

```
1 import components.simplereader.SimpleReader;
2 import components.simplereader.SimpleReader1L;
3 import components.simplewriter.SimpleWriter;
4 import components.simplewriter.SimpleWriter1L;
5 import components.utilities.FormatChecker;
6
7 /**
8  * Asks the user to input a mathematical constant, and which every
9  * 4 numbers
10 * they wish to input. Then uses the de Jager formula to calculate
11 * an
12 * approximation of the constant using the 4 numbers the user
13 * inputted. Also
14 * calculated the percent error between the approximation and the
15 * actual
16 * constant.
17 *
18 * @author Shyam Sai Bethina
19 */
20 public final class ABCDGuesser2 {
21
22     /**
23      * Private constructor so this utility class cannot be
24      * instantiated.
25      */
26     private ABCDGuesser2() {
27     }
28
29     /**
30      * Main method.
31      *
32      * @param args
33      *         the command line arguments
34      */
35     public static void main(String[] args) {
36         SimpleReader in = new SimpleReader1L();
37         SimpleWriter out = new SimpleWriter1L();
38         double[] abcd = new double[] { -5, -4, -3, -2, -1, -(1 /
39 (double) 2),
40 -(1 / (double) 3), -(1 / (double) 4), 0, 1 /
41 (double) 4,
42 1 / (double) 3, 1 / (double) 2, 1, 2, 3, 4, 5 };
43     }
```

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

```

75         c = abcd[cCounter];
76         for (dCounter = 0; dCounter < abcd.length;
77             dCounter++) {
78             d = abcd[dCounter];
79             constantApprox = Math.pow(w, a) *
Math.pow(x, b)
80                 * Math.pow(y, c) * Math.pow(z, d);
81             error = error(constantApprox, constant);
82
83             if (error < finalError) {
84                 finalApprox = constantApprox;
85                 finalError = error;
86
87             }
88         }
89     }
90 }
91 }
92
93     out.println("Final approximation of constant is " +
finalApprox);
94     out.print("Final error is: ");
95     out.print(finalError, 2, false);
96
97 }
98
99 /**
100  * Repeatedly asks the user for a positive real number until
the user enters
101  * one. Returns the positive real number.
102  *
103  * @param in
104  *         the input stream
105  * @param out
106  *         the output stream
107  * @return a positive real number entered by the user
108  */
109 private static double getPositiveDouble(SimpleReader in,
SimpleWriter out) {
110     String userInput = in.nextLine();
111     while (!FormatChecker.canParseDouble(userInput)) {
112         out.println("Input a positive real number: ");
113         userInput = in.nextLine();

```

```
114     }
115
116     return Double.parseDouble(userInput);
117 }
118
119 /**
120  * Repeatedly asks the user for a positive real number not
    equal to 1.0
121  * until the user enters one. Returns the positive real
    number.
122  *
123  * @param in
124  *         the input stream
125  * @param out
126  *         the output stream
127  * @return a positive real number not equal to 1.0 entered by
    the user
128  */
129 private static double getPositiveDoubleNotOne(SimpleReader in,
130     SimpleWriter out) {
131     String userInput = in.nextLine();
132     final double one = 1.0;
133     while (!FormatChecker.canParseDouble(userInput)) {
134         out.println("Input a real number not equal to 1.0");
135         userInput = in.nextLine();
136     }
137
138     while (Double.parseDouble(userInput) == one) {
139         out.println("Input a real number not equal to 1.0");
140         userInput = in.nextLine();
141     }
142
143     return Double.parseDouble(userInput);
144 }
145
146 /**
147  * Calculates the error between the constant approximation and
    the constant.
148  * Returns the resulting error.
149  *
150  * @param constantApprox
151  *         the approximation of the constant
152  * @param constant
153  *         the original constant the program is
```

```
    approximating to
154    * @return an percentage of the error based on the original
    constant
155    */
156    private static double error(double constantApprox, double
    constant) {
157        final double hundred = 100;
158        double error = (Math.abs(constantApprox - constant) /
    constant)
159            * hundred;
160
161        return error;
162    }
163 }
164
```