```java
import components.simplereader.SimpleReader;

/**
 * A program that takes four personal numbers from a user as well as a
 * mathematical constant and approximates the constant using the deJager formula
 *
 * @author VishalKumar
 *
 */
public final class ABCDGuesser2 {

    /**
     * Private constructor so this utility class cannot be instantiated.
     */
    private ABCDGuesser2() {
    }

    /**
     * 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.
     *
     * @param in
     *            the input stream
     * @param out
     *            the output stream
```

```
62      * @return a positive real number not equal to 1.0 entered by the user
63      */
64     private static double getPositiveDoubleNotOne(SimpleReader in,
65             SimpleWriter out) {
66         double num = 0;
67         // ask user for input
68         out.print("Enter a positive real number not equal to 1.0: ");
69         String input = in.nextLine();
70
71         // verify that input is positive and real and not equal to 1.0
72         while (!(FormatChecker.canParseDouble(input))) {
73             out.print("Please enter a positive real number not equal to 1.0: ");
74             input = in.nextLine();
75         }
76         while (Double.parseDouble(input) <= 0
77                 || Math.abs(Double.parseDouble(input) - 1) < .0001) {
78             out.print("Please enter a POSITIVE real number NOT equal to 1.0: ");
79             input = in.nextLine();
80         }
81
82         //convert input to double and return value
83         num = Double.parseDouble(input);
84         return num;
85
86     }
87
88     private static void printFinalResults(double w, double x, double y,
89             double z, double aVal, double bVal, double cVal, double dVal,
90             double error, SimpleWriter out, SimpleReader in) {
91         out.println("\nThe exponent of " + w + " is " + aVal);
92         out.println("The exponent of " + x + " is " + bVal);
93         out.println("The exponent of " + y + " is " + cVal);
94         out.println("The exponent of " + z + " is " + dVal);
95         out.println("Using the charming theory the approximate value is: "
96                 + (Math.pow(w, aVal)) * (Math.pow(x, bVal))
97                         * (Math.pow(y, cVal)) * (Math.pow(z, dVal)));
98         out.print("The error is: ");
99         out.print(error, 2, false);
100        out.print("%");
101    }
102
103    /**
104     * Main method.
105     *
106     * @param args
107     *            the command line arguments
108     */
109    public static void main(String[] args) {
110        SimpleReader in = new SimpleReader1L();
111        SimpleWriter out = new SimpleWriter1L();
112
113        // get constant from the user
114        out.println(
115                "Enter a mathmatical constant that you want to approximate");
116        double constant = getPositiveDouble(in, out);
117
118        // get four numbers from the user
```

```
119         out.println("\nEnter four numbers");
120         double w = getPositiveDoubleNotOne(in, out);
121         double x = getPositiveDoubleNotOne(in, out);
122         double y = getPositiveDoubleNotOne(in, out);
123         double z = getPositiveDoubleNotOne(in, out);
124
125         // array of 17 charming theory numbers
126         double[] charmNums = { -5, -4, -3, -2, -1, -1.0 / 2, -1.0 / 3, -1.0 / 4,
127                 0, 1.0 / 4, 1.0 / 3, 1.0 / 2, 1, 2, 3, 4, 5 };
128         int length = charmNums.length;
129
130         // calculate the the difference between initial approximate and constant
131         double difference = Math.abs(((w * charmNums[0]) * (x * charmNums[0])
132                 * (y * charmNums[0]) * (z * charmNums[0]) - constant));
133
134         // initialize exponent indexes, exponent values of charming theory
135         double aVal = 0, bVal = 0, cVal = 0, dVal = 0;
136
137         // loop until end of the charmNums array is reached
138         for (int d = 0; d < length; d++) {
139             for (int c = 0; c < length; c++) {
140                 for (int b = 0; b < length; b++) {
141                     for (int a = 0; a < length; a++) {
142
143                         // calculate how far off new difference is from the constant
144                         double currentDiff = Math
145                                 .abs((Math.pow(w, charmNums[a]))
146                                         * (Math.pow(x, charmNums[b]))
147                                         * (Math.pow(y, charmNums[c]))
148                                         * (Math.pow(z, charmNums[d]))
149                                         - constant);
150
151                         // assign new values if current diff is closer than difference
152                         if (currentDiff < difference) {
153                             difference = currentDiff;
154                             aVal = charmNums[a];
155                             bVal = charmNums[b];
156                             cVal = charmNums[c];
157                             dVal = charmNums[d];
158                         }
159                     }
160                 }
161             }
162         }
163
164         // calculate error and print final results
165         double error = (difference / constant) * 100;
166         printFinalResults(w, x, y, z, aVal, bVal, cVal, dVal, error, out, in);
167
168         // close input and output streams
169         in.close();
170         out.close();
171     }
172
173 }
174
```