```java
import components.simplereader.SimpleReader;

/**
 * Put a short phrase describing the program here.
 *
 * @author Jeff Bonner
 *
 */
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) {

        double num;

        while (true) {
            out.print("Enter a positive double: ");
            String input = in.nextLine();

            if (FormatChecker.canParseDouble(input)) {
                num = Double.parseDouble(input);
            } else {
                continue;
            }

            if (num > 0) {
                return num;
```

```
47              }
48          }
49      }
50
51      /**
52       * Repeatedly asks the user for a positive real number not
  equal to 1.0
53       * until the user enters one. Returns the positive real number.
54       *
55       * @param in
56       *             the input stream
57       * @param out
58       *             the output stream
59       * @return a positive real number not equal to 1.0 entered by
  the user
60       */
61      private static double getPositiveDoubleNotOne(SimpleReader in,
62              SimpleWriter out) {
63
64          double num;
65
66          while (true) {
67              out.print("Enter a positive double not equal to 1.0:
  ");
68              String input = in.nextLine();
69
70              if (FormatChecker.canParseDouble(input)) {
71                  num = Double.parseDouble(input);
72              } else {
73                  continue;
74              }
75
76              if (num > 0 && Double.compare(num, 1.0) != 0) {
77                  return num;
78              }
79          }
80      }
81
82      /**
83       * Main method.
84       *
85       * @param args
86       *             the command line arguments
87       */
88      public static void main(String[] args) {
```

```
89          SimpleReader in = new SimpleReader1L();
90          SimpleWriter out = new SimpleWriter1L();
91
92          out.println("Choose a Constant Mu");
93          double mu = getPositiveDouble(in, out);
94
95          out.println("Choose a Personal Number W");
96          double w = getPositiveDoubleNotOne(in, out);
97
98          out.println("Choose a Personal Number X");
99          double x = getPositiveDoubleNotOne(in, out);
100
101         out.println("Choose a Personal NumberY");
102         double y = getPositiveDoubleNotOne(in, out);
103
104         out.println("Choose a Personal Number Z");
105         double z = getPositiveDoubleNotOne(in, out);
106
107         final double[] exponents = { -5, -4, -3, -2, -1, -1.0 / 2,
    -1.0 / 3,
108                 -1.0 / 4, 0, 1.0 / 4, 1.0 / 3, 1.0 / 2, 1, 2, 3, 4,
    5 };
109         double difference = 0;
110         double minDifference = Math.pow(w, exponents[0])
111                 * Math.pow(x, exponents[0]) * Math.pow(y,
    exponents[0])
112                 * Math.pow(z, exponents[0]) - mu;
113         double a = 0, b = 0, c = 0, d = 0;
114         double bestA = 0, bestB = 0, bestC = 0, bestD = 0;
115         double jager = 0;
116         double bestJager = 0;
117
118         int i = 0, j = 0, k = 0, l = 0;
119         while (i < exponents.length) {
120             while (j < exponents.length) {
121                 while (k < exponents.length) {
122                     while (l < exponents.length) {
123                         a = exponents[i];
124                         b = exponents[j];
125                         c = exponents[k];
126                         d = exponents[l];
127
128                         jager = Math.pow(w, a) * Math.pow(x, b) *
    Math.pow(y, c)
129                                 * Math.pow(z, d);
```

```
130
131                         difference = jager - mu;
132
133                         if (Math.abs(difference) <
    Math.abs(minDifference)) {
134                             minDifference = difference;
135                             bestA = a;
136                             bestB = b;
137                             bestC = c;
138                             bestD = d;
139                             bestJager = jager;
140
141                         }
142                         l++;
143
144                     }
145                     k++;
146                     l = 0;
147                 }
148                 j++;
149                 k = 0;
150
151             }
152             i++;
153             j = 0;
154         }
155
156         final double relError = minDifference / mu * 100;
157         out.println("Best values:");
158         out.println("A (exponent for W) = " + bestA);
159         out.println("B (exponent for X) = " + bestB);
160         out.println("C (exponent for Y) = " + bestC);
161         out.println("D (exponent for Z) = " + bestD);
162
163         out.println("Best de Jager value: ");
164         out.print(bestJager, 2, false);
165         out.println("");
166
167         out.println("Relative error:");
168         out.print(relError, 2, false);
169         out.println("%");
170         /*
171          * Close input and output streams
172          */
173         in.close();
```

```
174         out.close();
175     }
176
177 }
178
```