```
 1: /**
 2:  * This applet generates shapes using Java's AWT Graphics API.
 3:  * It allows the user to click a button to generate a given shape in random
 4:  * sizes, colors, and locations set against a white background.
 5:  * The applet's height and width is set to the screen's height and width.
 6:  * The shape is defined as a parameter passed to the applet.
 7:  * Possible shapes include: 'circle', 'square', 'rectangle', and 'line'.
 8:  * If the user does not pass a 'shape' param or if param passed does not match
 9:  * one of the possible shapes, the applet defaults to drawing circles.
10:  *
11:  * @name    DrawShapes (Extra Credit 02)
12:  * @author  Ravi S. Ramphal
13:  * @class   CCSF CS111B
14:  * @date    2017.07.27
15:  * @version 1.0
16:  */
17:
18: import java.awt.*;
19: import java.awt.event.*;
20: import java.applet.*;
21:
22: // <applet code="DrawShapes" width="0" height="0">
23: //     <param name="shape" value="circle">
24: // </applet>
25:
26: public class DrawShapes extends Applet implements ActionListener
27: {
28:     /**
29:      * The current resolution of the target screen
30:      */
31:     private Dimension screenSize;
32:
33:     /**
34:      * The desired background color
35:      */
36:     private final Color BACKGROUND_COLOR = Color.WHITE;
37:
38:     /**
39:      * The minimum size (in pixels) of any shape generated
40:      */
41:     private final int MIN_SIZE = 20;
42:
43:     /**
44:      * The maximum size (in pixels) of any shape generated
45:      */
46:     private final int MAX_SIZE = 100;
47:
48:     /**
49:      * The size of the margin of the generating area
50:      */
51:     private final int MARGIN = 50;
52:
53:     /**
54:      * The X lower limit for any shape generated
55:      */
56:     private int minX;
57:
58:     /**
59:      * The X upper limit for any shape generated
60:      */
61:     private int maxX;
62:
63:     /**
64:      * The Y lower limit for any shape generated
65:      */
66:     private int minY;
67:
68:     /**
```

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

```
137:         maxX = screenSize.width - MARGIN;
138:         minY = MARGIN;
139:         maxY = screenSize.height - (MARGIN * 2);
140:     }
141:
142:     /**
143:      * This is the 'init' lifecycle method of the applet. It is executed once
144:      * on applet instantiation. It sets the applet size to the screen size,
145:      * defines the boundaries of the generation area, and sets the background
146:      * color. It also adds two buttons to the applet and assigns listeners.
147:      */
148:     public void init ()
149:     {
150:         setToScreenSize();
151:         defineBoundaries();
152:
153:         setBackground(BACKGROUND_COLOR);
154:
155:         addButton = new Button("Add Shape");
156:         clearButton = new Button("Clear Screen");
157:
158:         add(addButton);
159:         add(clearButton);
160:
161:         addButton.addActionListener(this);
162:         clearButton.addActionListener(this);
163:     }
164:
165:     /**
166:      * This is the 'actionPerformed' handler for the two buttons. They call
167:      * the correct respective actions depending on which button was pressed.
168:      *
169:      * @param event The ActionEvent fired by the buttons
170:      */
171:     public void actionPerformed (ActionEvent event)
172:     {
173:         if (event.getSource() == addButton)
174:         {
175:             Graphics g = getGraphics();
176:             paint(g);
177:             g.dispose();
178:         }
179:         else if (event.getSource() == clearButton)
180:         {
181:             clearScreen();
182:         }
183:     }
184:
185:     /**
186:      * This is the 'update' lifecycle method of the applet. It overwrites the
187:      * default 'update' method to only repain the screen WITHOUT clearing the
188:      * screen.
189:      *
190:      * @param graphics An instance of the Graphics class
191:      */
192:     public void update (Graphics graphics)
193:     {
194:         paint(graphics);
195:     }
196:
197:     /**
198:      * This method draws a circle.
199:      *
200:      * @param graphics An instance of the Graphics class
201:      */
202:     private void drawCircle (Graphics graphics)
203:     {
204:         int startX   = rand(minX, maxX);
```

```
205:         int startY   = rand(minY, maxY);
206:         int diameter = rand(MIN_SIZE, MAX_SIZE);
207:
208:         if ((startX + diameter) >= maxX) startX = maxX - diameter;
209:         if ((startY + diameter) >= maxY) startY = maxY - diameter;
210:
211:         graphics.fillOval(startX, startY, diameter, diameter);
212:     }
213:
214:     /**
215:      * This method draws a square.
216:      *
217:      * @param graphics An instance of the Graphics class
218:      */
219:     private void drawSquare (Graphics graphics)
220:     {
221:         int startX    = rand(minX, maxX);
222:         int startY    = rand(minY, maxY);
223:         int dimension = rand(MIN_SIZE, MAX_SIZE);
224:
225:         if ((startX + dimension) >= maxX) startX = maxX - dimension;
226:         if ((startY + dimension) >= maxY) startY = maxY - dimension;
227:
228:         graphics.fillRect(startX, startY, dimension, dimension);
229:     }
230:
231:     /**
232:      * This method draws a rectangle.
233:      *
234:      * @param graphics An instance of the Graphics class
235:      */
236:     private void drawRectangle (Graphics graphics)
237:     {
238:         int startX = rand(minX, maxX);
239:         int startY = rand(minY, maxY);
240:         int width  = rand(MIN_SIZE, MAX_SIZE);
241:         int height = rand(MIN_SIZE, MAX_SIZE);
242:
243:         if ((startX + width)  >= maxX) startX = maxX - width;
244:         if ((startY + height) >= maxY) startY = maxY - height;
245:
246:         graphics.fillRect(startX, startY, width, height);
247:     }
248:
249:     /**
250:      * This method draws a line.
251:      *
252:      * @param graphics An instance of the Graphics class
253:      */
254:     private void drawLine (Graphics graphics)
255:     {
256:         int startX = rand(minX, maxX);
257:         int startY = rand(minY, maxY);
258:         int endX   = rand(minX, maxX);
259:         int endY   = rand(minY, maxY);
260:
261:         graphics.drawLine(startX, startY, endX, endY);
262:     }
263:
264:     /**
265:      * This is the ‘paint‘ lifecycle method of the applet. It generates random
266:      * shapes of random color, location, and sizes and paints them to the
267:      * screen.
268:      */
269:     public void paint (Graphics graphics)
270:     {
271:         Color color = getRandomColor();
272:         graphics.setColor(color);
```

```
273:
274:         String shape = getParameter("shape");
275:         if (shape == null) shape = "circle";
276:         switch (shape)
277:         {
278:             case "circle":
279:                 drawCircle(graphics);
280:                 break;
281:             case "square":
282:                 drawSquare(graphics);
283:                 break;
284:             case "rectangle":
285:                 drawRectangle(graphics);
286:                 break;
287:             case "line":
288:                 drawLine(graphics);
289:                 break;
290:             default:
291:                 drawCircle(graphics);
292:                 break;
293:         }
294:     }
295: }
```