```
 1: /**
 2:  * This class allows the user to create a box defined by height, width, and
 3:  * depth. Constructors exist to accept specific dimensions, one dimension
 4:  * (effectively creating a cube), a Box object (to create a clone), or no
 5:  * dimensions (creating a default box with dimensions of zero).
 6:  *
 7:  * @author  Ravi S. Ramphal
 8:  * @class   CCSF CS111B
 9:  * @date    2017.06.22
10:  * @version 1.0
11:  */
12:
13: public class Box
14: {
15:     /**
16:      * Holds the height of a box.
17:      */
18:     int height;
19:
20:     /**
21:      * Holds the width of a box.
22:      */
23:     int width;
24:
25:     /**
26:      * Holds the depth of a box.
27:      */
28:     int depth;
29:
30:     /**
31:      * This is the parameterized constructor to create a box from a height,
32:      * width, and depth.
33:      *
34:      * param height The height of the box
35:      * param width  The width  of the box
36:      * param depth  The depth  of the box
37:      */
38:     public Box(int height, int width, int depth)
39:     {
40:         this.height = height;
41:         this.width  = width;
42:         this.depth  = depth;
43:     }
44:
45:     /**
46:      * This is the parameterized constructor to create a cube.
47:      *
48:      * param dimension The value for the box's height, width, and depth
49:      */
50:     public Box(int dimension)
51:     {
52:         // this.height = dimension;
53:         // this.width  = dimension;
54:         // this.depth  = dimension;
55:
56:         this(dimension, dimension, dimension);
57:     }
58:
59:     /**
60:      * This is the parameterized constructor to clone a given Box.
61:      *
62:      * param box An instance of 'Box' that you would like to clone
63:      */
64:     public Box(Box box)
65:     {
66:         // this.height = box.height;
67:         // this.width  = box.width;
68:         // this.depth  = box.depth;
```

```
 69:
 70:         this(box.height, box.width, box.depth);
 71:     }
 72:
 73:     /**
 74:      * This overrides the default constructor to create a box with
 75:      * zero dimensions if nothing is passed in.
 76:      */
 77:     public Box()
 78:     {
 79:         // this.height = 0;
 80:         // this.width  = 0;
 81:         // this.depth  = 0;
 82:
 83:         this(0, 0, 0);
 84:     }
 85:
 86:     /**
 87:      * This method returns the volume of the box.
 88:      *
 89:      * @return int This returns the volume of the box.
 90:      */
 91:     public int volume()
 92:     {
 93:         return this.height * this.width * this.depth;
 94:     }
 95:
 96:     /**
 97:      * This method allows a quick way for a user to output the dimensions of
 98:      * a box.
 99:      */
100:     public void show()
101:     {
102:         System.out.println("height : " + this.height);
103:         System.out.println("width  : " + this.width);
104:         System.out.println("depth  : " + this.height);
105:     }
106:
107:     /**
108:      * This method allows a quick way for a user to output the dimensions of
109:      * a box.
110:      *
111:      * @param  box     A given box to test against
112:      * @return boolean Returns whether or not all the dimensions of a box match
113:      */
114:     public boolean equals(Box box)
115:     {
116:         return (
117:             this.height == box.height &&
118:             this.width  == box.width  &&
119:             this.depth  == box.depth
120:         );
121:     }
122:
123:     /**
124:      * This method specifies what is returned when an instance of Box is cast
125:      * to a String.
126:      */
127:     public String toString()
128:     {
129:         return (
130:             String.format(
131:                 "(instance of Box: @height = %d; @width = %d; @depth = %d)",
132:                                 this.height,   this.width,   this.depth
133:             )
134:         );
135:     }
136: }
```