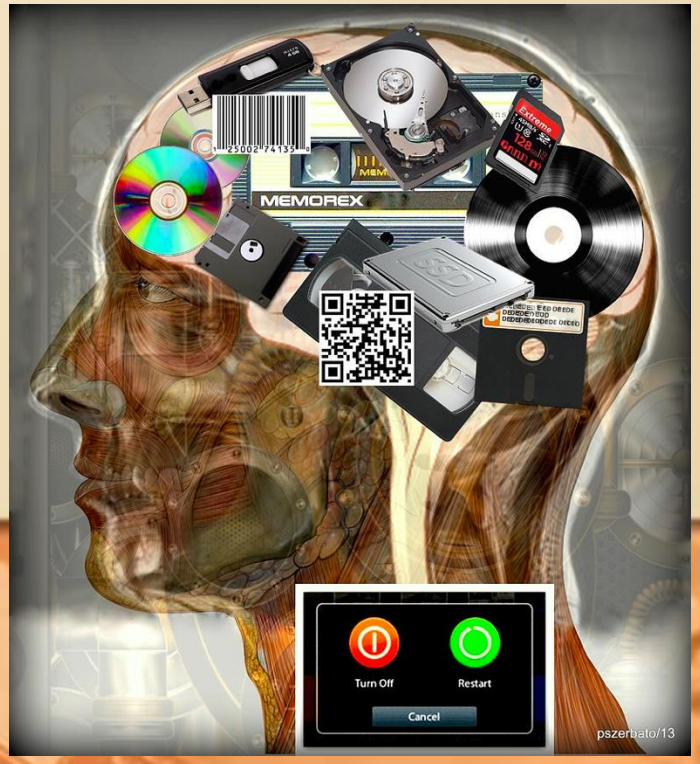
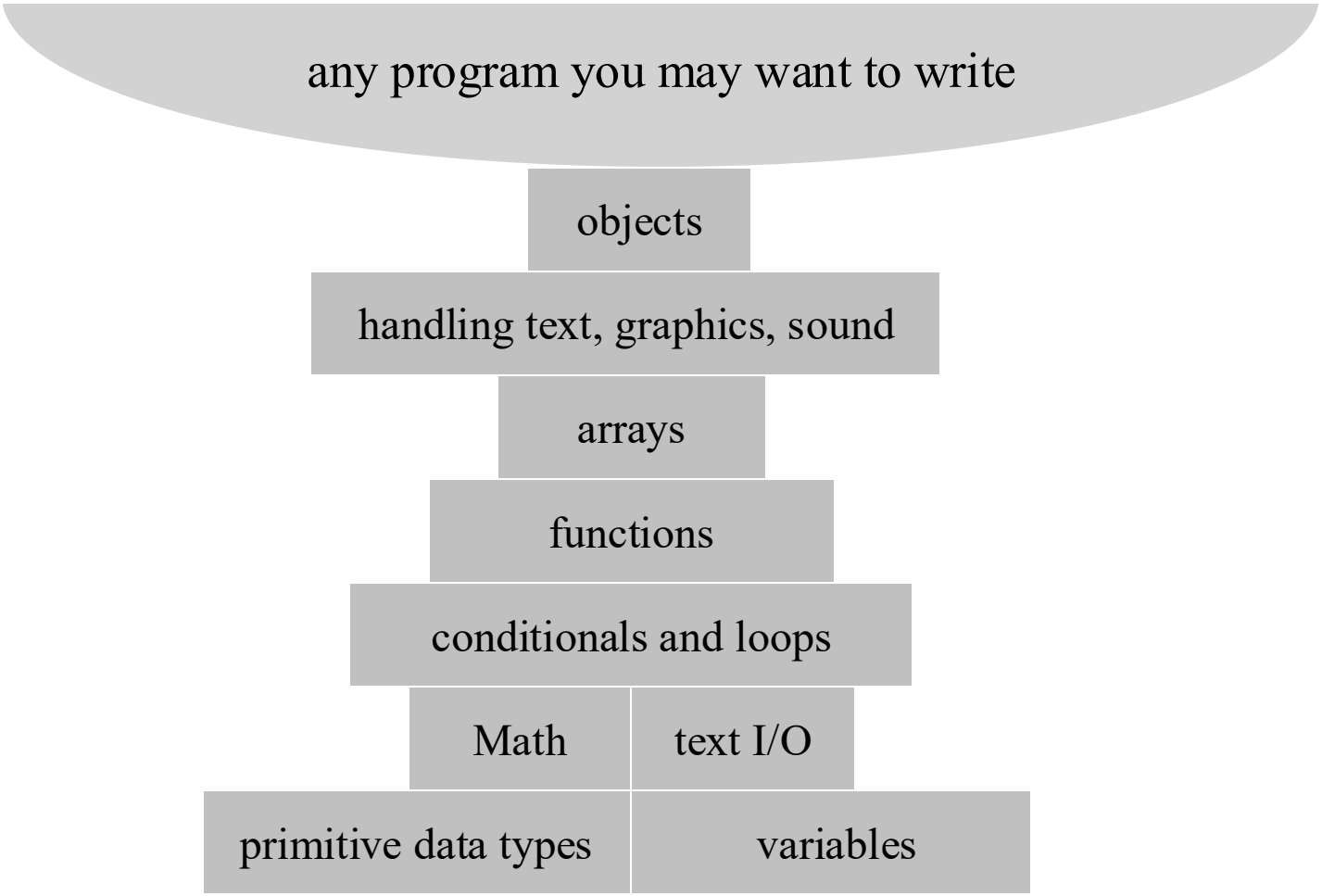
Lecture 6-2

