```
// joi/5/shapes/Line.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
 * A Line has a length and a paintChar used to paint
 * itself on a Screen.
 *
 * Subclasses of this abstract class specify the direction
 * of the Line.
 *
 * @version 5
 */

public abstract class Line
{
    protected int  length;      // length in (character) pixels.
    protected char paintChar;   // character used for painting.

    /**
     * Construct a Line.
     *
     * @param length length in (character) pixels.
     * @param paintChar character used for painting this Line.
     */

    protected Line( int length, char paintChar )
    {
        this.length    = length;
        this.paintChar = paintChar;
    }

    /**
     * Get the length of this line.
     *
     * @return the length in (character) pixels.
     */

    public int getLength()
    {
        return length;
    }

    /**
     * Set the length of this line.
     *
     * @param length the new length in (character) pixels.
     */

    public void setLength( int length )
    {
        this.length = length;
    }

    /**
```



```
     * Get the paintChar of this Line.
     *
     * @return the paintChar.
     */

    public char getPaintChar()
    {
        return paintChar;
    }

    /**
     * Set the paintChar of this Line.
     *
     * @param paintChar the new paintChar.
     */

    public void setPaintChar( char paintChar )
    {
        this.paintChar = paintChar;
    }

    /**
     * Paint this Line on Screen s at position (x,y).
     *
     * @param s the Screen on which this Line is to be painted.
     * @param x the x position for the line.
     * @param y the y position for the line.
     */

    public abstract void paintOn( Screen s, int x, int y );

    /**
     * Paint this Line on Screen s at position (0,0).
     *
     * @param s the Screen on which this Line is to be painted.
     */

    public void paintOn( Screen s )
    {
        paintOn( s, 0, 0 );
    }
}
```

```
 1  // joi/5/shapes/HLine.java
 2  //
 3  //
 4  // Copyright 2003 Bill Campbell and Ethan Bolker
 5
 6
 7  /**
 8   * An HLine is a horizontal Line.
 9   */
10
11  public class HLine extends Line
12  {
13      /**
14       * Construct an HLine having a paintChar and a length.
15       *
16       * @param length length in (character) pixels.
17       * @param paintChar character used for painting this Line.
18       */
19
20      public HLine( int length, char paintChar )
21      {
22          super( length, paintChar );
23      }
24
25      /**
26       * Paint this Line on Screen s at position (x,y).
27       *
28       * @param screen the Screen on which this Line is to be painted.
29       * @param x      the x position for the line.
30       * @param y      the y position for the line.
31       */
32
33      public void paintOn( Screen screen, int x, int y )
34      {
35          for ( int i = 0; i < length; i++ )
36              screen.paintAt( paintChar, x+i, y );
37      }
38
39      /**
40       * Unit test for class HLine.
41       */
42
43      public static void main( String[] args )
44      {
45          Terminal terminal = new Terminal();
46
47          terminal.println( "Self documenting unit test of HLine.");
48          terminal.println( "The two Screens that follow should match.");
49          terminal.println();
50          terminal.println( "Hard coded picture:");
51          terminal.println( "++++++++++++++++++++++");
52          terminal.println( "+xxxxxxxxxx          +");
53          terminal.println( "+xxxxx               +");
54          terminal.println( "+                   +");
55          terminal.println( "+   *****           +");
56          terminal.println( "+    1              +");
```



```
57          terminal.println( "+                   +");
58          terminal.println( "++++++++++++++++++++++");
59          terminal.println();
60
61          terminal.println( "Picture drawn using HLine methods:");
62          Screen screen = new Screen( 20, 6 );
63
64          Line hline = new HLine( 10, 'x' );
65          hline.paintOn( screen );
66
67          hline.setLength(5);
68          hline.paintOn( screen, 0, 1 );
69
70          hline.setPaintChar('*');
71          hline.paintOn( screen, 3, 3 );
72
73          hline.setLength(1);
74          hline.setPaintChar('1');
75          hline.paintOn( screen, 4, 4 );
76
77          screen.draw( terminal );
78      }
79
80  }
```

```
 1  // joi/5/shapes/VLine.java
 2  //
 3  //
 4  // Copyright 2003 Bill Campbell and Ethan Bolker
 5
 6  /**
 7   * A VLine is a vertical Line.
