```java
// joi/3/textfiles/TextFile.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

import java.util.Date;

/**
 * A TextFile mimics the sort of text file that one finds
 * on a computer’s file system.  It has an owner,
 * a create date (when the file was created),
 * a modification date (when the file was last modified),
 * and String contents.
 *
 * @version 3
 */

public class TextFile
{
    // Private Implementation

    private String owner;       // Who owns the file.
    private Date   createDate;  // When the file was created.
    private Date   modDate;     // When the file was last modified.
    private String contents;    // The text stored in the file.

    // Public Interface

    /**
     * Construct a new TextFile with given owner and
     * contents; set the creation and modification dates.
     *
     * @param owner the user who owns the file.
     * @param contents the file’s initial contents.
     */

    public TextFile( String owner, String contents )
    {
        this.owner    = owner;
        this.contents = contents;
        createDate    = new Date(); // date and time now
        modDate       = createDate;
    }

    /**
     * Replace the contents of the file.
     *
     * @param contents the new contents.
     */

    public void setContents( String contents )
    {
        this.contents = contents;
        modDate = new Date();
    }

```

```java
    /**
     * The contents of a file.
     *
     * @return String contents of the file.
     */

    public String getContents()
    {
        return contents;
    }

    /**
     * Append text to the end of the file.
     *
     * @param  text the text to be appended.
     */

    public void append( String text )
    {
        this.setContents( contents + text );
    }

    /**
     * Append a new line of text to the end of the file.
     *
     * @param  text the text to be appended.
     */

    public void appendLine( String text )
    {
        this.setContents(contents + ’\n’ + text);
    }

    /**
     * The size of a file.
     *
     * @return the integer size of the file
     *  (the number of characters in its String contents)
     */

    public int getSize()
    {
        int charCount;
        charCount = contents.length();
        return charCount;
    }

    /**
     * The data and time of the file’s creation.
     *
     * @return the file’s creation date and time.
     */

    public String getCreateDate()
    {
        return createDate.toString();
    }

    /**
     * The date and time of the file’s last modification.
     *
     * @return the date and time of the file’s last modification.
     */

    public String getModDate()
    {
        return modDate.toString();
    }

    /**
     * The file’s owner.
     *
     * @return the owner of the file.
     */

    public String getOwner()
    {
        return owner;
    }

    /**
     * A definition of main(), used only for testing this class.
     *
     * Executing
     * <pre>
     *     %> java TextFile
     * </pre>
     * produces the output:
     * <pre>
     * TextFile myTextFile contains 13 characters.
     * Created by bill, Sat Dec 29 14:02:37 EST 2001
     * Hello, world.
     *
     * append new line "How are you today?"
     * Hello, world.
     * How are you today?
     * TextFile myTextFile contains 32 characters.
     * Modified Sat Dec 29 14:02:38 EST 2001
     * </pre>
     */

    public static void main( String[] args )
    {
        Terminal terminal = new Terminal();
        TextFile myTextFile
            = new TextFile( "bill", "Hello, world." );
        terminal.println( "TextFile myTextFile contains " +
                          myTextFile.getSize() +
                          " characters.");
        terminal.println( "Created by " +
                          myTextFile.getOwner() + ", " +
                          myTextFile.getCreateDate() );
```

```java
        terminal.println( myTextFile.getContents() );
        terminal.println();
        terminal.println(
                 "append new line \"How are you today?\"" );
        myTextFile.appendLine( "How are you today?" );
        terminal.println( myTextFile.getContents() );
        terminal.println( "TextFile myTextFile contains " +
                          myTextFile.getSize() +
                          " characters.");
        terminal.println( "Modified "  +
                          myTextFile.getModDate() );
    }
}
```

```java
// joi/3/shapes/DemoShapes.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
 * A short demonstration program for HLine and Box.
 *
 * @version 3
 */

public class DemoShapes
{
    /**
     * Paint some shapes on a Screen and draw it to a Terminal.
     */

    public static void main( String[] args )
    {
        Terminal t = new Terminal();
        Screen   s = new Screen( 36, 12 );

        HLine h1 = new HLine( 10, 'R' );
        Box   b1 = new Box( 5, 6, 'G' );
        Box   b2 = new Box( 5, 6, 'B' );

        h1.paintOn( s ); // at position (0,0)
        b1.paintOn( s, 2, 2 );
        b2.paintOn( s, 4, 5 );

        t.println( "A Screen with an HLine and two Boxes:" );
        s.draw( t );
    }
}
```

```java
// joi/3/shapes/HLine.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
 * A horizontal line has a length and a paintChar used
 * used to paint the line on a Screen.
 *
 * @version 3
 */

public class HLine
{
    private int  length;      // length in (character) pixels.
    private char paintChar;   // character used for painting.

