

```
1 package euler;
2
3 /**
4  * Kelas untuk menghitung jumlah setiap elemen dalam sebuah deret aritmatik. Metode
5  * yang dipakai adalah metode Gauss ditambah dengan prinsip Inklusi-Eksklusi.
6  */
7 public class Gauss {
8     /**
9      * Fungsi untuk menjumlahkan semua bilangan kelipatan a1, a2, a3 ... an dari
10     * 1 sampai limit (inclusive). Sebagai contoh: Jumlah bilangan kelipatan 2 dan 3
11     * dari 1 sampai 100 adalah sum(new long[]{2L, 3L}, 0, 2, 100L).
12     *
13     * Bagian terkeren dari fungsi ini : ga ada if! Kekekekekek
14     *
15     * @param numbers array dari bilangan-bilangan yang diinginkan. Asumsi : array
16     * tidak null. Asumsi : setiap bilangan koprima (FPB-nya 1) dengan bilangan
17     * lainnya.
18     *
19     * @param lower batas bawah dari array. Asumsi : lower berada di dalam rentang
20     * yang diperbolehkan : 0 s/d numbers.length - 1
21     *
22     * @param length panjang array. Asumsi : lower + length <= numbers.length
23     *
24     * @param limit batas maksimal deret (inclusive). Asumsi : limit adalah bilangan
25     * bulat tidak negatif.
26     *
27     * @return jumlah bilangan kelipatan a1, a2, a3... an dari 1 sampai limit
28     */
29     public long sum(long[] numbers, int lower, int length, long limit){
30         assert(numbers != null);
31         // Siaga 2 : Pemeriksaan asumsi fpb(ai, aj) = 1 tidak dilakukan!!
32         assert(lower >= 0 && lower < numbers.length);
33         assert(lower + length <= numbers.length);
34         assert(limit >= 0L);
35
36         // Menghitung ukuran array : 2^length - 1
37         // Siaga 1 : Arithmetic Overflow!
38         // Siaga 1 : Out of memory!
39         int size = (1 << length) - 1;
40
41         // Deklarasi array
42         long[] ar = new long[size];
43
44         // Inisialisasi setiap elemen pada array dengan -1
45         for (int i = 0; i < size; i++) ar[i] = -1L;
46
47         // Deklarasi & inisialisasi return value
48         long result = 0L;
49
50         // Deklarasi & inisialisasi index array yang akan ditulisi
51         int idx = 0;
52
53         // Deklarasi & inisialisasi batas atas/guard dari loop
54         int upper = lower + length;
55
56         // Selama batas bawah masih belum 'bertemu' batas atas, lanjutkan iterasi
57         for (; lower < upper; lower++){
58             // Bilangan yang di-input
59             long number = numbers[lower];
60
61             // Baca semua isi array yang sudah pernah ditulisi
62             // Jaga loop untuk tidak membaca elemen yang belum ditulisi
63             for (int i = 0, j = idx; i < j; i++) {
64                 // Elemen berikut = -Bilangan yang diinput * isi array
65                 // Siaga 1 : Arithmetic Overflow!
66                 long element = -number * ar[i];
67
68                 // Tulis elemen tersebut ke dalam array
```

```
69         // Naikkan index
70         ar[idx++] = element;
71
72         // Hitung jumlah
73         // Tambahkan jumlah tersebut ke dalam hasil akhir
74         result += calculate(limit, element);
75     }
76
77     // Default : tulis bilangan yang di-input oleh user ke dalam array
78     // Naikkan index
79     ar[idx++] = number;
80
81     // Hitung jumlah
82     // Tambahkan jumlah tersebut ke dalam hasil akhir
83     result += calculate(limit, number);
84 }
85
86     return result;
87 }
88
89 /**
90  * Fungsi untuk menghitung jumlah semua bilangan kelipatan 'number' dari 1
91  * sampai 'limit' (inclusive). Sebagai contoh: Jumlah bilangan kelipatan 2 dari
92  * 1 sampai 100 dinyatakan sebagai calculate(2, 100) = 2550
93  *
94  * Bagian terkeren dari fungsi ini : mampu menerima 'number' positif ataupun
95  * negatif! xixixi
96  *
97  * @param limit batas maksimal deret yang ingin dihitung (inclusive). Asumsi :
98  * limit adalah bilangan bulat tidak negatif
99  *
100 * @param number kelipatan bilangan yang ingin dihitung. Asumsi : number adalah
101 * bilangan bulat tidak 0
102 *
103 * @return jumlah semua bilangan kelipatan number dari 1...limit
104 */
105 public long calculate(long limit, long number){
106     assert(limit >= 0L);
107     assert(number != 0L);
108
109     // Menghitung banyaknya elemen
110     long count = limit / number;
111
112     // Banyaknya elemen selalu positif
113     if (count < 0) count = -count;
114
115     // Jika ada elemen yang dimaksud maka akan dihitung.
116     // Jika tidak ada berarti 0
117     if (count > 0)
118         //  $S(n) = 1/2 * \text{banyaknya elemen} * (\text{elemen pertama} + \text{elemen terakhir})$ 
119         // Siaga 1 : Arithmetic Overflow!
120         return number * (count + 1L) * count / 2L;
121     else
122         return 0L;
123 }
124 }
```