```
 1  // joi/3/shapes/DemoShapes.java
 2  //
 3  //
 4  // Copyright 2003 Bill Campbell and Ethan Bolker
 5
 6  /**
 7   * A short demonstration program for HLine and Box.
 8   *
 9   * @version 3
10   */
11
12  public class DemoShapes
13  {
14      /**
15       * Paint some shapes on a Screen and draw it to a Terminal.
16       */
17
18      public static void main( String[] args )
19      {
20          Terminal t = new Terminal();
21          Screen   s = new Screen( 36, 12 );
22
23          HLine h1 = new HLine( 10, 'R' );
24          Box   b1 = new Box( 5, 6, 'G' );
25          Box   b2 = new Box( 5, 6, 'B' );
26
27          h1.paintOn( s ); // at position (0,0)
28          b1.paintOn( s, 2, 2 );
29          b2.paintOn( s, 4, 5 );
30
31          t.println( "A Screen with an HLine and two Boxes:" );
32          s.draw( t );
33      }
34  }
```

```
 1   // joi/3/shapes/HLine.java
 2   //
 3   //
 4   // Copyright 2003 Bill Campbell and Ethan Bolker
 5
 6   /**
 7    * A horizontal line has a length and a paintChar used
 8    * used to paint the line on a Screen.
 9    *
10    * @version 3
11    */
12
13   public class HLine
14   {
15       private int  length;       // length in (character) pixels.
16       private char paintChar;    // character used for painting.
17
18       /**
19        * Construct an HLine.
20        *
21        * @param length length in (character) pixels.
22        * @param paintChar character used for painting this Line.
23        */
24
25       public HLine( int length, char paintChar )
26       {
27           this.length = length;
28           this.paintChar = paintChar;
29       }
30
31       /**
32        * Paint this HLine on Screen s at position (x,y).
33        *
34        * @param s the Screen on which this line is to be painted.
35        * @param x the x position for the line.
36        * @param y the y position for the line.
37        */
38
39       public void paintOn( Screen s, int x, int y )
40       {
41           for ( int i = 0; i < length; i = i+1 ) {
42               s.paintAt( paintChar, x+i, y );
43           }
44       }
45
46       /**
47        * Paint this HLine on Screen s at position (0,0).
48        *
49        * @param s the Screen on which this line is to be painted.
50        */
51
52       public void paintOn( Screen s )
53       {
54           paintOn( s, 0, 0 );
55       }
56   }
```

```
 57      /**
 58       * Get the length of this line.
 59       *
 60       * @return the length in (character) pixels.
 61       */
 62
 63      public int getLength()
 64      {
 65          return length;
 66      }
 67
 68      /**
 69       * Set the length of this line.
 70       *
 71       * @param length the new length in (character) pixels.
 72       */
 73
 74      public void setLength( int length )
 75      {
 76          this.length = length;
 77      }
 78
 79      /**
 80       * Unit test for class HLine,
 81       * assuming Screen and Terminal work.
 82       */
 83
 84      public static void main( String[] args )
 85      {
 86          Terminal terminal = new Terminal();
 87
 88          terminal.println( "Unit test of HLine." );
 89          terminal.println( "You should see this Screen twice: " );
 90          terminal.println( "++++++++++++++++++++++++" );
 91          terminal.println( "+xxxxxxxxxx             +" );
 92          terminal.println( "+xxxxx                  +" );
 93          terminal.println( "+                       +" );
 94          terminal.println( "+          *****        +" );
 95          terminal.println( "+             1         +" );
 96          terminal.println( "+                       +" );
 97          terminal.println( "++++++++++++++++++++++++" );
 98          terminal.println( "" );
 99
100          Screen    screen  = new Screen( 20, 6 );
101
102          HLine hline1 = new HLine( 10, 'x' );
103          HLine hline2 = new HLine(  5, '*' );
104          HLine hline3 = new HLine(  1, '1' );
105
106          hline1.paintOn( screen );
107          hline1.setLength(5);
108          hline1.paintOn( screen, 0, 1 );
109          hline2.paintOn( screen, 3, 3 );
110          hline3.paintOn( screen, 4, 4 );
111
112          screen.draw( terminal );
```

```
113        }
114 }
```

```
1   // joi/3/shapes/Box.java
2   //
3   //
4   // Copyright 2003 Bill Campbell and Ethan Bolker
5
6   /**
7    * A Box has a width, a height and a paintChar used
8    * used to paint the Box on a Screen.
9    *
10   * <pre>
11   * Examples:
12   * new Box( 3, 4, 'G' )    new Box( 1, 1, '$' )
13   *
14   *        GGG
15   *        GGG                     $
16   *        GGG
17   *        GGG
18   * </pre>
19   *
20   * @version 3
21   */
22
23  public class Box
24  {
25      private int   width;          // width  in (character) pixels
26      private int   height;         // height in (character) pixels
27      private char  paintChar;      // character used for painting
28
29      /**
30       * Construct a box.
31       *
32       * @param width width in (character) pixels.
33       * @param height height in (character) pixels.
34       * @param paintChar character used for painting this Box.
