```
 1: /**
 2:  * This applet generates a screensaver using Java's AWT Graphics API.
 3:  * The screensaver generates a given shape in random sizes, colors, and
 4:  * locations set against a white background. The screensaver's height and width
 5:  * is set to the screen's height and width. The shape is defined as a parameter
 6:  * passed to the applet. Possible shapes include: 'circle', 'square',
 7:  * 'rectangle', and 'line'. If the user does not pass a 'shape' param or if
 8:  * param passed does not match one of the possible shapes, the screensaver
 9:  * defaults to drawing circles.
10:  *
11:  * In the future, this applet could be improved by making sure that all shapes
12:  * are rendered within the bounds of the screen so that no shapes are truncated.
13:  *
14:  * @name    ScreenSaver (Homework Assignment 05)
15:  * @author  Ravi S. Ramphal
16:  * @class   CCSF CS111B
17:  * @date    2017.07.17
18:  * @version 1.0
19:  */
20:
21: import java.awt.*;
22: import java.applet.*;
23:
24: // <applet code="ScreenSaver" width="0" height="0">
25: //     <param name="shape" value="circle">
26: // </applet>
27:
28: public class ScreenSaver extends Applet
29: {
30:     /**
31:      * A counter that keeps track of the current number of shapes generated
32:      */
33:     private int count = 0;
34:
35:     /**
36:      * The current resolution of the target screen
37:      */
38:     private Dimension screenSize;
39:
40:     /**
41:      * The desired background color
42:      */
43:     private final Color BACKGROUND_COLOR = Color.WHITE;
44:
45:     /**
46:      * The delay (in milliseconds) between when each new shape is generated
47:      */
48:     private final int DELAY_DURATION = 500;
49:
50:     /**
51:      * The maximum number of shapes that can be generated before the screen
52:      * clears and the screensaver restarts
53:      */
54:     private final int MAX_SHAPES = 500;
55:
56:     /**
57:      * The minimum size (in pixels) of any shape generated
58:      */
59:     private final int MIN_SIZE = 20;
60:
61:     /**
62:      * The maximum size (in pixels) of any shape generated
63:      */
64:     private final int MAX_SIZE = 100;
65:
66:     /**
67:      * This method sleeps the program for the duration passed in.
68:      *
```

```
 69:      * @param milliseconds The integer number of milliseconds to delay
 70:      */
 71:     private void delay (int milliseconds)
 72:     {
 73:         try
 74:         {
 75:             Thread.sleep(milliseconds);
 76:         }
 77:         catch (InterruptedException e) {}
 78:     }
 79:
 80:     /**
 81:      * This method returns a random integer between the provided lower limit
 82:      * and upper limit.
 83:      *
 84:      * @param  a   An int representing the lower limit
 85:      * @param  b   An int representing the upper limit
 86:      * @return int A random number between the two limits
 87:      */
 88:     private int rand (int a, int b)
 89:     {
 90:         return ((int)((b - a + 1) * Math.random() + a));
 91:     }
 92:
 93:     /**
 94:      * This method returns a randomly-generated Color.
 95:      *
 96:      * @return Color A random color
 97:      */
 98:     private Color getRandomColor ()
 99:     {
100:         return (new Color(rand(0, 255), rand(0, 255), rand(0, 255)));
101:     }
102:
103:     /**
104:      * This method clears the screen by setting the foreground to the
105:      * background color. Instead of using the main Graphics instance used in the
106:      * rest of the applet, this method generates a new instance and disposes
107:      * of it at the end to enable more flexible reusability.
108:      */
109:     private void clearScreen ()
110:     {
111:         Graphics g = getGraphics();
112:         g.setColor(getBackground());
113:         g.fillRect(0, 0, getSize().width, getSize().height);
114:         g.setColor(getForeground());
115:         g.dispose();
116:     }
117:
118:     /**
119:      * This method sets the applet's size to the size of the target screen.
120:      */
121:     private void setToScreenSize ()
122:     {
123:         Toolkit toolkit = Toolkit.getDefaultToolkit();
124:         screenSize = toolkit.getScreenSize();
125:         setSize(screenSize);
126:     }
127:
128:     /**
129:      * This is the 'init' lifecycle method of the applet. It is executed once
130:      * on applet instantiation. It sets the applet size to the screen size and
131:      * sets the background color.
