Compilerbau
– Prüfungsleistung –

Dozent: Martin Plümicke, Andeeas Stadelmeier

SS 2024

Spezifikation

Deklarationen:
- Σ: Eingabe–Alphabet
- JC: Menge aller syntaktisch korrekten Java–Klassen mit folgenden Einschränkungen:
  - keine generischen Klassen
  - keine abstrakten Klassen
  - keine Vererbung
  - 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 4–6 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

  Dokumentation: Erstellen der Dokumentation

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

  Scannen: `alex`–File oder Scanner von Hand programmieren

  Grammatik: Erstellen einer Mini-Java-Grammatik an Hand der Spezifikation

  Parsen: Erstellen des `happy`–Files oder des Kombinator–Parsers und Aufbau des abstrakten Syntaxbaums

- **Semantische Analyse:** Typisierung der abstrakten Syntax **(1 Person)**
- **Codeerzeugung (2–3 Personen):**
  - Aufbau eines abstrakten ClassFiles (1 Person)
  - Konstantenpool (1 Person)
  - **Nur bei Bearbeitung durch 3 Personen:** Umwandlung des ClassFiles in Bytecode (1 Person)
- **ggf. 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)
  - Händische Übersetzung aller Testfälle der abstrakten Syntax in getypte abstrakte Syntax (als Test–Eingaben für den Code-Generierer).
  - Händische Übersetzung aller Testfälle der getypten abstrakten Syntax in abstrakten Bytecode.
  - Automatische Tests, die die jeweiligen Testsuite mit den implementierten Funktionen des Teams vergleichen

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**