TrapezoidQ

N	h	Q	R
2	0,5	1,078939231666	-
4	0,25	1,075476649096	-0,00173
8	0,125	1,074555363351	-4,606428724285295E-04
16	0,0625	1,074320770500	-1,1729642538038650E-04
32	0,03125	1,074261837012	-2,9466744117301330E-05
64	0,01563	1,074247085483	-7,375764773498441E-06
128	0,00781	1,074243396460	-1,8445112769382990E-06
256	0,00391	1,074242474133	-4,611634915052677E-07
512	0,00195	1,074242243547	-1,1529310306457320E-07
1024	0,00098	1,074242185900	-2,882341521015520E-08

evenTrapezoidQ

N	h	Q	R
2	0,5	1,075046939171	-
4	0,25	1,074322454906	-9,05605331334991E-05
8	0,125	1,074248268103	-9,273350402322356E-06
16	0,0625	1,074242572883	-7,119024244550687E-07
32	0,03125	1,074242192516	-4,754592405897817E-08

2. Для задания 2 я взял коэфициенты из книги Вакульчик для k = 5:
x3 = 0 , x4 = − x2 = 0,5384693101, x5 = − x1 = 0,9061798459,
A3 = 0,5688888899 , A2 = A 4 = 0,4786286705, A1 = A5 = 0,2369268851;
Получилось:
1,0742420793859562

```
package base;

import static java.lang.Math.*;

import java.util.function.IntToDoubleFunction;
import java.util.stream.IntStream;

public class Solution {
	public static final double a = 1;
	public static final double b = 2;
	public static final int k = 5;
	public static final double epsilon = 1e-7;

	public static double f(double x) {
		return x / (1 + log(x));
	}

	public static double h(int N) {
		return (b - a) / N;
	}

	public static double trapezoidQ(int N) {
		double h = h(N);
		double sum = IntStream.range(1, N)
					.parallel()
```

```
                                        .mapToDouble(i -> a + i*h)
                                        .map(x -> f(x))
                                        .sum();
        return h * ((f(a) + f(b)) / 2  + sum);
    }

    public static double evenSimpsonQ(int N) {
        double h = h(N);
        double firstSum = IntStream.iterate(1, i -> i+=2)
                                        .takeWhile(i -> i < N)
                                        .mapToDouble(i -> a + i*h)
                                        .map(x -> f(x))
                                        .sum();
        double secondSum = IntStream.iterate(2, i -> i+=2)
                                        .takeWhile(i -> i < N-1)
                                        .mapToDouble(i -> a + i*h)
                                        .map(x -> f(x))
                                        .sum();
        return h *(f(a) + f(b) + 4*firstSum + 2*secondSum) / 3;
    }

    private static void task1(int m, IntToDoubleFunction quadrature) {
        int N = 2;
        double currentQ = quadrature.applyAsDouble(N);
//      System.out.println(currentQ);
        N *= 2;
        double nextQ = quadrature.applyAsDouble(N);
        double R = (nextQ - currentQ) / (1<<m - 1);
//      System.out.println(R);
        while(abs(R) > epsilon) {
            currentQ = nextQ;
//          System.out.println(currentQ);
            N *= 2;
            nextQ = quadrature.applyAsDouble(N);
            R = (nextQ - currentQ) / (1<<m - 1);
//          System.out.println(R);
        }
        System.out.println(nextQ);
    }

    private static void task2() {
        double[] x = new double[] {-0.9061798459, -0.5384693101, 0, 0.5384693101, 0.9061798459};
        double[] A = new double[] {0.2369268851, 0.4786286705, 0.5688888899, 0.4786286705,
0.2369268851};
        double answer = IntStream.range(0, k)
                                        .mapToDouble(i -> (b-a) * A[i] * f((a+b)/2 + (b-a)*x[i]/2) / 2)
                                        .sum();
        System.out.println(answer);
    }

    public static void main(String[] args) {
        task1(4, Solution::evenSimpsonQ);
        task2();
    }
}
```