen
  - keine Interfaces
  - keine Threads
  - keine Exceptions
  - keine Arrays
  - als Basistypen sind nur `int`, `boolean` und `char` zugelassen
  - keine Packages
  - keine Imports
  - keine Lambda–Expressions
- BC: Menge aller Bytecode–Files

Eingabe: $p \in \Sigma^*$

Vorbedingung: $\emptyset$

Ausgabe: $bc \in \mathsf{BC}^* \cup \{error\}$

Nachbedingungen:
- falls $p \in (JC)^*$, so ist $bc \in (\mathsf{BC})^*$ und p wird nach bc übersetzt wie es durch die Sprache Java definiert ist.
- falls $p \notin (JC)^*$, so ist $bc = error$.

Vorgehen

Arbeitsteam: Der Java–Compiler wird in einem Team von 5–7 Personen erstellt. Das Team wird nochmals unterteilt:

- **Gemeinsame Aufgabe**

  GIT-Repository: Einrichten eines GIT-Repositories auf den DHBW GITEA-Server

  Abstrakte Syntax: Aufbau der abstrakten Syntax aus dem Parsetree (Output von ANTLR)

- **Scannen/Parsen/Grammatik (1–2 Personen)**

  Grammatik (nur bei Bearbeitung durch 2 Personen): Erstellen einer Mini-Java-Grammatik an Hand der Spezifikation

  ANTLR–File: Erstellen des G4–Files

- **Semantische Analyse (1–2 Person)**

  Typisierung: Typisierung der abstrakten Syntax

  Vererbung (nur bei 2 Personen)

- **Codeerzeugung (1 Person):** Erzeugung des Bytecodes mit ASM
- **Tester (1 Person)**
  - Testsuite von Java–Files, die alle implementierten Features abdecken.
  - Händische Übersetzung aller Java-Files der Testsuite in die abstrakte Syntax (als Test–Eingaben für den Typ-Checker)
  - JUnit–Tests für Parser und Erzeugung der abstrakte Syntax
  - Händische Typisierung aller Testfälle der abstrakten Syntax (als Eingabe für die Codeerzeugung)
  - JUnit–Tests für die Typisierung
  - JUnit–Tests für den gesamten Compiler mit Hilfe von Reflections.
- **Projektleiter (1 Person)**
  - Projektleitung (Gesamtverantwortung)
  - Wöchentliche Berichte (Kurzpräsentation) über den Projektfortschritt
  - **UML-Klassendiagramm:** Erstellung und Aktualisierung eines Klassendiagramms
  - Definition der Schnittstellen, insbesondere der abstrakten Syntax
  - Zusammenfügen der Projektteile
  - Codesharing über git
  - Projektdokumentation

Prüfungsleistung

Die Arbeitsleistung wird bewertet an Hand

- des Gesamtergebnis des Teams
- der Arbeitsleistung jeder/s Studierenden

an Hand folgender Kriterien:

- Projektergebnis

- wöchentlicher Projektfortschritt
- Mitarbeit im Team

Das Projektergebnis muss folgendes beinhalten:

- (Kurz)dokumentation aus der hervorgeht welche Leistung der jeweiligen Studierende erbrachten hat.
- Im Teilprojekt muss folgendes vorliegen:
  - Eine Testsuite von Java–Programmen, für die der Compilerteil funktioniert.
  - Präsentation des Programms an Hand der erstellen Testsuites.
- Durchgehendes Beispiel, für das der gesamte Compiler funktioniert.
- **Abgabetermin: Letzte Semesterwoche**