

```
1: /**
2:  * This program provides a series of tests and sample calls to demo the usage
3:  * and features of the 'Box' class.
4:  *
5:  * @author Ravi S. Ramphal
6:  * @class CCSF CS111B
7:  * @date 2017.06.22
8:  * @version 1.0
9:  */
10:
11: public class DemoBox
12: {
13:     /**
14:      * This is a helper method used to print a line above and below a given
15:      * message (from the middle).
16:      *
17:      * param message The message that you would like to pad
18:      */
19:     static private void mid(String message)
20:     {
21:         System.out.println(message);
22:         System.out.println();
23:     }
24:
25:     /**
26:      * This is a helper method used to print a horizontal rule above a given
27:      * message.
28:      *
29:      * param message The message that you would like to prepend to
30:      */
31:     static private void pre(String message)
32:     {
33:         System.out.println();
34:         System.out.println("=====");
35:         System.out.println();
36:         mid(message);
37:     }
38:
39:     /**
40:      * This method takes an instance of 'Box' and calls '@volume' on it
41:      * providing additional information on dimensions so the user can ensure
42:      * that the correct volume is printed for the given dimensions.
43:      *
44:      * param box An instance of 'Box' that you would like to print volume for
45:      */
46:     static private void testVolume(Box box)
47:     {
48:         mid(
49:             "      * For a box of height " + box.height + ", width " + box.width +
50:             ", and depth " + box.depth + ", the volume returned is: " +
51:             box.volume()
52:         );
53:     }
54:
55:     /**
56:      * This method takes an instance of 'Box' and calls '@toString()' on it
57:      * in the context of concatenation.
58:      *
59:      * param box An instance of 'Box' that you would like to cast as a string
60:      */
61:     static private void testToString(Box box)
62:     {
63:         mid("      * Created: " + box);
64:     }
65:
66:     /**
67:      * This method takes an instance of 'Box' and calls '@show()' on it.
68:      *
```

```
69:      * param box An instance of 'Box' that you would like to call 'show' on
70:      */
71:  static private void testShow(Box box)
72:  {
73:      mid("      * Info:");
74:      box.show();
75:  }
76:
77:  /**
78:   * This method takes two instances of 'Box' and calls '@equals()' to compare
79:   * them and prints the result.
80:   *
81:   * param box1 The first instance of 'Box' that you would like to compare
82:   * param box2 The second instance of 'Box' that you would like to compare
83:   */
84:  static private void testEquals(Box box1, Box box2)
85:  {
86:      String qualifier = (box2.equals(box1)) ? "are" : "are not";
87:
88:      mid("      * " + box1 + " and " + box2 + " " + qualifier + " equal");
89:  }
90:
91:  /**
92:   * This is a testing method used to create a box with three dimensions and
93:   * test 'volume', 'toString', and 'show'.
94:   */
95:  static private void testThreeDimensions()
96:  {
97:      pre("TESTING CREATION OF BOX WITH THREE DIMENSIONS");
98:
99:      Box box = new Box(3, 6, 9);
100:
101:      testVolume(box);
102:      testToString(box);
103:      testShow(box);
104:  }
105:
106:  /**
107:   * This is a testing method used to create a cube and
108:   * test '@volume()', '@toString()', and '@show()' via helper methods.
109:   */
110:  static private void testCube()
111:  {
112:      pre("TESTING CREATION OF CUBE");
113:
114:      Box box = new Box(5);
115:
116:      testVolume(box);
117:      testToString(box);
118:      testShow(box);
119:  }
120:
121:  /**
122:   * This is a testing method used to create a box by cloning another box and
123:   * test '@volume()', '@toString()', and '@show()' via helper methods. It
124:   * also compares it to the original via '@equals()' and compares it to
125:   * another cube of different dimensions.
126:   */
127:  static private void testClone()
128:  {
129:      pre("TESTING CREATION OF A BOX BY CLONING ANOTHER BOX");
130:
131:      Box box1 = new Box(3);
132:      Box box2 = new Box(box1);
133:      Box box3 = new Box(4);
134:
135:      testEquals(box1, box2);
136:      testEquals(box2, box3);
```

```
137:         testVolume(box2);
138:         testToString(box2);
139:         testShow(box2);
140:     }
141:
142:     /**
143:      * This is a testing method used to create a box of zero dimensions using
144:      * the default constructor and then runs tests to
145:      * test '@volume()', '@toString()', and '@show()' via helper methods.
146:      */
147:     static private void testDefault()
148:     {
149:         pre("TESTING CREATION OF DEFAULT ZERO-DIMENSION BOX");
150:
151:         Box box = new Box();
152:
153:         testVolume(box);
154:         testToString(box);
155:         testShow(box);
156:     }
157:
158:     /**
159:      * This is the main function of this demo class to call the other testing
160:      * methods.
161:      */
162:     public static void main(String ... args)
163:     {
164:         testThreeDimensions();
165:         testCube();
166:         testClone();
167:         testDefault();
168:     }
169: }
```