# Compilerbau
# – Prüfungsleistung –
# Java–TX

Dozent: Julian Schmidt, Martin Plümicke

SoSe 2026

## Spezifikation

**Deklarationen:**
- $\Sigma$: 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 Personen erstellt. Das Team wird nochmals unterteilt:

- **Gemeinsame Aufgabe**

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

  **UML-Klassendiagramm:** Erstellung und Aktualisierung eines Klassendiagramms

  **Abstrakte Syntax:** Aufbau der abstrakten Syntax aus dem Parsetree (Output von ANTLR)
  Die abstracte Syntax wird in Java–TX erstellt.

  **Schnittstellen:** Programmierung der Schnittstellen zwischen den Projektteilen

  **Dokumentation:** Erstellen der Dokumentation

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

  **ANTLR–File:** Erstellen des G4–Files aus einer gegebenen Java G4-Datei. Diese Teilaufabe wird in Java gelöst.

- **Semantische Analyse (1 Person)**

  **Typisierung:** Typisierung der abstrakten Syntax erfolgt in Java–TX.

- **Codeerzeugung (1 Person):** Erzeugung des Bytecodes mit ASM
  Die Codeerzeugung wird in Java–TX erstellt, wobei die Java–Methoden der ASM–Bibliothek verwendet werden.

- **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.

## 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**

**Bei der Bewertung der Implementierung in Java–TX wird berücksichtigt ob und ggf. welche Schwierigkeiten durch Java–TX bzw. durch den Compiler aufgetreten sind.**