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

    public HLine( int length, char paintChar )
    {
        this.length = length;
        this.paintChar = paintChar;
    }

    /**
     * Paint this HLine 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 void paintOn( Screen s, int x, int y )
    {
        for ( int i = 0; i < length; i = i+1 ) {
            s.paintAt( paintChar, x+i , y );
        }
    }

    /**
     * Paint this HLine 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 );
    }

```

```java
    /**
     * 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;
    }

    /**
     * Unit test for class HLine,
     * assuming Screen and Terminal work.
     */

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

        terminal.println( "Unit test of HLine.");
        terminal.println( "You should see this Screen twice: " );
        terminal.println( "++++++++++++++++++++++");
        terminal.println( "+xxxxxxxxxx          +");
        terminal.println( "+xxxxx               +");
        terminal.println( "+                    +");
        terminal.println( "+   *****            +");
        terminal.println( "+    1               +");
        terminal.println( "+                    +");
        terminal.println( "++++++++++++++++++++++");
        terminal.println( "");

        Screen   screen   = new Screen( 20, 6 );

        HLine hline1 = new HLine( 10, 'x' );
        HLine hline2 = new HLine(  5, '*' );
        HLine hline3 = new HLine(  1, '1' );

        hline1.paintOn( screen );
        hline1.setLength(5);
        hline1.paintOn( screen, 0, 1 );
        hline2.paintOn( screen, 3, 3 );
        hline3.paintOn( screen, 4, 4 );

        screen.draw( terminal );
    }
}
```

```java
// joi/3/shapes/Box.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
 * A Box has a width, a height and a paintChar used
 * used to paint the Box on a Screen.
 *
 * Examples:
 * <pre>
 *  new Box( 3, 4, ’G’ )   new Box( 1, 1, ’$’ )
 *
 *         GGG                     $
 *         GGG
 *         GGG
 *         GGG
 * </pre>
 *
 * @version 3
 */

public class Box
{
    private int  width;      // width  in (character) pixels
    private int  height;     // height in (character) pixels
    private char paintChar;  // character used for painting

    /**
     * Construct a box.
     *
     * @param width width in (character) pixels.
     * @param height height in (character) pixels.
     * @param paintChar character used for painting this Box.
     */

    public Box( int width, int height, char paintChar )
    {
        this.width     = width;
        this.height    = height;
        this.paintChar = paintChar;
    }

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

    public void paintOn( Screen s, int x, int y )
    {
        HLine hline = new HLine( width, paintChar );
        for ( int i = 0; i < height; i++ ) {
            hline.paintOn( s, x, y+i );
```

```java
        }
    }

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

    public void paintOn( Screen s )
    {
        paintOn( s, 0, 0 );  // or this.paintOn(s,0,0);
    }

    /**
     * Get the width of this Box.
     *
     * @return width of box (expressed as a number
     *         of characters).
     */

    public int getWidth()
    {
        return width;
    }

    /**
     * Get the height of this Box.
     *
     * @return the height in (character) pixels.
     */

    public int getHeight()
    {
        return height;
    }

    /**
     * Set the width of this Box.
     *
     * @param width the new width in (character) pixels.
     */

    public void setWidth( int width )
    {
        this.width = width;
    }

    /**
     * Set the height of this Box.
     *
     * @param height the new height in (character) pixels.
     */

    public void setHeight( int height )
    {
        this.height = height;
    }

    /**
     * Unit test for class Box,
     * assuming Screen and Terminal work.
     */

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

        terminal.println( "Unit test of Box.");
        terminal.println( "You should see this Screen twice: " );
        terminal.println( "++++++++++++++++++++++");
        terminal.println( "+RRRR                +");
        terminal.println( "+RRRR                +");
        terminal.println( "+RRGGG               +");
        terminal.println( "+RRGGG               +");
        terminal.println( "+RRGGG               +");
        terminal.println( "+  GGRRRRRRR         +");
        terminal.println( "++++++++++++++++++++++");
        terminal.println();

        Screen screen   = new Screen( 20, 6 );

        Box box1 = new Box( 4, 5, 'R' );
        Box box2 = new Box( 3, 4, 'G' );

        box1.paintOn( screen );
        box2.paintOn( screen, 2, 2 );

        // test reference model for objects
        box2 = box1;
        int oldWidth = box2.getWidth();
        box1.setWidth(oldWidth+3);
        box2.paintOn( screen, 4, 5 );

        screen.draw( terminal );
    }
}
```

```java
// joi/3/shapes/TestShapes.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
 * A program to test shapes.
 *
 * @version 3
 */

class TestShapes
{
    /**
     * Paint shapes on a Screen and draw it to a Terminal.
