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

/**
 * A program that takes four personal numbers from a user as well as a
 * mathematical constant and approximates the constant using the de Jager
 * formula
 *
 * @author VishalKumar
 *
 */
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 = 0;
        // ask user for input
        out.print("Enter a positive real number: ");
        String input = in.nextLine();

        // verify that input is positive and real
        while (!(FormatChecker.canParseDouble(input))) {
            out.print("Please enter a positive real number: ");
            input = in.nextLine();
        }
        while (Double.parseDouble(input) <= 0) {
            out.print("Please enter a POSITIVE real number: ");
            input = in.nextLine();
        }

        //convert input to double and return value
        num = Double.parseDouble(input);
        return num;

    }

    /**
     * Repeatedly asks the user for a positive real number not equal to 1.0
     * until the user enters one. Returns the positive real number.
```

```
58       *
59       * @param in
60       *            the input stream
61       * @param out
62       *            the output stream
63       * @return a positive real number not equal to 1.0 entered by the user
64       */
65      private static double getPositiveDoubleNotOne(SimpleReader in,
66              SimpleWriter out) {
67          double num = 0;
68          // ask user for input
69          out.print("Enter a positive real number not equal to 1.0: ");
70          String input = in.nextLine();
71
72          // verify that input is positive and real and not equal to 1.0
73          while (!(FormatChecker.canParseDouble(input))) {
74              out.print("Please enter a positive real number not equal to 1.0: ");
75              input = in.nextLine();
76          }
77          while (Double.parseDouble(input) <= 0
78                  || Math.abs(Double.parseDouble(input) - 1) < .0001) {
79              out.print("Please enter a POSITIVE real number NOT equal to 1.0: ");
80              input = in.nextLine();
81          }
82
83          //convert input to double and return value
84          num = Double.parseDouble(input);
85          return num;
86
87      }
88
89      /**
90       * Main method.
91       *
92       * @param args
93       *            the command line arguments
94       */
95      public static void main(String[] args) {
96          SimpleReader in = new SimpleReader1L();
97          SimpleWriter out = new SimpleWriter1L();
98
99          // get constant from the user
100         out.println(
101                 "Enter a mathmatical constant that you want to approximate");
102         double constant = getPositiveDouble(in, out);
103
104         // get four numbers from the user
105         out.println("\nEnter four numbers");
106         double w = getPositiveDoubleNotOne(in, out);
107         double x = getPositiveDoubleNotOne(in, out);
108         double y = getPositiveDoubleNotOne(in, out);
109         double z = getPositiveDoubleNotOne(in, out);
110
111         // array of 17 charming theory numbers
112         double[] charmNums = { -5, -4, -3, -2, -1, -1.0 / 2, -1.0 / 3, -1.0 / 4,
113                 0, 1.0 / 4, 1.0 / 3, 1.0 / 2, 1, 2, 3, 4, 5 };
114         int length = charmNums.length;
```

```
115
116         // calculate the the difference between initial approximate and constant
117         double difference = Math.abs(((w * charmNums[0]) * (x * charmNums[0])
118                 * (y * charmNums[0]) * (z * charmNums[0]) - constant));
119
120         // initialize exponent indexes, exponent values of charming theory
121         int a = 0, b = 0, c = 0, d = 0;
122         double aVal = 0, bVal = 0, cVal = 0, dVal = 0;
123
124         // loop until end of the charmNums array is reached
125         while (d < length) {
126             while (c < length) {
127                 while (b < length) {
128                     while (a < length) {
129
130                         // calculate how far off new difference is from the constant
131                         double currentDiff = Math
132                                 .abs((Math.pow(w, charmNums[a]))
133                                         * (Math.pow(x, charmNums[b]))
134                                         * (Math.pow(y, charmNums[c]))
135                                         * (Math.pow(z, charmNums[d]))
136                                         - constant);
137
138                         // assign new values if current diff is closer than difference
139                         if (currentDiff < difference) {
140                             difference = currentDiff;
141                             aVal = charmNums[a];
142                             bVal = charmNums[b];
143                             cVal = charmNums[c];
144                             dVal = charmNums[d];
145                         }
146                         a++;
147                     }
148                     b++;
149                     a = 0;
150                 }
151                 c++;
152                 b = 0;
153             }
154             d++;
155             c = 0;
156         }
157
158         // calculate error and print final results
159         double error = (difference / constant) * 100;
160
161         out.println("\nThe exponent of " + w + " is " + aVal);
162         out.println("The exponent of " + x + " is " + bVal);
163         out.println("The exponent of " + y + " is " + cVal);
164         out.println("The exponent of " + z + " is " + dVal);
165         out.println("Using the charming theory the approximate value is: "
166                 + (Math.pow(w, aVal)) * (Math.pow(x, bVal))
167                         * (Math.pow(y, cVal)) * (Math.pow(z, dVal)));
168         out.print("The error is: ");
169         out.print(error, 2, false);
170         out.print("%");
171
```

```
172         // close input and output streams
173         in.close();
174         out.close();
175     }
176 
177 }
178 
```