 8   */
 9
10  public class VLine extends Line
11  {
12      /**
13       * Construct a VLine having a paintChar and a length.
14       *
15       * @param length length in (character) pixels.
16       * @param paintChar character used for painting this Line.
17       */
18
19      public VLine( int length, char paintChar )
20      {
21          super( length, paintChar );
22      }
23
24      /**
25       * Paint this Line on Screen s at position (x,y).
26       *
27       * @param screen the Screen on which this Line is to be painted.
28       * @param x      the x position for the line.
29       * @param y      the y position for the line.
30       */
31
32      public void paintOn( Screen screen, int x, int y )
33      {
34          for ( int i = 0; i < length; i++ )
35              screen.paintAt( paintChar, x, y+i );
36      }
37
38      /**
39       * Unit test for class VLine.
40       */
41
42      public static void main( String[] argv )
43      {
44          Terminal terminal = new Terminal();
45
46          terminal.println( "Self documenting unit test of VLine.");
47          terminal.println( "The two Screens that follow should match.");
48          terminal.println();
49          terminal.println( "Hard coded picture:");
50          terminal.println( "+++++++++");
51          terminal.println( "+xx     +");
52          terminal.println( "+xx     +");
53          terminal.println( "+xx     +");
54          terminal.println( "+xx *   +");
55          terminal.println( "+xx *1  +");
56          terminal.println( "+x  *   +");
```



```
57          terminal.println( "+x  *   +");
58          terminal.println( "+   *   +");
59          terminal.println( "+       +");
60          terminal.println( "+++++++++");
61          terminal.println();
62
63          terminal.println( "Picture drawn using VLine methods:");
64          Screen screen = new Screen( 7,  9 );
65
66          Line vLine = new VLine(  7, 'x' );
67          vLine.paintOn( screen );
68
69          vLine.setLength(5);
70          vLine.paintOn( screen, 1, 0 );
71
72          vLine.setPaintChar('*');
73          vLine.paintOn( screen, 3, 3 );
74
75          vLine.setLength(1);
76          vLine.setPaintChar('1');
77          vLine.paintOn( screen, 4, 4 );
78
79          screen.draw( terminal );
80      }
81
82  }
```

```
// joi/5/shapes/ShapeOnScreen.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

// This file is used in one of the Chapter 5 exercises on shapes.

/**
 * A ShapeOnScreen models a Shape to be painted at
 * a given position on a Screen.
 *
 * @see Shape
 * @see Screen
 *
 * @version 5
 */

public class ShapeOnScreen
{
    private Shape shape;
    private int x;
    private int y;

    /**
     * Construct a ShapeOncreen.
     *
     * @param shape the Shape
     * @param x its x coordinate
     * @param y its y coordinate
     */

    public ShapeOnScreen( Shape shape, int x, int y )
    {
        this.shape = shape;
        this.x     = x;
        this.y     = y;
    }

    /**
     * What Shape does this ShapeOnScreen represent?
     *
     * @return the Shape.
     */

    public Shape getShape() {
        return shape;
    }

    /**
     * The current x coordinate of this ShapeOnScreen.
     *
     * @return the x coordinate.
     */

    public int getX() {
        return x;
```



```
    }

    /**
     * The current y coordinate of this ShapeOnScreen.
     *
     * @return the y coordinate.
     */

    public int getY() {
        return y;
    }

    /**
     * Unit test.
     */

    public static void main( String[] args ) {
        ShapeOnScreen sos = new ShapeOnScreen( null, 5, 7);
        System.out.println("Shape: " + sos.getShape());
        System.out.println("x:     " + sos.getX());
        System.out.println("y:     " + sos.getY());
    }
}
```