**Standard drawing (basic methods).** Up to this point, our input/output abstractions have focused exclusively on text strings. Now we introduce an abstraction for producing drawings as output. This library is easy to use and allows us to take advantage of a visual medium to cope with far more information than is possible with just text. As with standard input/output, our standard drawing abstraction is implemented in a library `StdDraw` that you can access by downloading the file `StdDraw.java` from the booksite into your working directory. Standard draw is very simple: we imagine an abstract drawing device capable of drawing lines and points on a two-dimensional canvas. The device is capable of responding to the commands to draw basic geometric shapes that our programs issue in the form of calls to static methods in `StdDraw`, including methods for drawing lines, points, text strings, circles, rectangles, and polygons. Like the methods for standard input and standard output, these methods are nearly self-documenting: `StdDraw.line()` draws a straight line segment connecting the point \((x_0, y_0)\) with the point \((x_1, y_1)\) whose coordinates are given as arguments. `StdDraw.point()` draws a spot centered on the point \((x, y)\) whose coordinates are given as arguments, and so forth, as illustrated in the diagrams at right. Geometric shapes can be filled (in black, by default). The default scale is the unit square (all coordinates are between 0 and 1). The standard implementation displays the canvas in a window on your computer’s screen, with black lines and points on a white background.
public class StdDraw

static void line(double x0, double y0, double x1, double y1)
static void point(double x, double y)
static void text(double x, double y, String s)
static void circle(double x, double y, double r)
static void filledCircle(double x, double y, double r)
static void ellipse(double x, double y, double rw, double rh)
static void filledEllipse(double x, double y, double rw, double rh)
static void square(double x, double y, double r)
static void filledSquare(double x, double y, double r)
static void rectangle(double x, double y, double rw, double rh)
static void filledRectangle(double x, double y, double rw, double rh)
static void polygon(double[] x, double[] y)
static void filledPolygon(double[] x, double[] y)

API for our library of static methods for standard drawing (drawing methods)

Standard drawing (control methods). The library also includes methods to change the scale and size of the canvas, the color and width of the lines, the text font, and the timing of drawing (for use in animation). As arguments for setPenColor() you can use one of the predefined colors BLACK, BLUE, CYAN, DARK_GRAY, GRAY, GREEN, LIGHT_GRAY, MAGENTA, ORANGE, PINK, RED, BOOK_RED, WHITE, and YELLOW that are defined as constants in StdDraw (so we refer to one of them with code like StdDraw.RED). The window also includes a menu option to save your drawing to a file, in a format suitable for publishing on the web.

public class StdDraw

static void setXscale(double x0, double x1) reset x range to (x0, x1)
static void setYscale(double y0, double y1) reset y range to (y0, y1)
static void setPenRadius(double r) set pen radius to r
static void setPenColor(Color c) set pen color to c
static void setFont(Font f) set text font to f
static void setCanvasSize(int w, int h) set canvas to w-by-h window
static void clear(Color c) clear the canvas; color it c
static void show(int dt) show all; pause dt milliseconds

API for our library of static methods for standard drawing (control methods)
IN THIS BOOK, we use StdDraw for data analysis and for creating visual representations of algorithms in operation. The table at on the opposite page indicates some possibilities; we will consider many more examples in the text and the exercises throughout the book. The library also supports animation—of course, this topic is treated primarily on the booksite.
<table>
<thead>
<tr>
<th>data</th>
<th>plot implementation (code fragment)</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>int N = 100; StdDraw.setXscale(0, N); StdDraw.setYscale(0, N<em>N); StdDraw.setPenRadius(.01); for (int i = 1; i &lt;= N; i++) { StdDraw.point(i, i); StdDraw.point(i, i</em>i); StdDraw.point(i, i*Math.log(i)); }</td>
<td><img src="image1.png" alt="Graph example" /></td>
<td></td>
</tr>
<tr>
<td>int N = 100; StdDraw.setXscale(0, N); StdDraw.setYscale(0, N<em>N); StdDraw.setPenRadius(.01); for (int i = 1; i &lt;= N; i++) { StdDraw.point(i, i); StdDraw.point(i, i</em>i); StdDraw.point(i, i*Math.log(i)); }</td>
<td><img src="image2.png" alt="Graph example" /></td>
<td></td>
</tr>
<tr>
<td>int N = 50; double[] a = new double[N]; for (int i = 0; i &lt; N; i++) a[i] = StdRandom.random(); for (int i = 0; i &lt; N; i++) { double x = 1.0*i/N; double y = a[i]/2.0; double rw = 0.5/N; double rh = a[i]/2.0; StdDraw.filledRectangle(x, y, rw, rh); }</td>
<td><img src="image3.png" alt="Graph example" /></td>
<td></td>
</tr>
<tr>
<td>int N = 50; double[] a = new double[N]; for (int i = 0; i &lt; N; i++) a[i] = StdRandom.random(); Arrays.sort(a); for (int i = 0; i &lt; N; i++) { double x = 1.0*i/N; double y = a[i]/2.0; double rw = 0.5/N; double rh = a[i]/2.0; StdDraw.filledRectangle(x, y, rw, rh); }</td>
<td><img src="image4.png" alt="Graph example" /></td>
<td></td>
</tr>
</tbody>
</table>

**StdDraw plotting examples**