/**
 * 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 );
    }
}

/**
 * Copyright 2003 Bill Campbell and Ethan Bolker
 */
/* An HLine is a horizontal Line. */

public class HLine extends Line {

  public HLine( int length, char paintChar ) {
    super( length, paintChar );
  }

  public void paintOn( Screen screen, int x, int y ) {
    for ( int i = 0; i < length; i++ )
      screen.paintAt( paintChar, x+i, y );
  }

  public static void main( String[] args ) {
    Terminal terminal = new Terminal();

    terminal.println( "Self documenting unit test of HLine.");
    terminal.println( "The two Screens that follow should match:");
    terminal.println( "++++++++++++++++++++++");
    terminal.println( "+xxxxxxxxxx          +");
    terminal.println( "+xxxxx               +");
    terminal.println( "+                    +");
    terminal.println( "+   *****            +");
    terminal.println( "+    1               +");
    terminal.println( "+                    +");
    terminal.println( "++++++++++++++++++++++");

    terminal.println( "Picture drawn using HLine methods:");
    Screen screen = new Screen( 20, 6 );
    Line hline = new HLine( 10, 'x' );
    hline.paintOn( screen );
    hline.setLength(5);
    hline.paintOn( screen, 0, 1 );
    hline.setPaintChar('*');
    hline.paintOn( screen, 3, 3 );
    hline.setLength(1);
    hline.setPaintChar('1');
    hline.paintOn( screen, 4, 4 );
    screen.draw( terminal );
  }
}

public class VLine extends Line {

    public VLine(int length, char paintChar)
    {
        super(length, paintChar);
    }

    public void paintOn(Screen screen, int x, int y)
    {
        for (int i = 0; i < length; i++)
            screen.paintAt(paintChar, x + i, y);
    }

    public static void main(String[] argv)
    {
        Terminal terminal = new Terminal();

        screen.println("Self documenting unit test of VLine.");
        screen.println("The two Screens that follow should match.");
        screen.println();
        screen.println("Hard coded picture:");
        screen.println("+********+");
        screen.println("+xx    xx+");
        screen.println("+xx    xx+");
        screen.println("+xx    xx+");
        screen.println("+xx *   xx+");
        screen.println("+xx *1  xx+");
        screen.println("+x  *   xx+");
        screen.println("+x  *   xx+");
        screen.println("+   *   xx+");
        screen.println("+       xx+");
        screen.println("+********+");
        screen.println();

        screen.println("Picture drawn using VLine methods:");
        Screen screen = new Screen(7, 9);
        Line vLine = new VLine(7, 'x');
        vLine.paintOn(screen);
        vLine.setLength(5);
        vLine.paintOn(screen, 1, 0);
        vLine.setPaintChar('*');
        vLine.paintOn(screen, 3, 3);
        vLine.setLength(1);
        vLine.setPaintChar('1');
        vLine.paintOn(screen, 4, 4);
        screen.draw(terminal);
    }
}
// joi/5/shapes/ShapeOnScreen.java

/**
 * 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 ShapeOnscreen.
 *
 * @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());
}
}