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

/**
 * Asks the user to input a mathematical constant, and which every 4 numbers
 * they wish to input. Then uses the de Jager formula to calculate an
 * approximation of the constant using the 4 numbers the user inputted. Also
 * calculated the percent error between the approximation and the actual
 * constant.
 *
 * @author Shyam Sai Bethina
 *
 */

public final class ABCDGuesser1 {

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

    /**
     * Main method.
     *
     * @param args
     *            the command line arguments
     */
    public static void main(String[] args) {
        SimpleReader in = new SimpleReader1L();
        SimpleWriter out = new SimpleWriter1L();
        double[] abcd = new double[] { -5, -4, -3, -2, -1, -(1 / (double) 2),
                -(1 / (double) 3), -(1 / (double) 4), 0, 1 / (double) 4,
                1 / (double) 3, 1 / (double) 2, 1, 2, 3, 4, 5 };
```

```
38
39          final double hundred = 100;
40
41          out.println("Input a constant to be approximated: ");
42          double constant = getPositiveDouble(in, out);
43
44          out.println("Input the first positive real number not
    equal to 1.0: ");
45          double w = getPositiveDoubleNotOne(in, out);
46
47          out.println("Input the second positive real number not
    equal to 1.0: ");
48          double x = getPositiveDoubleNotOne(in, out);
49
50          out.println("Input the third positive real number not
    equal to 1.0: ");
51          double y = getPositiveDoubleNotOne(in, out);
52
53          out.println("Input the fourth positive real number not
    equal to 1.0: ");
54          double z = getPositiveDoubleNotOne(in, out);
55
56          int aCounter = 0;
57          int bCounter = 0;
58          int cCounter = 0;
59          int dCounter = 0;
60
61          double a = abcd[aCounter];
62          double b = abcd[bCounter];
63          double c = abcd[cCounter];
64          double d = abcd[cCounter];
65
66          double constantApprox = Math.pow(w, a) * Math.pow(x, b) *
    Math.pow(y, c)
67                  * Math.pow(z, d);
68          double error = (Math.abs(constantApprox - constant) /
    constant)
69                  * hundred;
70          double finalApprox = constantApprox;
71          double finalError = error;
72
73          while (aCounter < abcd.length) {
74              a = abcd[aCounter];
75              bCounter = 0;
```

```
 76
 77              while (bCounter < abcd.length) {
 78                  b = abcd[bCounter];
 79                  cCounter = 0;
 80
 81                  while (cCounter < abcd.length) {
 82                      c = abcd[cCounter];
 83                      dCounter = 0;
 84
 85                      while (dCounter < abcd.length) {
 86                          d = abcd[dCounter];
 87
 88                          constantApprox = Math.pow(w, a) *
    Math.pow(x, b)
 89                                  * Math.pow(y, c) * Math.pow(z, d);
 90                          error = (Math.abs(constantApprox -
    constant) / constant)
 91                                  * hundred;
 92
 93                          if (error < finalError) {
 94                              finalApprox = constantApprox;
 95                              finalError = error;
 96
 97                          }
 98                          dCounter++;
 99                      }
100                      cCounter++;
101                  }
102                  bCounter++;
103              }
104              aCounter++;
105
106          }
107
108          out.println("Final approximation of constant is " +
    finalApprox);
109          out.print("Final error is: ");
110          out.print(finalError, 2, false);
111
112      }
113
114      /**
115       * Repeatedly asks the user for a positive real number until
    the user enters
```

```
116      * one. Returns the positive real number.
117      *
118      * @param in
119      *            the input stream
120      * @param out
121      *            the output stream
122      * @return a positive real number entered by the user
123      */
124     private static double getPositiveDouble(SimpleReader in,
    SimpleWriter out) {
125         String userInput = in.nextLine();
126         while (!FormatChecker.canParseDouble(userInput)) {
127             out.println("Input a positive real number: ");
128             userInput = in.nextLine();
129         }
130
131         return Double.parseDouble(userInput);
132     }
133
134     /**
135      * Repeatedly asks the user for a positive real number not
    equal to 1.0
136      * until the user enters one. Returns the positive real
    number.
137      *
138      * @param in
139      *            the input stream
140      * @param out
141      *            the output stream
142      * @return a positive real number not equal to 1.0 entered by
    the user
143      */
144     private static double getPositiveDoubleNotOne(SimpleReader in,
145             SimpleWriter out) {
146         String userInput = in.nextLine();
147         final double one = 1.0;
148         while (!FormatChecker.canParseDouble(userInput)) {
149             out.println("Input a real number not equal to 1.0");
150             userInput = in.nextLine();
151         }
152
153         while (Double.parseDouble(userInput) == one) {
154             out.println("Input a real number not equal to 1.0");
155             userInput = in.nextLine();
```

```
156         }
157
158         return Double.parseDouble(userInput);
159     }
160 }
161
```