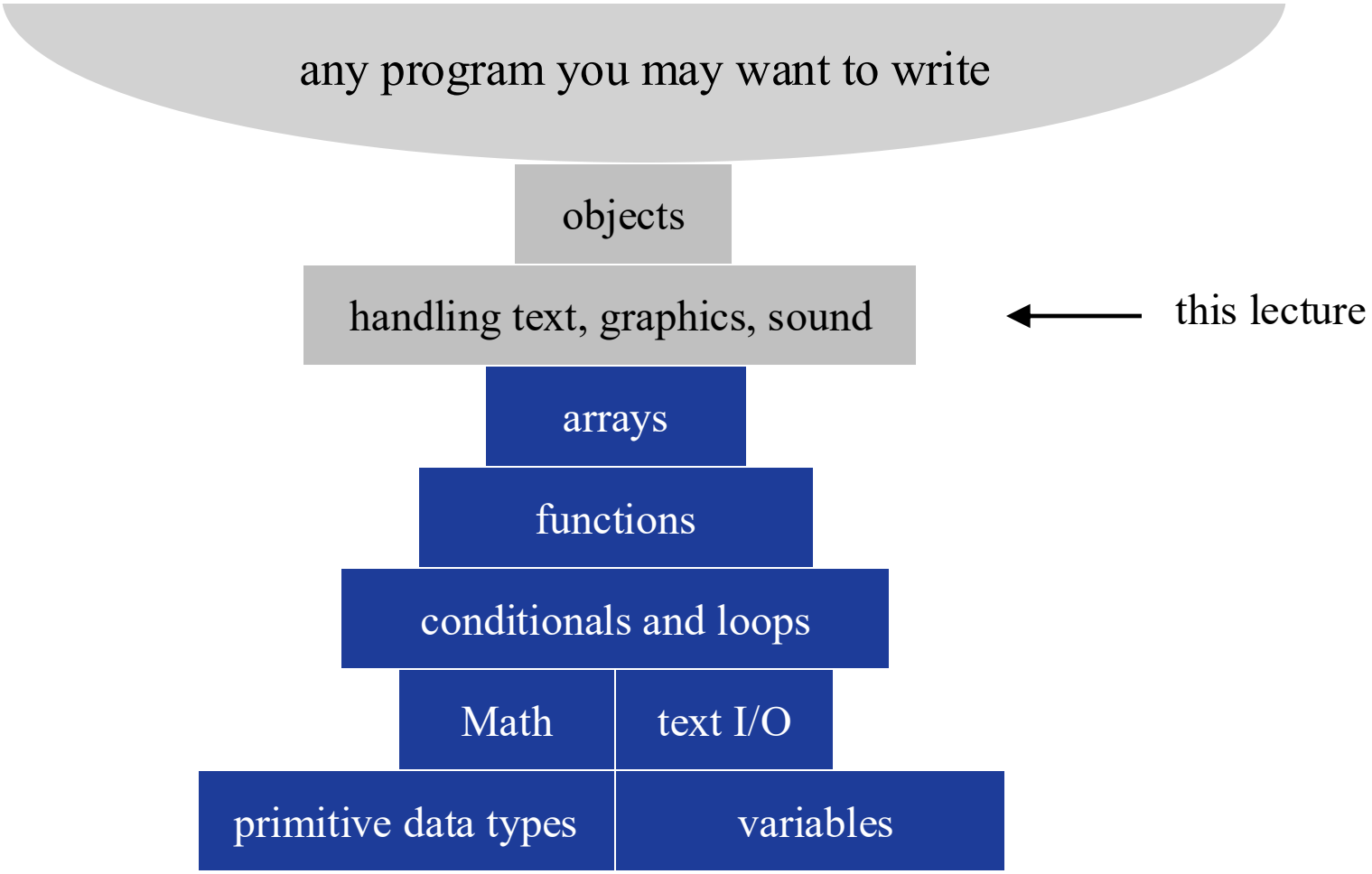
Multimedia, Part I



The big picture



The big picture



Typical I/O (Input/ Output) devices

Some input devices:


Keyboard


Mouse


Microphone


Camera


Mass storage


Network

Some output devices:


Display


Printer


Speakers

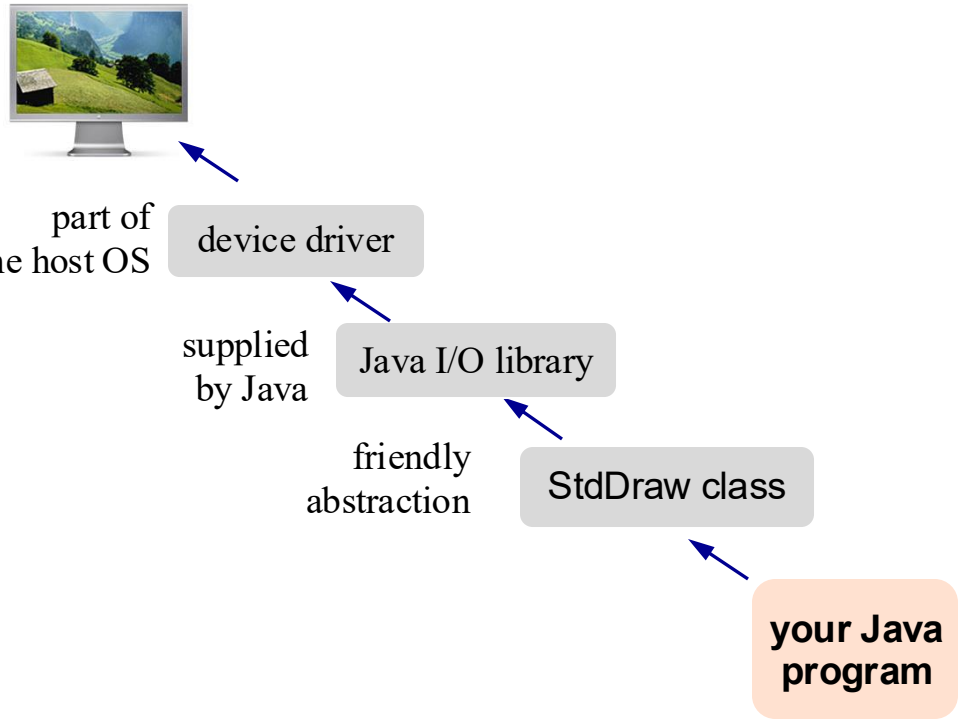

Mass storage


Network

Goal: write programs that create and process text, graphics, animation, and sound

How: using various language extensions, implemented by *libraries*.

