Automaten und formale Sprachen
Scanner und Parser

Aufgabe 1 Funktionsweise eines Compilers

Klone das Projekt 🗀 `rechenmaschine` und öffne es in BlueJ. Das Programm ist eine (vereinfachte) Umsetzung eines Compilers (und Interpreters) für *Plusterme*. (Auf Arbeitsblatt 10 findest du den DEA zur Sprache.)

Analysiere das Programm und wie der Übersetzungsvorgang eines Compilers implementiert wurde. Beachte auch die Verwendung der `switch`-Anweisung. (Recherchiere ihren Aufbau, falls sie dir neu ist.)

Welche Aufgabe hat in diesem Beispiel die `scan`-Methode, welche die `parse`-Methode? Könnte man beide zur Vereinfachung auch zusammenfassen?

💡 **Tipp:** *Teste das Programm mit verschiedenen Eingaben und studiere vor allem die Fehlermeldungen bei ungültigen Plustermen.*

Aufgabe 2 Minusterme

Ändere das Programm so ab, dass statt *Plustermen* nun *Minusterme* erkannt und berechnet werden.

Erweitere das Programm dann so, dass als Rechenoperation nun Addition *und* Subtraktion erlaubt sind.

Aufgabe 3 Rechenterme

Auf Arbeitsblatt 5 wurde die Sprache der gültigen Rechenterme (ohne Klammerung) definiert.

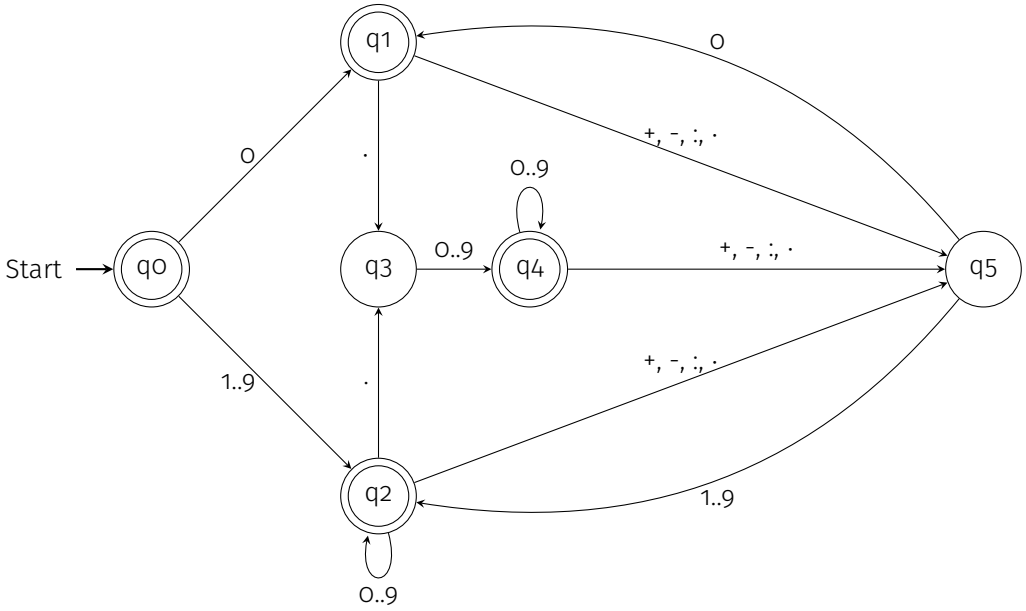

Abbildung 1: [NEA] Rechenausdruck

Passe das Programm schrittweise an (Scanner, Parser, Interpreter), so dass gültige Rechenterme übersetzt und korrekt ausgerechnet werden. Achte auch auf aussagekräftige Fehlermeldungen bei der Übersetzung.

💡 **Hinweis:** Für die Interpretation des Rechenterms ignorieren wir die Punkt-vor-Strich Regel und rechnen einfach von links nach rechts.

★ Aufgabe 4 Semantische Analyse

Die Sprache für gültige Rechenterme hat noch ein Problem: Tauscht im Rechenterm bei einer Division als Divisor die 0 (Null) auf, dann kann das Ergebnis nicht berechnet werden, auch wenn die Eingabe syntaktisch korrekt ist.

Implementiere eine neue Methode `boolean analysiere()`, die die Tokenliste darauf prüft, ob nach einem Geteiltzeichen (`:`) eine Null auftaucht und mit einer entsprechenden Fehlermeldung abbricht.