Höhere Programmierkonzepte
Praktikum V

Prof. Dr. Nikolaus Wulff

5. – 14. Januar 2021

1 Der *Axela* Function Service

```
package de.lab4inf.axela.script;
import java.util.function.ToDoubleFunction;

@FunctionalInterface
public interface AxelaFunction extends ToDoubleFunction<double[]> {
  /**
   * Function evaluation mapping tuple x=(x1,...,xn) to y = f(x1 ,..., xn)
   * using varargs.
   *
   * @param x the double array tuple x1 ,..., xn
   * @return f(x1 ,..., xn)
   */
  default double apply(double... x) {
    return applyAsDouble(x);
  }
}
```

Listing 1: Deklaration einer Schnittstelle für Funktionen $\mathbb{R}^n \to \mathbb{R}$.

Die ersten "Gehversuche" mit mit dem Parser im Praktikum IV waren erfolgreich und es gilt freie Variablen und Funktionen zu integrieren, um die gewünschte Funktionalität endgültig auszuimplementieren.

Aufgabe

Die EBNF Grammatik und die Knotenhierarchie enthalten bereits Variablen- und Funktionsknoten. Diese gilt es nun geeignet "scharf zu schalten", so dass Sie mathematische Ausdrücke und Funktionen frei definieren und auswerten können. Ähnlich zu Praktikum IV gibt es auch für Praktikum V einen JUnitTest, der Ihnen die Verwendung der Scriptsprache illustriert und einen Integrationstest liefert - **aber eigene Funktionale Tests nicht ersetzt!**

1. Variablen sollen frei definierbar sein, sind per Zuweisung zu setzen und deren Werte können innerhalb des Script abgerufen werden.

2. Funktionen sollen die Schnittstelle `AxelaFunction` aus Listing (1) implementieren, die `ToDoubleFunction` erweitert und es erlaubt mit Varargs vom primitiven Typ `double` ohne Wrapper Klassen zu arbeiten.

3. Bei den Funktionen kann es sich sowohl um frei per JavaCC definierte Funktionen des Script handeln, als auch um „klassische Java-Methoden“ der Klasse `java.lang.Math` wie z.B. Math.sin, Math.pow etc. die geeignet in der ScriptEngine eingebaut werden müssen.

4. Funktionen können sowohl innerhalb des Script als auch außerhalb des Script als “freie Funktionen” innerhalb einer Java Anwendung verwendet werden. Sie werden für diese Funktionalität einen weiteren Besucher benötigen, der Ihnen erlaubt Funktionen anstatt double Werte zurückzuliefern[1]. Orientieren Sie sich für die Verwendung von freien Funktionen an den entsprechenden testFunctionXXX Methoden des Praktikum5Test.

5. Das einheitliche Verhalten Ihrer Implementierung wird mit dem hinterlegten `Praktikum5Test` überprüft, dieser Test kann aber per Vererbung erweitern werden.

   Diese Testklasse selber darf nicht verändert werden[2]!

6. Entwickeln Sie für ihre interne(n) Besucher eigene JUnit funktionale white-box Tests, um die Testüberdeckung zu verbessern und Ihre JUnit-Kenntnisse zu vertiefen.

[1] Ähnliches gilt für Matrizen und Vektoren, aber dies wird für Praktikum V nicht erforderlich sein.

[2] D.h. weder der Paket- noch der Klassenname werden geändert und es werden auch keine Methoden hinzugefügt oder gelöscht, gleiches geht für Konstanten und Variablen etc. Dieser zur Verfügung gestellte Test ist “read-only” und ein Teil der Spezifikation.