The big picture



The big picture

part of the host OS

device driver

supplied by Java

Java I/O library

friendly abstraction

StdDraw class

In class

Java I/O library

device driver

your Java program

StdAudio class

In class

Java I/O library

device driver

Java I/O library

device driver

Same basic software hierarchy for many I/O devices

files

The big picture

part of the host OS

device driver

supplied by Java

Java I/O library

device driver

Java I/O library

StdDraw class

In class

your Java program

StdAudio class

In class

The programmer's view of I/O:

Interacting with these classes

Java I/O library

device driver

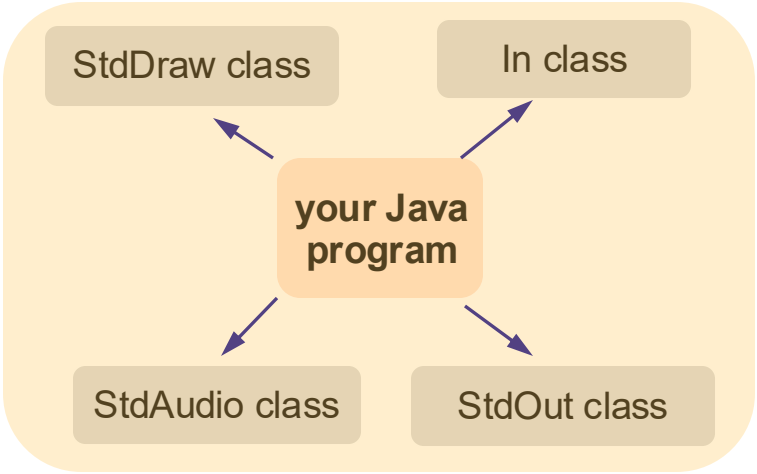
Java I/O library