     */

    public static void main( String[] argv )
    {
        Terminal t = new Terminal();
        Screen s;

        t.println( "An empty 10 x 3 Screen:" );
        s = new Screen( 10, 3 );
        s.draw( t );

        t.println( "A 20 x 10 Screen with 3 HLines:" );
        s = new Screen( 20, 10 );
        HLine h1 = new HLine( 10, 'R' );
        HLine h2 = new HLine( 15, 'G' );

        h1.paintOn( s, 0, 0 );
        h2.paintOn( s, 0, 1 );
        (new HLine( 15, 'B' )).paintOn( s, 0, 2 ); // tricky to read
        s.draw( t );

        t.println( "Clear that screen,");
        s.clear();

        t.println( "draw 3 Boxes (2 overlapping):" );
        Box   b = new Box( 6, 5, 'R' );
        b.paintOn( s, 1, 1 );
        b = new Box( 7, 4, 'G'); // create a new (different) Box b
        b.paintOn( s, 2, 3 );    // paint Box b on s
        b.paintOn( s, 17, 5 );   // paint Box b partly off the Screen
        s.draw( t );
    }
}
```

```java
// joi/3/shapes/InteractiveShapes.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
 * Interactive program to study shapes.
 *
 * @version3
 */

public class InteractiveShapes
{
    public static void main( String[] args )
    {
        Terminal t = new Terminal();
        Screen s = new Screen(
                              t.readInt("screen width:  "),
                              t.readInt("screen height: "));
        char c = 'a';
        int x,y;
        while ( t.readYesOrNo("more")) {
            char shape = t.readChar("h(line), b(ox), c(lear): ");
            switch (shape) {
            case 'h':
                int length = t.readInt("HLine length: ");
                x          = t.readInt("x coordinate: ");
                y          = t.readInt("y coordinate: ");
                (new HLine(length, c++)).paintOn(s,x,y);
                break;
            case 'b':
                int w = t.readInt("Box width: ");
                int h = t.readInt("Box height: ");
                x     = t.readInt("x coordinate: ");
                y     = t.readInt("y coordinate: ");
                (new Box(w,h,c++)).paintOn(s,x,y);
                break;
            case 'c':
                s.clear();
                break;
            default:
                t.println("try again");
                continue;
            }
            s.draw(t);
        }
    }
}
```

```java
// joi/3/shapes/TextLine.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

// This file contains stubs for the methods.

/**
 * A horizontal line of character text.
 *
 * @version 3
 */

public class TextLine
{
    /**
     * Construct a TextLine.
     *
     * @param text the text of the line.
     */

    public TextLine( String text )
    {
    }

    /**
     * Paint this TextLine 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 void paintOn( Screen s, int x, int y )
    {
    }

    /**
     * Draw the TextLine to 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 );
    }

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

    public int getLength()
    {
```

```java
        return 0; // replace with the right answer
    }

    /**
     * Unit test for class.TextLine,
     * assuming Screen and Terminal work.
     */

    public static void main( String[] args )
    {
    }
}
```

```java
// joi/3/shapes/Screen.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
 * A Screen is a (width*height) grid of (character) 'pixels'
 * on which we may paint various shapes.  It can be drawn to
 * a Terminal.
 *
 * @version 3
 */

public class Screen
{
    /**
     * The character used to paint the screen's frame.
     */

    private static final char FRAMECHAR = '+';
    private static final char BLANK = ' ';
    private int width;
    private int height;
    private char[][] pixels;

    /**
     * Construct a Screen.
     *
     * @param width the number of pixels in the x direction.
     * @param height the number of pixels in the y direction.
     */

    public Screen( int width, int height )
    {
        this.width  = width;
        this.height = height;
        pixels = new char[width][height];
        clear();
    }

    /**
     * Clear the Screen, painting a blank at every pixel.
     */

    public void clear()
    {
        for (int x  = 0; x < width; x++) {
            for ( int y = 0; y < height; y++ ) {
                pixels[x][y] = BLANK;
            }
        }
    }

    /**
     * Paint a character pixel at position (x,y).
     *
```

```java
     * @param c the character to be painted.
     * @param x the (horizontal) x position.
     * @param y the (vertical) y position.
     */

    public void paintAt( char c, int x, int y )
    {
        if ( 0 <= x && x < width &&
             0 <= y && y < height) {
            pixels[x][y] = c;
        }
        // Otherwise off the Screen - nothing is painted.
    }

    /**
     * How wide is this Screen?
     *
     * @return the width.
     */

    public int getWidth()
    {
        return width;
    }

    /**
     * How high is this Screen?
     *
     * @return the height.
     */

    public int getHeight()
    {
        return height;
    }

    /**
     * Draw this Screen on a Terminal.
     *
     * @param t the Terminal on which to draw this Screen.
     */

    public void draw( Terminal t )
    {
        for ( int col = -1; col < width+1 ; col++ ) {  // top edge
            t.print(FRAMECHAR);
        }
        t.println();
        for ( int row = 0; row < height; row++ ) {
            t.print(FRAMECHAR);                    // left edge
            for ( int col = 0; col < width; col++ ) {
                t.print( pixels[col][row] );
            }
            t.println( FRAMECHAR );                 // right edge
        }
        for ( int col = -1; col < width+1 ; col++ ) {  // bottom edge
```

```
                t.print(FRAMECHAR);
            }
            t.println();
        }
    }
```