

```
1 import components.simplereader.SimpleReader;
2
3 /**
4  * Execution of the de Jager formula.
5  *
6  * @author Vaishnavi Kasabwala
7  */
8 public final class ABCDGuesser2 {
9
10     /**
11      * Private constructor so this utility class cannot be instantiated.
12      */
13     private ABCDGuesser2() {
14     }
15
16     /**
17      * Repeatedly asks the user for a positive real number until the user enters
18      * one. Returns the positive real number.
19      *
20      * @param in
21      *         the input stream
22      * @param out
23      *         the output stream
24      * @return a positive real number entered by the user
25      */
26     private static double getPositiveDouble(SimpleReader in, SimpleWriter out) {
27
28         out.println(
29             "Please enter a positive, decimal point number that you would like to
30             estimate.");
31         String x = in.nextLine();
32
33         while (!FormatChecker.canParseDouble(x) || Double.parseDouble(x) <= 0) {
34             out.println(
35                 "Error. Please enter a positive, decimal point number.");
36             x = in.nextLine();
37         }
38
39         return Double.parseDouble(x);
40     }
41
42     /**
43      * Repeatedly asks the user for a positive real number not equal to 1.0
44      * until the user enters one. Returns the positive real number.
45      *
46      * @param in
47      *         the input stream
48      * @param out
49      *         the output stream
50      * @return a positive real number not equal to 1.0 entered by the user
51      */
52     private static double getPositiveDoubleNotOne(SimpleReader in,
53         SimpleWriter out) {
54         String x = in.nextLine();
55         // double x = getPositiveDouble(in, out);
56     }
57
58 }
59
60
```

```

61     while (!FormatChecker.canParseDouble(x) || Double.parseDouble(x) <= 1) {
62         out.println(
63             "Error. Please enter a positive, decimal point number that is not 1.");
64         x = in.nextLine();
65     }
66
67     return Double.parseDouble(x);
68 }
69
70 /**
71  * Evaluates the percent error.
72  */
73 private static void error(SimpleWriter out, double mu,
74     double approx) {
75     double error = Math.abs((mu - approx) / mu) * 100;
76     out.println("The estimated percent error is " + error + " .");
77 }
78
79
80 /**
81  * Main method.
82  *
83  * @param args
84  *     the command line arguments
85  */
86 public static void main(String[] args) {
87     SimpleReader in = new SimpleReader1L();
88     SimpleWriter out = new SimpleWriter1L();
89
90     /*
91     * Gets values for the variables mu, w, x, y, and z.
92     */
93
94     double mu = getPositiveDouble(in, out);
95     out.println(
96         "Please enter your first personal number. Must be a positive, decimal point
97         number not equal to zero.");
98     double w = getPositiveDoubleNotOne(in, out);
99     out.println(
100         "Please enter your second personal number. Must be a positive, decimal point
101         number not equal to zero.");
102     double x = getPositiveDoubleNotOne(in, out);
103     out.println(
104         "Please enter your third personal number. Must be a positive, decimal point
105         number not equal to zero.");
106     double y = getPositiveDoubleNotOne(in, out);
107     out.println(
108         "Please enter your fourth personal number. Must be a positive, decimal point
109         number not equal to zero.");
110     double z = getPositiveDoubleNotOne(in, out);
111
112     double epsilon = 0.01;
113     double[] arr = { -5.0, -4.0, -3.0, -2.0, -1.0, -1.0 / 2.0, -1.0 / 3.0,
114         -1.0 / 4.0, 0, 1.0 / 4.0, 1.0 / 3.0, 1.0 / 2.0, 1.0, 2.0, 3.0,
115         4.0, 5.0 };
116     double a, b, c, d;
117     double aVal, bVal, cVal, dVal;

```

```
114     double temp = 0, approx = 0;
115     double[] save = new double[4];
116
117     for (int i = 0; i < arr.length; i++) {
118         a = arr[i];
119         aVal = Math.pow(w, a);
120         for (int j = 0; j < arr.length; j++) {
121             b = arr[j];
122             bVal = Math.pow(w, b);
123             for (int k = 0; k < arr.length; k++) {
124                 c = arr[k];
125                 cVal = Math.pow(w, c);
126                 for (int l = 0; l < arr.length; l++) {
127                     d = arr[l];
128                     dVal = Math.pow(w, d);
129                     temp = (aVal * bVal * cVal * dVal);
130                     if (Math.abs(mu - temp) < Math.abs(mu - approx)) {
131                         approx = mu - temp;
132                         save[0] = a;
133                         save[1] = b;
134                         save[2] = c;
135                         save[3] = d;
136                     }
137                 }
138             }
139         }
140     }
141
142     out.println("The estimated value is " + approx + " .");
143     out.println("The exponent for w is " + save[0] + " .");
144     out.println("The exponent for x is " + save[1] + " .");
145     out.println("The exponent for y is " + save[2] + " .");
146     out.println("The exponent for z is " + save[3] + " .");
147     error(out, mu, approx);
148
149     /*
150     * Close input and output streams
151     */
152     in.close();
153     out.close();
154 }
155 }
156
```