35       */
36
37      public Box( int width, int height, char paintChar )
38      {
39          this.width     = width;
40          this.height    = height;
41          this.paintChar = paintChar;
42      }
43
44      /**
45       * Paint this Box on Screen s at position (0,0).
46       *
47       * @param s the screen on which this box is to be painted.
48       * @param x the x position for the box.
49       * @param y the y position for the box.
50       */
51      public void paintOn( Screen s, int x, int y )
52      {
53          HLine hline = new HLine( width, paintChar );
54          for ( int i = 0; i < height; i++ )
55              hline.paintOn( s, x, y+i );
56
```

```
57      }
58
59      /**
60       * Paint this Box on Screen s at position (0,0).
61       *
62       * @param s the Screen on which this box is to be painted.
63       */
64      public void paintOn( Screen s )
65      {
66          paintOn( s, 0, 0 );  // or this.paintOn(s,0,0);
67      }
68
69      /**
70       * Get the width of this Box.
71       *
72       * @return width of box (expressed as a number
73       *         of characters).
74       */
75      public int getWidth()
76      {
77          return width;
78      }
79
80      /**
81       * Get the height of this Box.
82       *
83       * @return the height in (character) pixels.
84       */
85      public int getHeight()
86      {
87          return height;
88      }
89
90      /**
91       * Set the width of this Box.
92       *
93       * @param width the new width in (character) pixels.
94       */
95      public void setWidth( int width )
96      {
97          this.width = width;
98      }
99
100     /**
101      * Set the height of this Box.
102      *
103      * @param height the new height in (character) pixels.
104      */
105     public void setHeight( int height )
106     {
107
108
109
110
111
112
```

```
113          this.height = height;
114      }
115
116      /**
117       * Unit test for class Box,
118       * assuming Screen and Terminal work.
119       */
120
121      public static void main( String[] args )
122      {
123          Terminal terminal = new Terminal();
124
125          terminal.println( "Unit test of Box." );
126          terminal.println( "You should see this Screen twice: " );
127          terminal.println( "+++++++++++++++++++++++++" );
128          terminal.println( "+RRRR                    +" );
129          terminal.println( "+RRRR                    +" );
130          terminal.println( "+RRGGG                   +" );
131          terminal.println( "+RRGGG                   +" );
132          terminal.println( "+RRGGG                   +" );
133          terminal.println( "+  GGRRRRRRR             +" );
134          terminal.println( "+++++++++++++++++++++++++" );
135          terminal.println();
136
137          Screen screen   = new Screen( 20, 6 );
138
139          Box box1 = new Box( 4, 5, 'R' );
140          Box box2 = new Box( 3, 4, 'G' );
141
142          box1.paintOn( screen );
143          box2.paintOn( screen, 2, 2 );
144
145          // test reference model for objects
146          box2 = box1;
147          int oldWidth = box2.getWidth();
148          box1.setWidth(oldWidth+3);
149          box2.paintOn( screen, 4, 5 );
150
151          screen.draw( terminal );
152      }
153 }
```

```
 1   // joi/3/shapes/TestShapes.java
 2   //
 3   //
 4   // Copyright 2003 Bill Campbell and Ethan Bolker
 5
 6   /**
 7    * A program to test shapes.
 8    *
 9    * @version 3
10    */
11
12   class TestShapes
13   {
14      /**
15       * Paint shapes on a Screen and draw it to a Terminal.
16       */
17
18      public static void main( String[] argv )
19      {
20         Screen s;
21
22         Terminal t = new Terminal();
23
24         t.println( "An empty 10 x 3 Screen:" );
25         s = new Screen( 10, 3 );
26         s.draw( t );
27
28         t.println( "A 20 x 10 Screen with 3 HLines:" );
29         s = new Screen( 20, 10 );
30         HLine h1 = new HLine( 10, 'R' );
31         HLine h2 = new HLine( 15, 'G' );
32
33         h1.paintOn( s, 0, 0 );
34         h2.paintOn( s, 0, 1 );
35         (new HLine( 15, 'B' )).paintOn( s, 0, 2 ); // tricky to read
36         s.draw( t );
37
38         t.println( "Clear that screen," );
39         s.clear();
40
41         t.println( "draw 3 Boxes (2 overlapping):" );
42         Box    b = new Box( 6, 5, 'R' );
43         b.paintOn( s, 1, 1 );
44         b = new Box( 7, 4, 'G' ); // create a new (different) Box b
45         b.paintOn( s, 2, 3 ); // paint Box b on s
46         b.paintOn( s, 17, 5 ); // paint Box b partly off the Screen
47         s.draw( t );
48      }
49   }
```

```java
1  // joi/3/shapes/InteractiveShapes.java
2  //
3  //
4  // Copyright 2003 Bill Campbell and Ethan Bolker
5
6  /**
7   * Interactive program to study shapes.
