```
1  import components.simplereader.SimpleReader;
6
7  /**
8   * Put a short phrase describing the program here.
9   *
10  * @author Put your name here
11  *
12  */
13 public final class ABCDGuesser1 {
14
15     /**
16      * No argument constructor--private to prevent instantiation.
17      */
18     private ABCDGuesser1() {
19     }
20
21     /**
22      * Main method.
23      *
24      * @param args
25      *            the command line arguments
26      */
27     public static void main(String[] args) {
28         SimpleReader in = new SimpleReader1L();
29         SimpleWriter out = new SimpleWriter1L();
30         double[] exp = { -5, -4, -3, -2, -1, -0.5, -0.333, -0.25, 0, 0.25,
31                 0.333, 0.5, 1, 2, 3, 4, 5 };
32         double total = -1, closest = -999999999;
33         double bestW = -1, bestX = -1, bestY = -1, bestZ = -1;
34         int i = 0, j = 0, k = 0, l = 0;
35
36         // get u
37         double u = getPositiveDouble(in, out);
38
39         // get w x y z
40         double w = getPositiveDoubleNotOne(in, out);
41         double x = getPositiveDoubleNotOne(in, out);
42         double y = getPositiveDoubleNotOne(in, out);
43         double z = getPositiveDoubleNotOne(in, out);
44
45         // big boy loop
46         while (i <= 16) {
47             while (j <= 16) {
48                 while (k <= 16) {
49                     while (l <= 16) {
50                         // does calculations
51                         total = Math.pow(w, exp[i]);
52                         total += Math.pow(x, exp[j]);
53                         total += Math.pow(y, exp[k]);
54                         total += Math.pow(z, exp[l]);
55
56                         // if total is the closest to u so far
57                         if (Math.abs(u - total) < Math.abs(u - closest)) {
58                             closest = total;
59                             bestW = exp[i];
60                             bestX = exp[j];
61                             bestY = exp[k];
62                             bestZ = exp[l];
63                         }
```

```
64                          l++;
65                      }
66                      k++;
67                      l = 0;
68                  }
69                  j++;
70                  k = 0;
71              }
72              i++;
73              j = 0;
74          }
75
76          // do some math
77          double percentError = Math.abs((closest - u) / u * 100 );
78
79          // print results
80          System.out.println("u = " + u);
81          System.out.println("(" + w + "^" + bestW + ")" + " + " + "(" + x + "^"
82                  + bestX + ")" + " + " + "(" + y + "^" + bestY + ")" + " + "
83                  + "(" + z + "^" + bestZ + ")" + " = "
84                  + String.format("%.2f", closest));
85          System.out.println(
86                  "Percent Error: " + String.format("%.2f", percentError) + "%");
87
88          // close stuff
89          in.close();
90          out.close();
91      }
92
93      /**
94       * Repeatedly asks the user for a positive real number until the user enters
95       * one. Returns the positive real number.
96       *
97       * @param in
98       *            the input stream
99       * @param out
100      *            the output stream
101      * @return a positive real number entered by the user
102      */
103     private static double getPositiveDouble(SimpleReader in, SimpleWriter out) {
104         System.out.print("Enter a positive number: ");
105         String input = in.nextLine();
106         boolean looping = true;
107         double num = -1;
108
109         while (looping) {
110             if (FormatChecker.canParseInt(input)) {
111                 num = Integer.parseInt(input);
112                 if (num > 0) {
113                     looping = false;
114                 }
115             } else {
116                 System.out.print("Enter a positive number: ");
117                 input = in.nextLine();
118             }
119         }
120
121         return num;
122     }
```

```
123
124     /**
125      * Repeatedly asks the user for a positive real number not equal to 1.0
126      * until the user enters one. Returns the positive real number.
127      *
128      * @param in
129      *             the input stream
130      * @param out
131      *             the output stream
132      * @return a positive real number not equal to 1.0 entered by the user
133      */
134     private static double getPositiveDoubleNotOne(SimpleReader in,
135             SimpleWriter out) {
136         boolean looping = true;
137         double num = -1;
138
139         while (looping) {
140             num = getPositiveDouble(in, out);
141             if (num != 1.0) {
142                 looping = false;
143             }
144         }
145
146         return num;
147     }
148
149 }
150
```