```java
import components.simplereader.SimpleReader;

/**
 * CSE 2221 Project #3.
 *
 * @author Faye Leigh
 */
public final class ABCDGuesser1 {

    /**
     * No argument constructor--private to prevent instantiation.
     */
    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) {
        double output = 0.0;
        boolean flag = true;
        String input;

        while (flag) {
            out.print("Please enter a positive number: ");
            input = in.nextLine();

            /*
             * Checks that input contains a number and is positive
             */
            if (FormatChecker.canParseDouble(input)) {
                output = Double.parseDouble(input);
                if (output > 0) {
                    flag = false;
                } else {
                    out.println("Number was not positive.");
                }
            } else {
                out.println("Input was not an number.");
            }
        }
        return output;
    }

    /**
     * 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,
            SimpleWriter out) {
```

```
 68         double output = 0.0;
 69         boolean flag = true;
 70         String input;
 71
 72         while (flag) {
 73             out.print("Please enter a number greater than 1.0: ");
 74             input = in.nextLine();
 75
 76             /*
 77              * Checks that input contains a number and is greater than 1.0
 78              */
 79             if (FormatChecker.canParseDouble(input)) {
 80                 output = Double.parseDouble(input);
 81                 if (output > 1.0) {
 82                     flag = false;
 83                 } else {
 84                     out.println("Number was not greater than 1.0");
 85                 }
 86             } else {
 87                 out.println("Input was not an number.");
 88             }
 89         }
 90         return output;
 91
 92     }
 93
 94     /**
 95      * Main method.
 96      *
 97      * @param args
 98      *            the command line arguments
 99      */
100     public static void main(String[] args) {
101         SimpleReader in = new SimpleReader1L();
102         SimpleWriter out = new SimpleWriter1L();
103         final double[] deJagerNum = { -5.0, -4.0, -3.0, -2.0, -1.0, -0.5,
104                 -1.0 / 3.0, -0.25, 0, 0.25, 1.0 / 3.0, 0.5, 1.0, 2.0, 3.0, 4.0,
105                 5.0 };
106         final double toPercent = 100;
107         final int size = deJagerNum.length;
108         double a = 0, b = 0, c = 0, d = 0, w = 0, x = 0, y = 0, z = 0, mu = 0,
109                 approximate = 0, bestApproximate = 0, eps = 1.0;
110         int i = 0, j = 0, k = 0, l = 0;
111
112         /*
113          * Asks user for a positive number and 4 more numbers greater than 1
114          */
115         out.println(
116                 "Choose a physical or mathematical constant you wish to
    approximate.");
117         mu = getPositiveDouble(in, out);
118         out.println(
119                 "Enter 4 numbers greater than 1.0 that have some personal meaning.");
120         out.println("First number (w)");
121         w = getPositiveDoubleNotOne(in, out);
122         out.println("Second number (x)");
123         x = getPositiveDoubleNotOne(in, out);
124         out.println("Third number (y)");
125         y = getPositiveDoubleNotOne(in, out);
126         out.println("Fourth number (z)");
127         z = getPositiveDoubleNotOne(in, out);
128
129         while (i < size) {
```

```
130             while (j < size) {
131                 while (k < size) {
132                     while (l < size) {
133                         /*
134                          * Approximates mu with all possible combinations of the
135                          * 17 de Jager exponents
136                          */
137                         approximate = Math.pow(w, deJagerNum[i])
138                                 * Math.pow(x, deJagerNum[j])
139                                 * Math.pow(y, deJagerNum[k])
140                                 * Math.pow(z, deJagerNum[l]);
141                         /*
142                          * Tests for lowest relative error for each approximate.
143                          * Saves approximate, error, and exponents if lowest
144                          * error is found
145                          */
146                         if (Math.abs(approximate - mu) / mu < eps) {
147                             eps = Math.abs(approximate - mu) / mu;
148                             bestApproximate = approximate;
149                             a = deJagerNum[i];
150                             b = deJagerNum[j];
151                             c = deJagerNum[k];
152                             d = deJagerNum[l];
153                         }
154                         l++;
155                     }
156                     k++;
157                     l = 0;
158                 }
159                 j++;
160                 k = 0;
161                 l = 0;
162             }
163             i++;
164             j = 0;
165             k = 0;
166             l = 0;
167         }
168 
169         out.println();
170         out.println("Constant: " + mu);
171         out.println("Best approximate: " + bestApproximate);
172         out.print("Relative error: ");
173         out.print(eps * toPercent, 2, false);
174         out.println("%");
175         out.println("Formula: (w^a)(x^b)(y^c)(z^d)");
176         out.println("w: " + w + ", x: " + x + ", y: " + y + ", z: " + z);
177         out.println("a: " + a + ", b: " + b + ", c: " + c + ", d: " + d);
178 
179         in.close();
180         out.close();
181     }
182 
183 }
184 
```