8   *
9   * @version3
10  */
11
12 public class InteractiveShapes
13 {
14    public static void main( String[] args )
15    {
16       Terminal t = new Terminal();
17       Screen s = new Screen(
18                      t.readInt("screen width:  "),
19                      t.readInt("screen height: "));
20
21       char c = 'a';
22       int x,y;
23       while ( t.readYesOrNo("more") ) {
24          char shape = t.readChar("h(line), b(ox), c(lear): ");
25          switch (shape) {
26          case 'h':
27             int length = t.readInt("HLine length: ");
28             x = t.readInt("x coordinate: ");
29             y = t.readInt("y coordinate: ");
30             (new HLine(length, c++).paintOn(s,x,y);
31             break;
32          case 'b':
33             int w = t.readInt("Box width: ");
34             int h = t.readInt("Box height: ");
35             x = t.readInt("x coordinate: ");
36             y = t.readInt("y coordinate: ");
37             (new Box(w,h,c++).paintOn(s,x,y);
38             break;
39          case 'c':
40             s.clear();
41             break;
42          default:
43             t.println("try again");
44             continue;
45          }
46          s.draw(t);
47       }
48    }
49 }
```

```
1   // joi/3/shapes/TextLine.java
2   //
3   //
4   // Copyright 2003 Bill Campbell and Ethan Bolker
5   //
6   // This file contains stubs for the methods.
7
8   /**
9    * A horizontal line of character text.
10   *
11   * @version 3
12   */
13
14  public class TextLine
15  {
16      /**
17       * Construct a TextLine.
18       *
19       * @param text the text of the line.
20       */
21
22      public TextLine( String text )
23      {
24      }
25
26      /**
27       * Paint this TextLine on Screen s at position (x,y).
28       *
29       * @param s the Screen on which this line is to be painted.
30       * @param x the x position for the line.
31       * @param y the y position for the line.
32       */
33
34      public void paintOn( Screen s, int x, int y )
35      {
36      }
37
38      /**
39       * Draw the TextLine to Screen s at position (0,0).
40       *
41       * @param s the Screen on which this line is to be painted.
42       */
43
44      public void paintOn( Screen s )
45      {
46          paintOn( s, 0, 0 );
47      }
48
49      /**
50       * Get the length of this line.
51       *
52       * @return the length in (character) pixels.
53       */
54
55      public int getLength()
56      {
```

```
57          return 0; // replace with the right answer
58      }
59
60      /**
61       * Unit test for class.TextLine,
62       * assuming Screen and Terminal work.
63       */
64
65      public static void main( String[] args )
66      {
67      }
68  }
```

```java
 1   // joi/3/shapes/Screen.java
 2   //
 3   //
 4   // Copyright 2003 Bill Campbell and Ethan Bolker
 5
 6   /**
 7    * A Screen is a (width*height) grid of (character) 'pixels'
 8    * on which we may paint various shapes.  It can be drawn to
 9    * a Terminal.
10    *
11    * @version 3
12    */
13
14   public class Screen
15   {
16     /**
17      * The character used to paint the screen's frame.
18      */
19
20     private static final char FRAMECHAR = '+';
21     private static final char BLANK = ' ';
22     private int width;
23     private int height;
24     private char[][] pixels;
25
26     /**
27      * Construct a Screen.
28      *
29      * @param width the number of pixels in the x direction.
30      * @param height the number of pixels in the y direction.
31      */
32
33     public Screen( int width, int height )
34     {
35       this.width  = width;
36       this.height = height;
37       pixels = new char[width][height];
38       clear();
39     }
40
41     /**
42      * Clear the Screen, painting a blank at every pixel.
43      */
44
45     public void clear()
46     {
47       for (int x = 0; x < width; x++) {
48         for ( int y = 0; y < height; y++ ) {
49           pixels[x][y] = BLANK;
50         }
51       }
52     }
53
54     /**
55      * Paint a character pixel at position (x,y).
56      *
```

```java
57    * @param c the character to be painted.
58    * @param x the (horizontal) x position.
59    * @param y the (vertical) y position.
60    */
61
62   public void paintAt( char c, int x, int y )
63   {
64     if ( 0 <= x && x < width &&
65          0 <= y && y < height) {
66       pixels[x][y] = c;
67     }
68     // Otherwise off the Screen - nothing is painted.
69   }
70
71   /**
72    * How wide is this Screen?
73    *
74    * @return the width.
75    */
76
77   public int getWidth()
78   {
79     return width;
80   }
81
82   /**
83    * How high is this Screen?
84    *
85    * @return the height.
86    */
87
88   public int getHeight()
89   {
90     return height;
91   }
92
93   /**
94    * Draw this Screen on a Terminal.
95    *
96    * @param t the Terminal on which to draw this Screen.
97    */
98
99   public void draw( Terminal t )
100  {
101    for ( int col = -1; col < width+1 ; col++ ) {     // top edge
102      t.print(FRAMECHAR);
103    }
104    t.println();
105    for ( int row = 0; row < height; row++ ) {
106      t.print(FRAMECHAR);                             // left edge
107      for ( int col = 0; col < width; col++ ) {
108        t.print( pixels[col][row] );
109      }
110      t.println( FRAMECHAR );                         // right edge
111    }
112    for ( int col = -1; col < width+1 ; col++ ) {     // bottom edge
```

```
113            t.print(FRAMECHAR);
114        }
115        t.println();
116    }
117 }
```