

```
1 import components.simplereader.SimpleReader;
6
7 /**
8  * CSE 2221 Project #3.
9  *
10 * @author Faye Leigh
11 */
12 public final class ABCDGuesser2 {
13
14     /**
15      * No argument constructor--private to prevent instantiation.
16      */
17     private ABCDGuesser2() {
18     }
19
20     /**
21      * Repeatedly asks the user for a positive real number until the user enters
22      * one. Returns the positive real number.
23      *
24      * @param in
25      *         the input stream
26      * @param out
27      *         the output stream
28      * @return a positive real number entered by the user
29      */
30     private static double getPositiveDouble(SimpleReader in, SimpleWriter out) {
31         double output = 0.0;
32         boolean flag = true;
33         String input;
34
35         /*
36          * Run until user input satisfies conditions
37          */
38         while (flag) {
39             out.print("Please enter a positive number: ");
40             input = in.nextLine();
41
42             /*
43              * Checks that input contains a number and is positive
44              */
45             if (FormatChecker.canParseDouble(input)) {
46                 output = Double.parseDouble(input);
47                 if (output > 0) {
48                     flag = false;
49                 } else {
50                     out.println("Number was not positive.");
51                 }
52             } else {
53                 out.println("Input was not an number.");
54             }
55         }
56         return output;
57     }
58
59     /**
60      * Repeatedly asks the user for a positive real number not equal to 1.0
61      * until the user enters one. Returns the positive real number.
62      *
63      * @param in
64      *         the input stream
65      * @param out
66      *         the output stream
67      * @return a positive real number not equal to 1.0 entered by the user
```

```
68     */
69     private static double getPositiveDoubleNotOne(SimpleReader in,
70         SimpleWriter out) {
71         double output = 0.0;
72         boolean flag = true;
73         String input;
74
75         /*
76          * Run until user input satisfies conditions
77          */
78         while (flag) {
79             out.print("Please enter a number greater than 1.0: ");
80             input = in.nextLine();
81
82             /*
83              * Checks that input contains a number and is greater than 1.0
84              */
85             if (FormatChecker.canParseDouble(input)) {
86                 output = Double.parseDouble(input);
87                 if (output > 1.0) {
88                     flag = false;
89                 } else {
90                     out.println("Number was not greater than 1.0");
91                 }
92             } else {
93                 out.println("Input was not an number.");
94             }
95         }
96         return output;
97     }
98
99     /**
100     * Runs the de Jager approximation. Returns a real number.
101     *
102     * @param w
103     *         first personal number
104     * @param i
105     *         first exponent
106     * @param x
107     *         second personal number
108     * @param j
109     *         second exponent
110     * @param y
111     *         third personal number
112     * @param k
113     *         third exponent
114     * @param z
115     *         fourth personal number
116     * @param l
117     *         fourth exponent
118     * @return the real number approximation
119     */
120     private static double getApproximate(double w, double i, double x, double j,
121         double y, double k, double z, double l) {
122         return Math.pow(w, i) * Math.pow(x, j) * Math.pow(y, k)
123             * Math.pow(z, l);
124     }
125
126     /**
127     * Main method.
128     *
129     * @param args
130     *         the command line arguments
```

```

131     */
132     public static void main(String[] args) {
133         SimpleReader in = new SimpleReader1L();
134         SimpleWriter out = new SimpleWriter1L();
135         final double[] deJagerNum = { -5.0, -4.0, -3.0, -2.0, -1.0, -0.5,
136             -1.0 / 3.0, -0.25, 0, 0.25, 1.0 / 3.0, 0.5, 1.0, 2.0, 3.0, 4.0,
137             5.0 };
138         final double toPercent = 100;
139         final int size = deJagerNum.length;
140         double a = 0, b = 0, c = 0, d = 0, w = 0, x = 0, y = 0, z = 0, mu = 0,
141             approximate = 0, bestApproximate = 0, eps = 1.0;
142         int i = 0, j = 0, k = 0, l = 0;
143
144         /*
145          * Asks user for a positive number and 4 more numbers greater than 1
146          */
147         out.println(
148             "Choose a physical or mathematical constant you wish to
approximate.");
149         mu = getPositiveDouble(in, out);
150         out.println(
151             "Enter 4 numbers greater than 1.0 that have some personal meaning.");
152         out.println("First number (w)");
153         w = getPositiveDoubleNotOne(in, out);
154         out.println("Second number (x)");
155         x = getPositiveDoubleNotOne(in, out);
156         out.println("Third number (y)");
157         y = getPositiveDoubleNotOne(in, out);
158         out.println("Fourth number (z)");
159         z = getPositiveDoubleNotOne(in, out);
160
161         for (i = 0; i < size; i++) {
162             for (j = 0; j < size; j++) {
163                 for (k = 0; k < size; k++) {
164                     for (l = 0; l < size; l++) {
165                         /*
166                          * Approximates mu with all possible combinations of the
167                          * 17 de Jager exponents
168                          */
169                         approximate = getApproximate(w, deJagerNum[i], x,
170                             deJagerNum[j], y, deJagerNum[k], z,
171                             deJagerNum[l]);
172                         /*
173                          * Tests for lowest relative error for each approximate.
174                          * Saves approximate, error, and exponents if lowest
175                          * error is found
176                          */
177                         if (Math.abs(approximate - mu) / mu < eps) {
178                             eps = Math.abs(approximate - mu) / mu;
179                             bestApproximate = approximate;
180                             a = deJagerNum[i];
181                             b = deJagerNum[j];
182                             c = deJagerNum[k];
183                             d = deJagerNum[l];
184                         }
185                     }
186                 }
187             }
188         }
189
190         out.println();
191         out.println("Constant: " + mu);
192         out.println("Best approximate: " + bestApproximate);

```

```
193     out.print("Relative error: ");
194     out.print(eps * toPercent, 2, false);
195     out.println("%");
196     out.println("Formula: (w^a) (x^b) (y^c) (z^d)");
197     out.println("w: " + w + ", x: " + x + ", y: " + y + ", z: " + z);
198     out.println("a: " + a + ", b: " + b + ", c: " + c + ", d: " + d);
199
200     in.close();
201     out.close();
202 }
203
204 }
205
```