Übungsblatt 2

Adrian Schollmeyer

Aufgabe 1

(a) $d = 6; i = 6; b = \mathit{true}$

(b) $d = 6; i = 5; b = \mathit{true}$

(c) $d = 2; i = 6; b = \mathit{true}$

(d) $d = 2; i = 7; b = \mathit{false}$

(e) $d = 4.75; i = 7; b = \mathit{false}$

(f) $d = 4.75; i = 49; b = \mathit{false}$

Aufgabe 2

(a) primitiv darstellbar; boolean

(b) primitiv darstellbar; int, long, float, double

(c) nicht primitiv darstellbar

(d) nicht primitiv darstellbar

(e) nicht primitiv darstellbar

(f) primitiv darstellbar; float, double

(g) primitiv darstellbar; char

(h) primitiv darstellbar; long

Aufgabe 3

```
static public int sum1(int[] arr)
{
  int buf = 0;
  for (int i = 0; i < arr.length; i++) {
    buf += arr[i];
  }
  return buf;
}

static public int sum2(int[] arr)
{
  int buf = 0;
  int i = 0;
  while (i < arr.length) {
    buf += arr[i++];
  }
  return buf;
}

static public int sum3(int[] arr)
{
  if (arr.length == 0)
    return 0;

  int buf = 0;
  int i = 0;
  do {
    buf += arr[i++];
  } while (i < arr.length);
}
```

Aufgabe 4

Listing 1: PiMain.java

```
package me.adrian;

public class PiMain
{
    static public double piN()
    {
        double cache = 4. / 1.;
        int currentDivisor = 1;
        double lastVal = 0.;
        short factor = 1;
        long nIterations = 0;

        while (Math.abs(cache - lastVal) > .00000001) {
```

```
            currentDivisor += 2;
            factor *= -1;
            lastVal = cache;
            cache += factor * (4. / currentDivisor);
            nIterations++;
            //Thread.yield();
        }

        System.out.println("--> Complete iteration count: " + nIterations);
        return cache;
    }

    static public void main(String[] args)
    {
        System.out.println("--> Pi approx. " + piN());
    }
}
```

Der Algorithmus benötigt 199 999 997 Iterationen und liefert das Ergebnis:

$$\pi \approx 3.1415926485894077$$

Aufgabe 5

(a) $n_{Teilfolgen} = n_{Elemente}!$

(b)

Listing 2: PSumMain.java

```
package me.adrian;

public class PSumMain
{
    static public long partialSum(int[] src, int begin, int end)
    {
        long res = 0l;
        for (int i = begin; i < end; i++) {
            try {
                res += src[i];
            } catch (ArrayIndexOutOfBoundsException e) {
                e.printStackTrace();
                break;
            }
        }
        return res;
    }

    static public long[] maxPartialSum(int[] src)
    {
```

```
        if (src.length < 1) {
            return new long[] { -1l, -1l };
        }
        long lastMax = 0; // Nicht Long.MIN_VALUE, da 0 minimale Teilsumme
            fuer leere Teilmenge sein darf
        int lastB = -1;
        int lastE = -1;
        for (int b = 0; b < src.length; b++) {
            for (int e = 1; e <= src.length; e++) {
                long partialSum = partialSum(src, b, e);
                if (partialSum > lastMax) {
                    lastMax = partialSum;
                    lastB = b;
                    lastE = e;
                }
            }
        }
        return new long[] { lastMax, lastB, lastE - 1l };
    }

    static public void main(String[] args)
    {
        //int[] a = { 31, -41, 59, 26, -53, 58, 97, -93, -23, 84 };
        int[] a = {-1, -1, -1, 1};
        long[] res = maxPartialSum(a);
        System.out.println("Maximale Teilsumme: " + res[0]);
        System.out.println("i = " + res[1]);
        System.out.println("j = " + res[2]);
    }
}
```