device driver

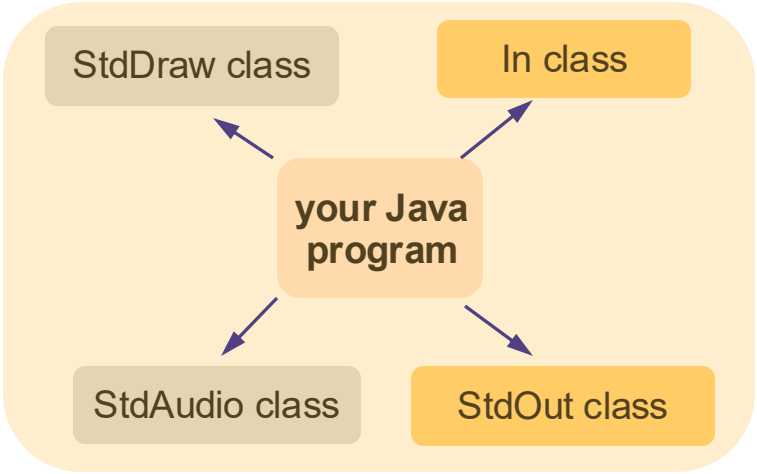
files

Same basic software hierarchy for many I/O devices

The big picture



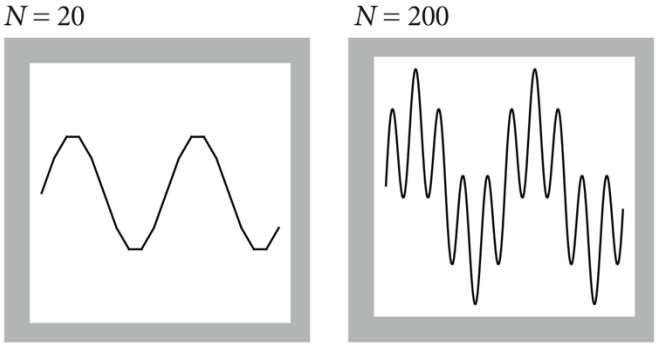
The big picture

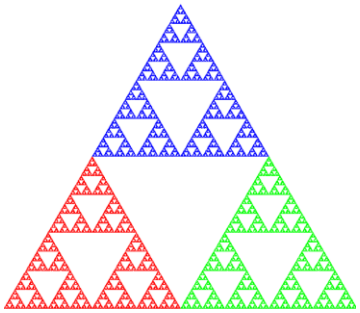


Lecture plan

- Graphics
- Animation
- Audio

Graphics

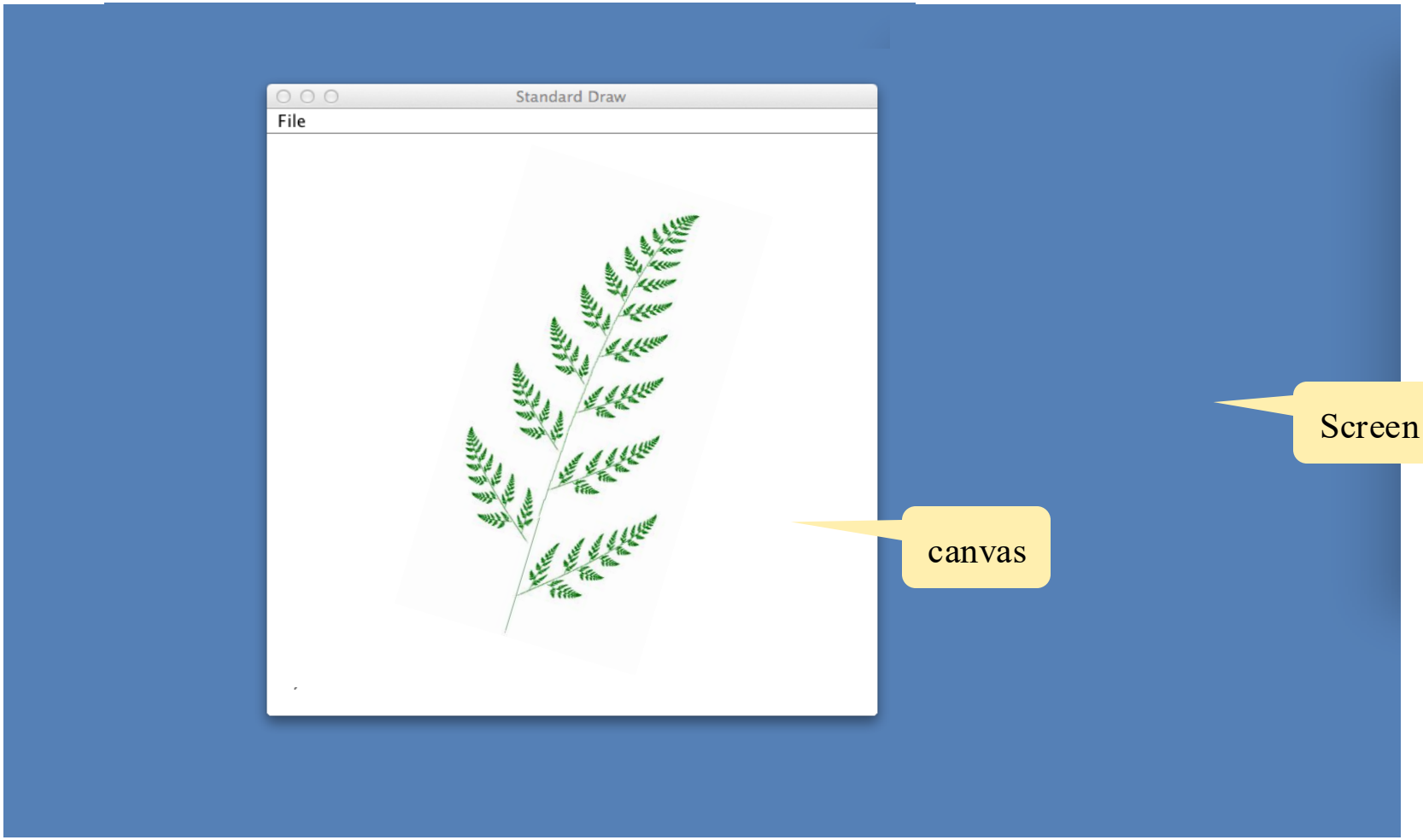




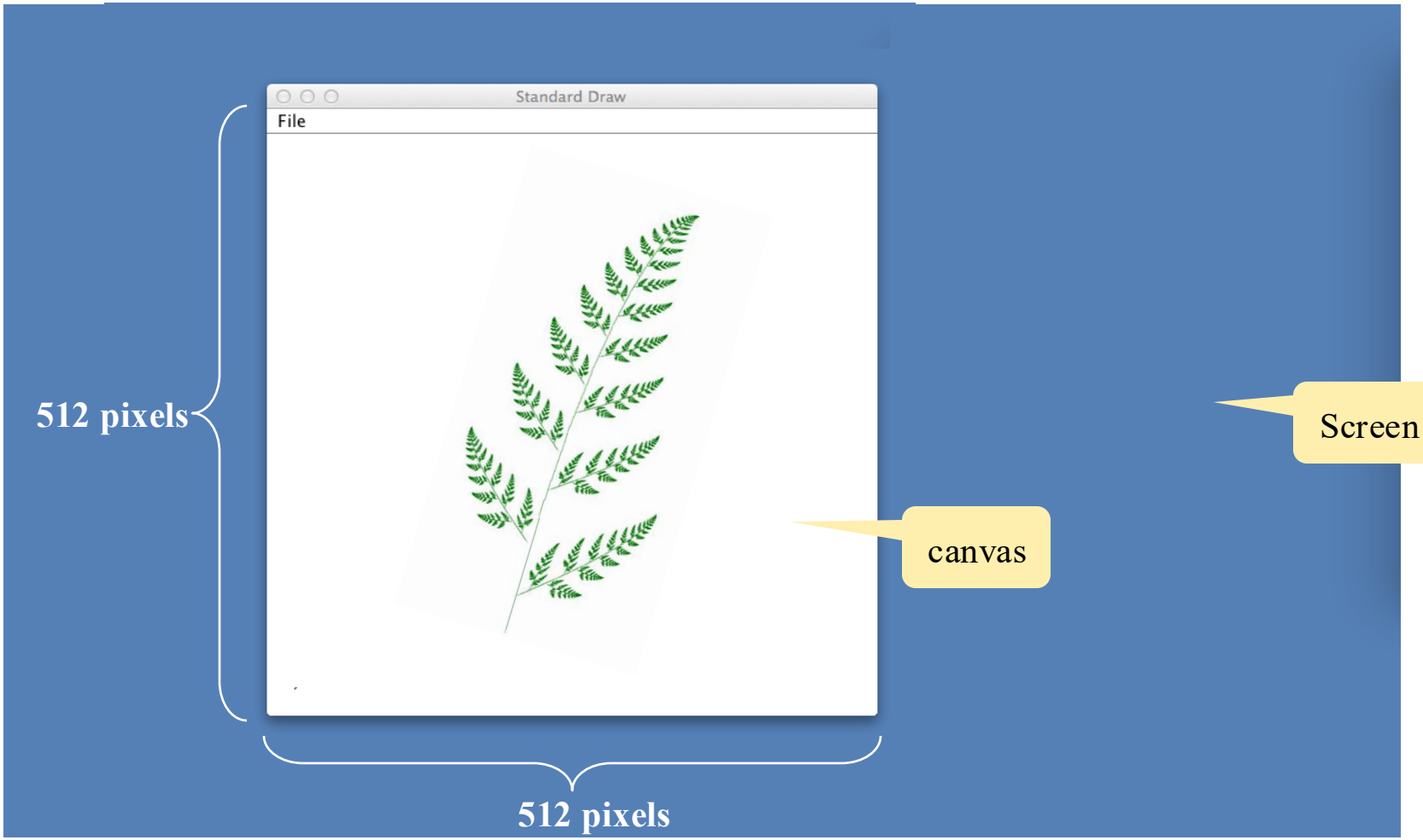
Computer graphics: The art and science of displaying and manipulating images

- The science: Building a mathematical model
- The art: Implementing the model using a library of graphics primitives:
  draw a *point*, draw a *line*, draw a *circle*, etc.
- `StdDraw`: An open source graphics class, used in this course.

“Canvas” (drawing area)

