09 – Verständnisfragen zur selbstständigen Beantwortung

(1) Was wird unter „zeilenweises Debugging“ verstanden?

(2) Erläutern Sie das Problem der ursprünglichen Lösung zur Ermittlung von pi nach Madhava-Leibniz.

(3) Denken Sie über die Lösung der Methode `addVektoren` nach. Sind die Stichprobenbeispiele günstig gewählt? Was könnte hier vielleicht schief gehen?

09 – Aufgabensammlung

(1) Das Wallis-Produkt von John Wallis 1655 formuliert und gibt eine Näherung für $\pi/_2$ mit der folgenden Vorschrift:

$$\frac{\pi}{2} = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \dots = \prod_{i=1}^{\infty} \frac{4i^2}{(2i-1) \cdot (2i+1)}$$

Implementieren Sie dieses Verfahren und geben Sie damit eine Näherung für π an. Wenn Ihnen das keine Schwierigkeiten bereitet hat, dann erweitern Sie Ihre Funktionssammlung zur Näherung von π doch noch um die beiden folgenden:

$$\frac{\pi}{2 \cdot \sqrt{2}} = 1 + \frac{1}{3} - \frac{1}{5} - \frac{1}{7} + \frac{1}{9} + \frac{1}{11} - \dots \text{ (Newton)}$$

$$\frac{\pi}{3 \cdot \sqrt{3}} = 1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{5} + \frac{1}{7} - \frac{1}{8} - \dots \text{ (Euler)}$$

Wenn Sie das Thema der Annährungen von π noch weiter fesselt, dann schauen Sie sich die sehenswerte Wikipedia-Seite dazu an und implementieren Sie weitere Verfahren [1].

[1] https://en.wikipedia.org/wiki/Approximations_of_π

09 – Aufgabensammlung

(2) Wir haben die Madhava-Leibniz-Reihe als Funktion bereits implementiert. Leider ist die Konvergenzgeschwindigkeit der Reihe sehr langsam, denn n korrekte Nachkommastellen erhalten wir nach $^1/_2\,10^n$ Iterationen. Erweitern Sie den folgenden Programmabschnitt zum Abgleich mit der bereits definierten Konstanten `Math.PI`:

```
public static void main(String[] args) {
   int[] durchlauf = {5, 50, 500, 5000, 50000, 500000,
                      5000000, 50000000, 500000000};
   char[] pi_leibniz, pi_biblio;
   for (int j=0; j<durchlauf.length; j++) {
      long startZeit = System.currentTimeMillis();
      double piLeibniz = piMadhavaLeibniz(durchlauf[j]);
      long stopZeit = System.currentTimeMillis();
      // wir wandeln die beiden double-Werte in Arrays
      // vom Datentyp char um
      pi_leibniz = (Double.toString(piLeibniz)).toCharArray();
      pi_biblio  = (Double.toString(Math.PI)).toCharArray();

      // Ausgabe der Iterationsanzahl und der benötigen Zeit
      System.out.println("Iterationen: " + durchlauf[j] +
         " (" + (stopZeit-startZeit)+" ms)");
      // Ausgabe des gemeinsamen Teils
      System.out.println("Genauigkeit: " +
         (String)(Double.toString(Math.PI)).subSequence(0, s));
      System.out.println();
}
```

```
/* TODO:
 * In diesem Abschnitt sollen die ersten Zeichen der
 * beiden Listen pi_leibniz und pi_biblio verglichen
 * und die Anzahl der Übereinstimmungen in s
 * gespeichert werden
 */
```

Falls Sie Aufgabe 2 erfolgreich bearbeitet haben, könnten Sie die Näherung vom Wallis-Produkt in die Vergleichsklasse ebenfalls einbauen.