```java
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;
import components.utilities.FormatChecker;

/**
 * Execution of the de Jager formula.
 *
 * @author Vaishnavi Kasabwala
 *
 */
public final class ABCDGuesser1 {

    /**
     * Private constructor so this utility class cannot be instantiated.
     */
    private ABCDGuesser1() {
    }

    /**
     * Repeatedly asks the user for a positive real number until the user enters
     * one. Returns the positive real number.
     *
     * @param in
     *            the input stream
     * @param out
     *            the output stream
     * @return a positive real number entered by the user
     */
    private static double getPositiveDouble(SimpleReader in, SimpleWriter out) {

        out.println(
                "Please enter a positive, decimal point number that you would like to estimate.");
        String x = in.nextLine();

        while (!FormatChecker.canParseDouble(x) || Double.parseDouble(x) <= 0) {
            out.println(
                    "Error. Please enter a positive, decimal point number.");
            x = in.nextLine();
        }

        return Double.parseDouble(x);
    }

    /**
     * Repeatedly asks the user for a positive real number not equal to 1.0
     * until the user enters one. Returns the positive real number.
     *
     * @param in
     *            the input stream
     * @param out
     *            the output stream
     * @return a positive real number not equal to 1.0 entered by the user
     */
    private static double getPositiveDoubleNotOne(SimpleReader in,
```

```
57              SimpleWriter out) {
58          String x = in.nextLine();
59          // double x = getPositiveDouble(in, out);
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       * .
72       */
73      private static void error() {
74
75      }
76
77      /**
78       * Main method.
79       *
80       * @param args
81       *             the command line arguments
82       */
83      public static void main(String[] args) {
84          SimpleReader in = new SimpleReader1L();
85          SimpleWriter out = new SimpleWriter1L();
86
87          /*
88           * Gets values for the variables mu, w, x, y, and z.
89           */
90
91          double mu = getPositiveDouble(in, out);
92          out.println(
93                  "Please enter your first personal number. Must be a positive, decimal point
    number not equal to zero.");
94          double w = getPositiveDoubleNotOne(in, out);
95          out.println(
96                  "Please enter your second personal number. Must be a positive, decimal point
    number not equal to zero.");
97          double x = getPositiveDoubleNotOne(in, out);
98          out.println(
99                  "Please enter your third personal number. Must be a positive, decimal point
    number not equal to zero.");
100         double y = getPositiveDoubleNotOne(in, out);
101         out.println(
102                 "Please enter your fourth personal number. Must be a positive, decimal point
    number not equal to zero.");
103         double z = getPositiveDoubleNotOne(in, out);
104
105         double epsilon = 0.01;
106         double[] arr = { -5.0, -4.0, -3.0, -2.0, -1.0, -1.0 / 2.0, -1.0 / 3.0,
107                 -1.0 / 4.0, 0, 1.0 / 4.0, 1.0 / 3.0, 1.0 / 2.0, 1.0, 2.0, 3.0,
108                 4.0, 5.0 };
109         double a, b, c, d;
```

```
110         double aVal, bVal, cVal, dVal;
111         double temp = 0, approx = 0;
112         double[] save = new double[4];
113 
114         int i = 0, j = 0, k = 0, l = 0;
115         while (i < arr.length) {
116             a = arr[i];
117             aVal = Math.pow(w, a);
118             while (j < arr.length) {
119                 b = arr[j];
120                 bVal = Math.pow(w, b);
121                 while (k < arr.length) {
122                     c = arr[k];
123                     cVal = Math.pow(w, c);
124                     while (l < arr.length) {
125                         d = arr[l];
126                         dVal = Math.pow(w, d);
127                         temp = (aVal * bVal * cVal * dVal);
128                         if (Math.abs(mu - temp) < Math.abs(mu - approx)) {
129                             approx = mu - temp;
130                             save[0] = a;
131                             save[1] = b;
132                             save[2] = c;
133                             save[3] = d;
134                         }
135                         l++;
136                     }
137                     k++;
138                 }
139                 j++;
140             }
141             i++;
142         }
143 
144         out.println("The estimated value is " + approx + " .");
145         out.println("The exponent for w is " + save[0] + " .");
146         out.println("The exponent for x is " + save[1] + " .");
147         out.println("The exponent for y is " + save[2] + " .");
148         out.println("The exponent for z is " + save[3] + " .");
149 
150         double error = Math.abs((mu - approx) / mu) * 100;
151         out.println("The estimated percent error is " + error + " .");
152 
153         /*
154          * Close input and output streams
155          */
156         in.close();
157         out.close();
158     }
159 }
160 
```