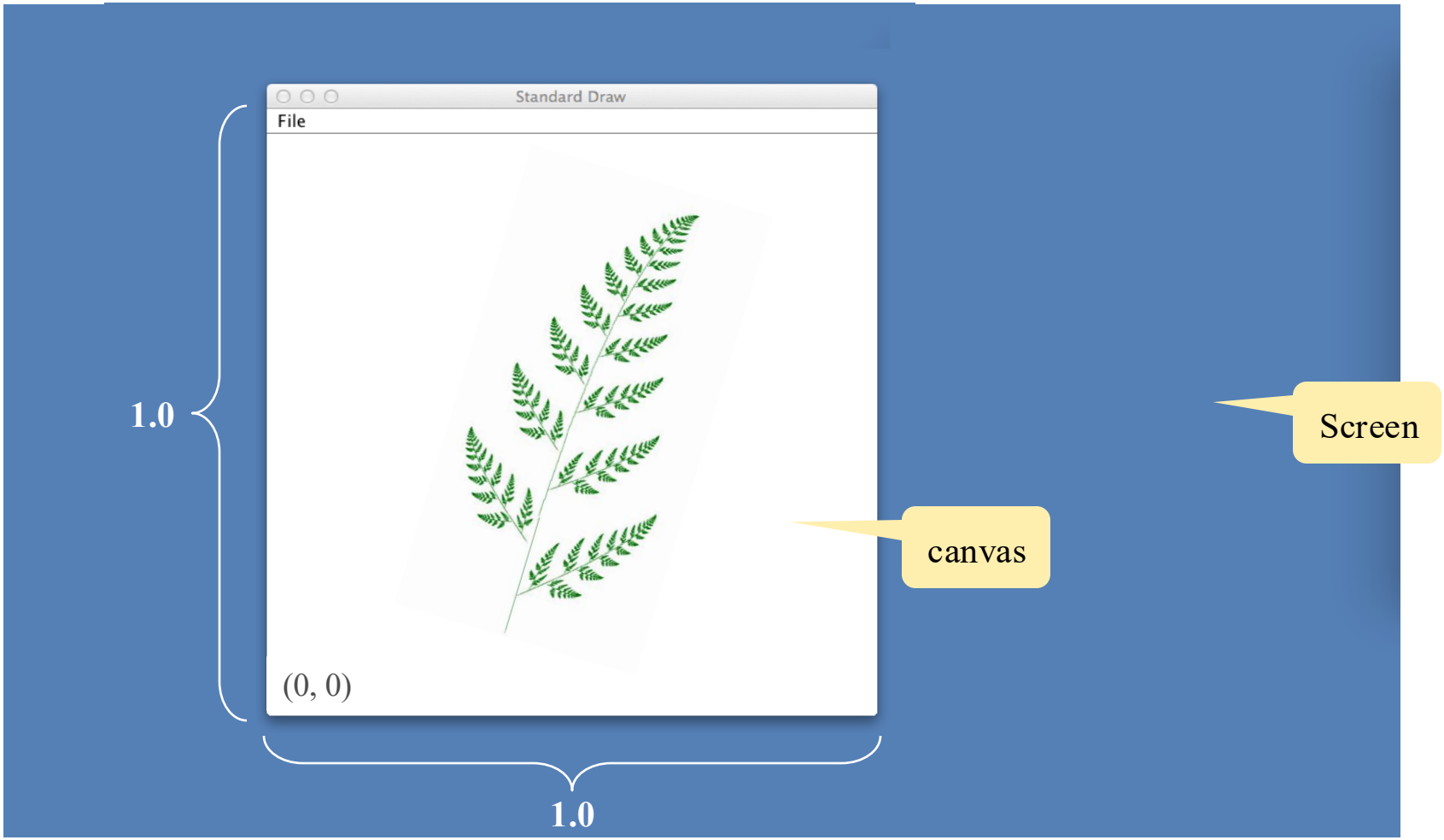


“Canvas” (drawing area)



- Physical view: a ”canvas” consisiting (by default) of 512 rows of 512 pixels each, which includes an invisible 10-pixel all around “frame”

“Canvas” (drawing area)



- Physical view: a ”canvas” consisiting (by default) of 512 rows of 512 pixels each, which includes an invisible 10-pixel all around “frame”
- Logical view: a 1.0 by 1.0 coordinate system, (0,0) at the bottom-left corner.

Line drawing

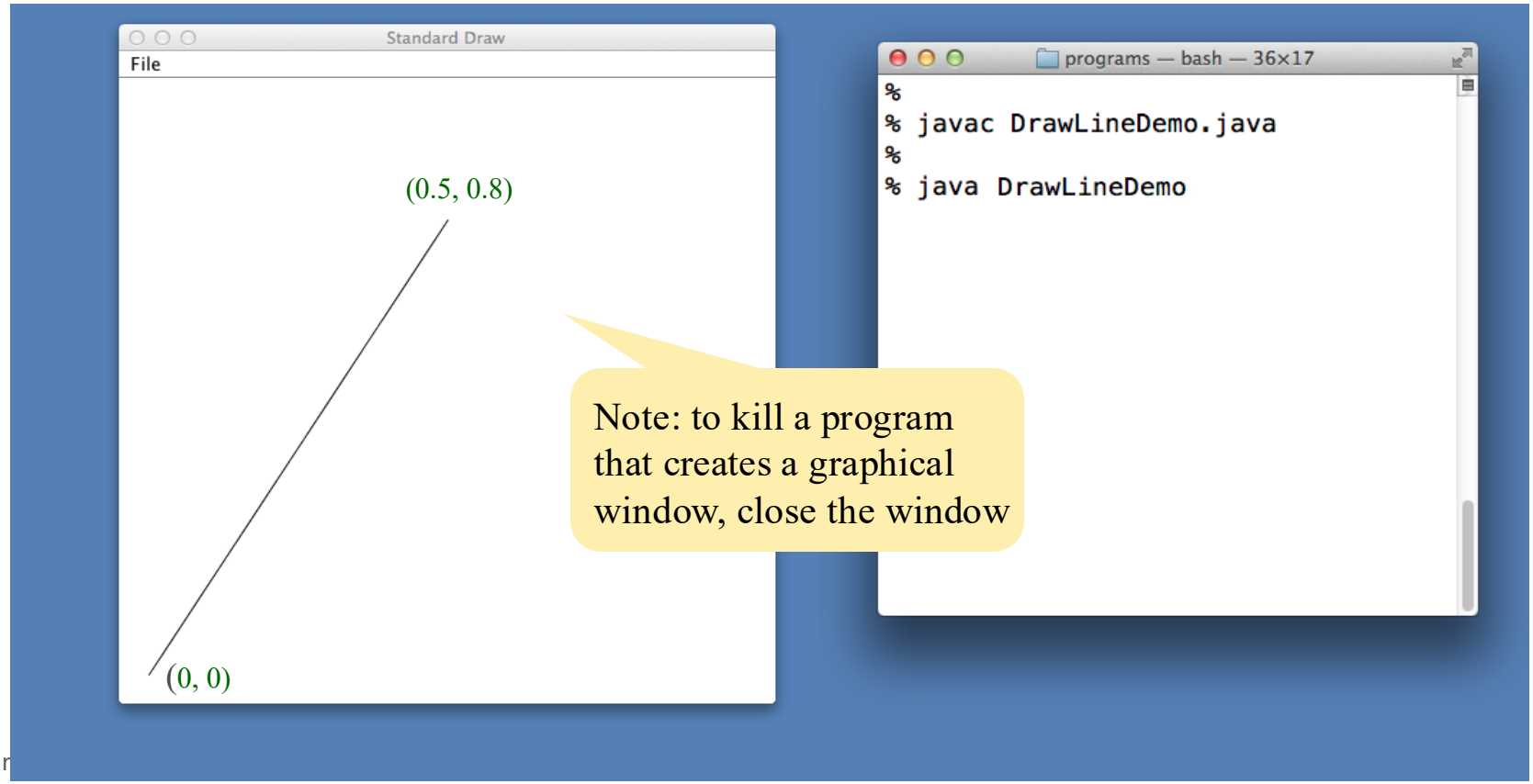
```
public class DrawLineDemo {
   public static void main(String[] args) {
      // draws a line between (0,0) and (0.5,0.8)
      StdDraw.line(0.0, 0.0, 0.5, 0.8);
   }
}
```

Remember to compile `StdDraw.java`!