132:      */
133:     public void init ()
134:     {
135:         setToScreenSize();
136:         setBackground(BACKGROUND_COLOR);
```

```
137:     }
138:
139:     /**
140:      * This is the 'update' lifecycle method of the applet. It overwrites the
141:      * default 'update' method to only repain the screen WITHOUT clearing the
142:      * screen.
143:      *
144:      * @param graphics An instance of the Graphics class
145:      */
146:     public void update (Graphics graphics)
147:     {
148:         paint(graphics);
149:     }
150:
151:     /**
152:      * This method draws a circle at the provided X and Y location.
153:      *
154:      * @param graphics An instance of the Graphics class
155:      * @param startX   The X integer value of the location to draw the circle
156:      * @param startY   The Y integer value of the location to draw the circle
157:      */
158:     private void drawCircle (Graphics graphics, int startX, int startY)
159:     {
160:         int diameter = rand(MIN_SIZE, MAX_SIZE);
161:         graphics.fillOval(startX, startY, diameter, diameter);
162:     }
163:
164:     /**
165:      * This method draws a square at the provided X and Y location.
166:      *
167:      * @param graphics An instance of the Graphics class
168:      * @param startX   The X integer value of the location to draw the square
169:      * @param startY   The Y integer value of the location to draw the square
170:      */
171:     private void drawSquare (Graphics graphics, int startX, int startY)
172:     {
173:         int dimension = rand(MIN_SIZE, MAX_SIZE);
174:         graphics.fillRect(startX, startY, dimension, dimension);
175:     }
176:
177:     /**
178:      * This method draws a rectangle at the provided X and Y location.
179:      *
180:      * @param graphics An instance of the Graphics class
181:      * @param startX   The X integer value of the location to draw the rectangle
182:      * @param startY   The Y integer value of the location to draw the rectangle
183:      */
184:     private void drawRectangle (Graphics graphics, int startX, int startY)
185:     {
186:         int width  = rand(MIN_SIZE, MAX_SIZE);
187:         int height = rand(MIN_SIZE, MAX_SIZE);
188:         graphics.fillRect(startX, startY, width, height);
189:     }
190:
191:     /**
192:      * This method draws a line at the provided X and Y location.
193:      *
194:      * @param graphics An instance of the Graphics class
195:      * @param startX   The X integer value of the location to draw the line
196:      * @param startY   The Y integer value of the location to draw the line
197:      */
198:     private void drawLine (Graphics graphics, int startX, int startY)
199:     {
200:         int endX = rand(0, screenSize.width);
201:         int endY = rand(0, screenSize.height);
202:         graphics.drawLine(startX, startY, endX, endY);
203:     }
204:
```

```
205:     /**
206:      * This is the 'paint' lifecycle method of the applet. It generates random
207:      * shapes of random color, location, and sizes and paints them to the
208:      * screen with a set delay in between each render. Once the number of shapes
209:      * on the screen has hit the maximum limit, the screen is cleared and
210:      * the counter is reset to start afresh.
211:      */
212:     public void paint (Graphics graphics)
213:     {
214:         int startX = rand(0, screenSize.width);
215:         int startY = rand(0, screenSize.height);
216: 
217:         Color color = getRandomColor();
218:         graphics.setColor(color);
219: 
220:         String shape = getParameter("shape");
221:         if (shape == null) shape = "circle";
222:         switch (shape)
223:         {
224:             case "circle":
225:                 drawCircle(graphics, startX, startY);
226:                 break;
227:             case "square":
228:                 drawSquare(graphics, startX, startY);
229:                 break;
230:             case "rectangle":
231:                 drawRectangle(graphics, startX, startY);
232:                 break;
233:             case "line":
234:                 drawLine(graphics, startX, startY);
235:                 break;
236:             default:
237:                 drawCircle(graphics, startX, startY);
238:                 break;
239:         }
240: 
241:         delay(DELAY_DURATION);
242:         repaint();
243: 
244:         count++;
245: 
246:         if (count == MAX_SHAPES)
247:         {
248:             clearScreen();
249:             count = 0;
250:         }
251:     }
252: }
```