
```

1 package euler;
2
3 /**
4  * Kelas Logarithm untuk menghitung logaritma. Metode perhitungan menggunakan Binary
5  * Logarithm, yaitu logaritma dalam basis 2.
6  *
7  * @author Hendra Jaya
8  * @since 17 Oktober 2010
9  */
10 public class Logarithm {
11     /**
12      * Nilai lb(10) dan lb(e).
13      * Dipersiapkan untuk Common Logarithm dan Natural Logarithm.
14      * Kedua nilai dihitung menggunakan toleransi galat {10}^{-20}
15      */
16     private static final double lb_10 = 3.321928094887362D;
17     private static final double lb_e = 1.4426950408889634D;
18
19     /**
20      * Toleransi galat. Di-set ke {10}^{-20}. Cukup untuk penggunaan sehari-hari
21      */
22     private static final double err = 1E-20D;
23
24     /**
25      * Common Logarithm. Logaritma dalam basis 10
26      * @param x bilangan real positif
27      * @return log_10(x)
28      */
29     public static double log(double x){
30         return lb(x) / lb_10;
31     }
32
33     /**
34      * Logarithm. Logaritma dalam basis sesuai keinginan user
35      * @param base bilangan real positif yang akan dijadikan basis logaritma
36      * @param x bilangan real positif
37      * @return log_base(x)
38      */
39     public static double log(double base, double x){
40         return lb(x) / lb(base);
41     }
42
43     /**
44      * Natural Logarithm. Logaritma dalam basis e. Yang dimaksud dengan e adalah
45      * bilangan Euler, yaitu 2.71828182845904523536...
46      * @param x bilangan real positif
47      * @return log_e(x)
48      */
49     public static double ln(double x){
50         return lb(x) / lb_e;
51     }
52
53     /**
54      * Binary Logarithm. Logaritma dalam basis 2
55      * @param x
56      * @return log_2(x)
57      */
58     public static double lb(double x){
59         assert(x > 0D);
60
61         double result = 0D;
62
63         while (x < 1){
64             result -= 1D;
65             x *= 2D;
66         }
67
68         while (x >= 2){

```

```
69         result += 1D;
70         x /= 2D;
71     }
72
73     double frac = 1D;
74     while(frac > err){
75         frac /= 2D;
76         x *= x;
77
78         if (x >= 2D){
79             x /= 2D;
80             result += frac;
81         }
82     }
83
84     return result;
85 }
86 }
```