Notes:

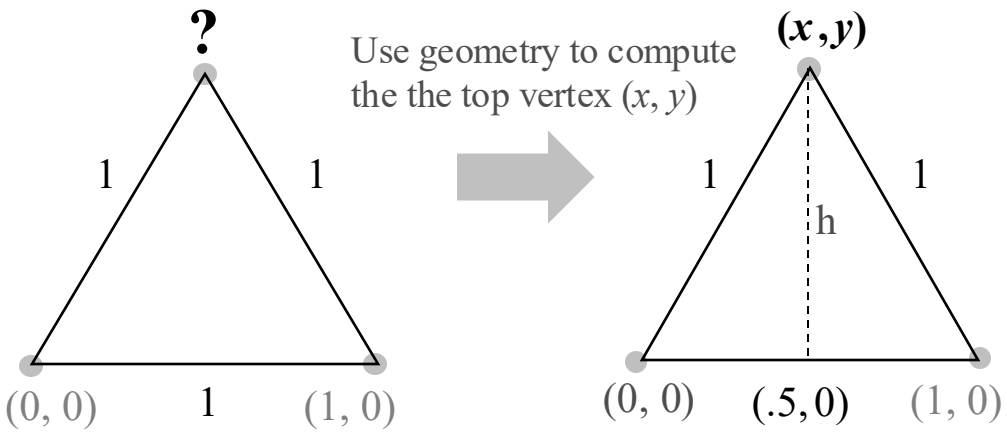
`StdDraw` creates a default canvass

The default canvas dimensions, line width, colors, etc. can be easily changed using `StdDraw` functions.



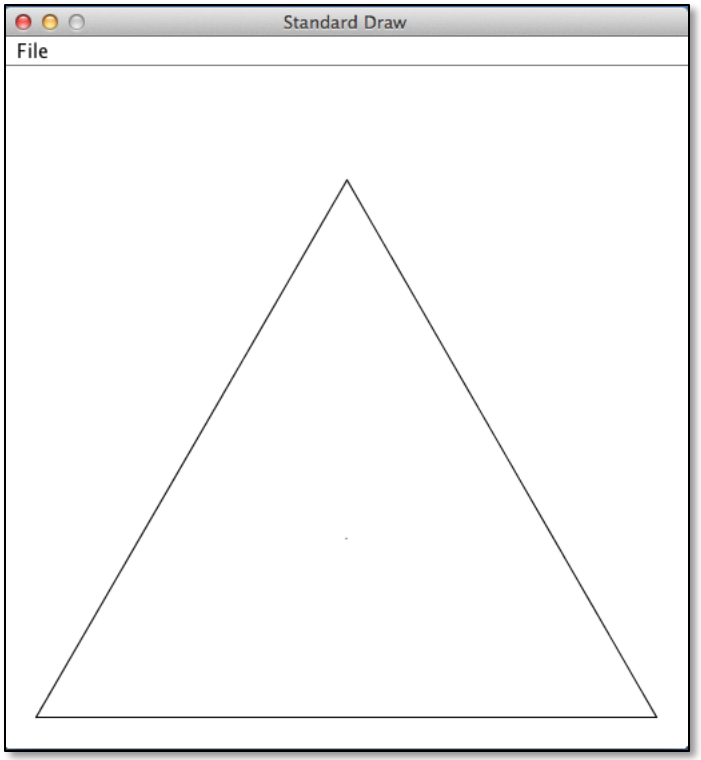
Line drawing

Task: Draw an equilateral triangle with side length 1; Left bottom corner at (0,0).




```
% java EqTriangle
%
```

```
public class EqTriangle {
   public static void main(String[] args) {
      //// You do it, using StdDraw.drawLine()
```



The `StdDraw` class

StdDraw: a library for drawing graphics

```
void line(double x0, double y0, double x1, double y1)
void point(double x, double y)
void text(double x, double y, String s)
void circle(double x, double y, double r)
void filledCircle(double x, double y, double r)
void square(double x, double y, double r)
void filledSquare(double x, double y, double r)
void polygon(double[] x, double[] y)
void filledPolygon(double[] x, double[] y)
```

Complete API (click)

`void setXscale(double x0, double x1)`	*reset x range to* (x_0, x_1)
`void setYscale(double y0, double y1)`	*reset y range to* (y_0, y_1)
`void setPenRadius(double r)`	*set pen radius to* `r`
`void setPenColor(Color c)`	*set pen color to* `c`
`void setFont(Font f)`	*set text font to* `f`
`void setCanvasSize(int w, int h)`	*set canvas to* `w`*-by-*`h` *window*
`void clear(Color c)`	*clear the canvas; color it* `c`
`void show(int dt)`	*show all; pause* `dt` *milliseconds*
`void save(String filename)`	*save to a* `.jpg` *or* `w.png` *file*

Note: Methods with the same names but no arguments reset to default values.

Data visualization

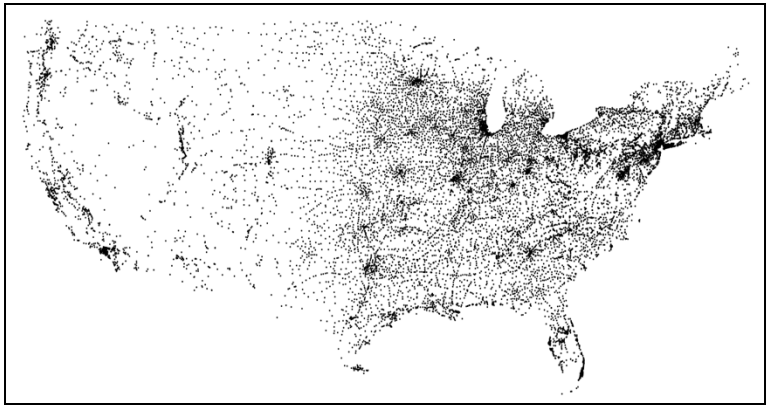
Given: Geographical coordinates of 13000+ cities and villages (with population > 500)

Task: Make the data visible

convention: first 4 values are the data's *xMin*, *yMin*, *xMax*, *yMax*

```
% more USA.txt
669905.0 247205.0 1244962.0 700000.0
 1097038.8890   245552.7780
 1103961.1110   247133.3330
 1104677.7780   247205.5560
 1108586.1110   249238.8890
 ...
 % java PlotMap USA.txt
```

1200 by 800 pixels canvas



Data visualization

Given: Geographical coordinates of 13000+ cities and villages (with population > 500)

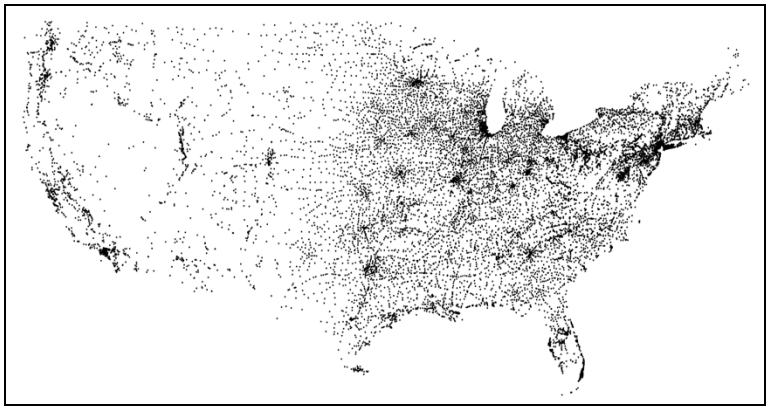
Task: Make the data visible

```
/** Reads data points (geographical coordinates) from a file,
 *  and draws them. */
// Uses classes In, StdDraw
public class PlotMap {
  public static void main(String[] args) {
    In in = new In(args[0]); // Input file reader
    // Sets the canvass physical dimensions to
    // a fixed "landscape" frame
    StdDraw.setCanvasSize(1000,800); // (width,height)
    // Scales the canvass logical dimensions according
    // to the min and max values in the data
    double xMin = in.readDouble();
    double yMin = in.readDouble();
    double xMax = in.readDouble();
    double yMax = in.readDouble();
    StdDraw.setXscale(xMin, xMax);
    StdDraw.setYscale(yMin, yMax);

    // Draws the data points
    while (!in.isEmpty()) {
        StdDraw.point(in.readDouble(),
                      in.readDouble());
    }
  }
}
```

convention: first 4 values are the data's *xMin*, *yMin*, *xMax*, *yMax*

```
% more USA.txt
669905.0 247205.0 1244962.0 700000.0
 1097038.8890   245552.7780
 1103961.1110   247133.3330
 1104677.7780   247205.5560
 1108586.1110   249238.8890
 ...
 % java PlotMap USA.txt
```

1200 by 800 pixels canvas



Data visualization

Given: Geographical coordinates

Task: Make the data visible

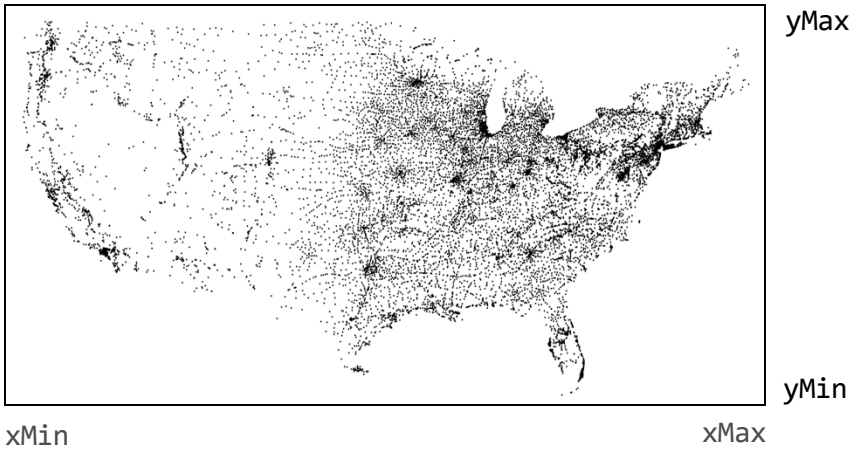
```
/** Reads data points (geographical coordinates) from a file,
 *  and draws them. */

// Uses classes In, StdDraw

public class PlotMap {
  public static void main(String[] args) {

    In in = new In(args[0]); // Input file reader

    // Sets the canvass physical dimensions to
    // a fixed "landscape" frame
    StdDraw.setCanvasSize(1000,800); // (width,height)

    // Scales the canvass logical dimensions according
    // to the min and max values in the data
    double xMin = in.readDouble();
    double yMin = in.readDouble();
    double xMax = in.readDouble();
    double yMax = in.readDouble();
    StdDraw.setXscale(xMin, xMax);
    StdDraw.setYscale(yMin, yMax);

    // Draws the data points
    while (!in.isEmpty()) {
        StdDraw.point(in.readDouble(),
                      in.readDouble());
    }
  }
}
```

Improvements (self study)

1. Can we make a good choice for an aspect ratio, instead of the fixed 1000 by 800?
2. Should we trust the first 4 min-max values?
3. Can we draw the data points randomly, instead of by the order in which they appear in the file?

Tip: To investigate and implement, start by reading all the data points into an array.

1000 by 800 pixels canvas



Function plotting

Plot $f(x) = x^2$

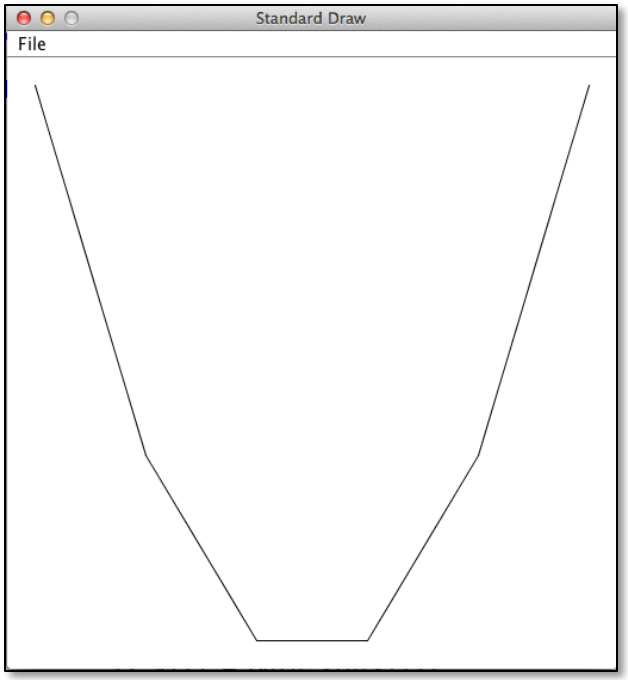
Function to draw: Hard-coded into the program

Range (xMin, xMax): command-line arguments

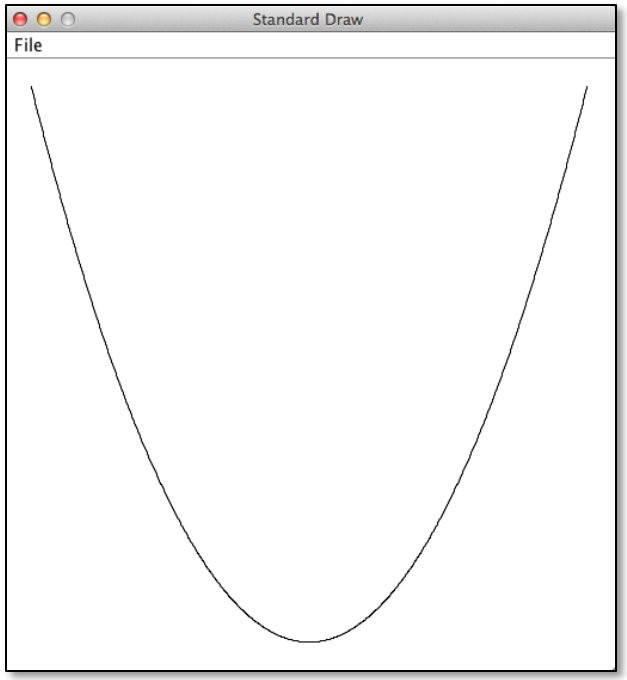
N (number of segments): command-line argument

xMin xMax N

```
% java PlotFunction -100 100 5
```

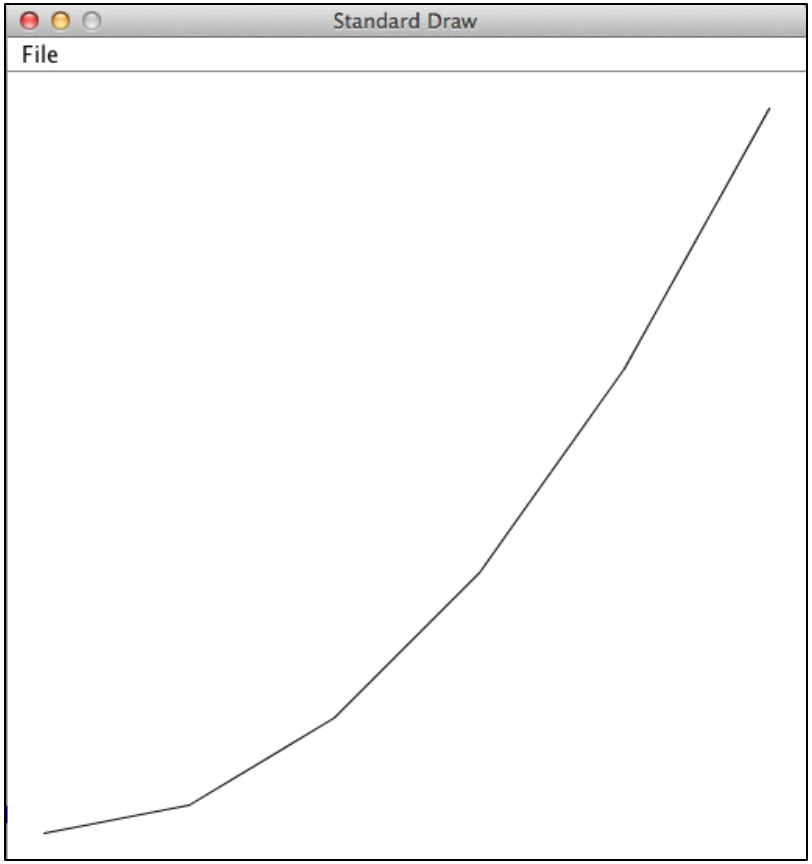



```
% java PlotFunction -100 100 1000
```



Function plotting

Plot $f(x) = x^2$

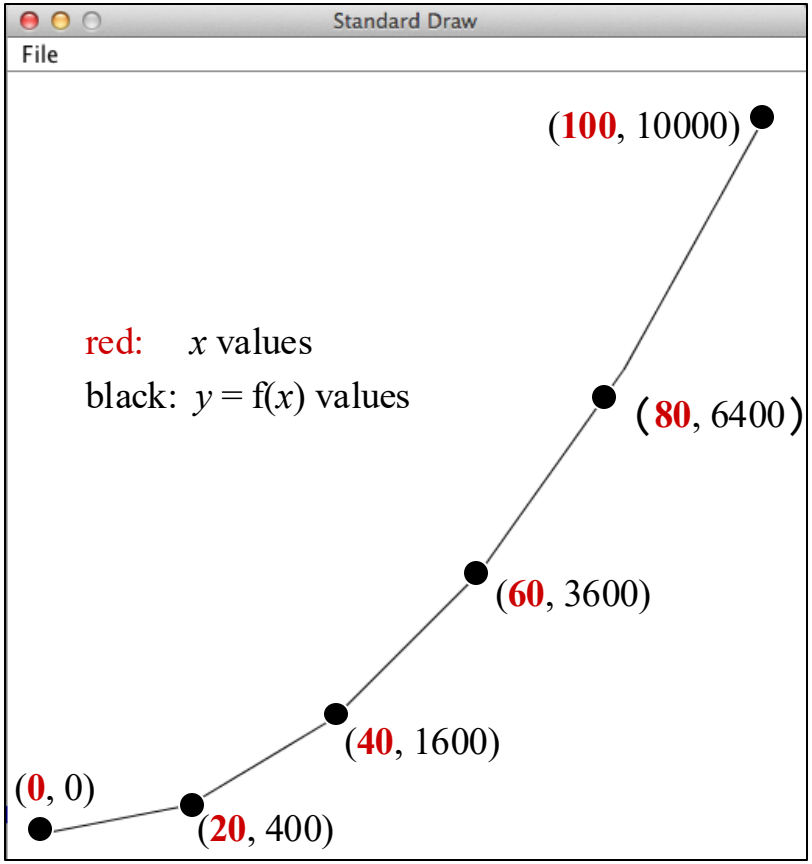


Function plotting

Plot $f(x) = x^2$

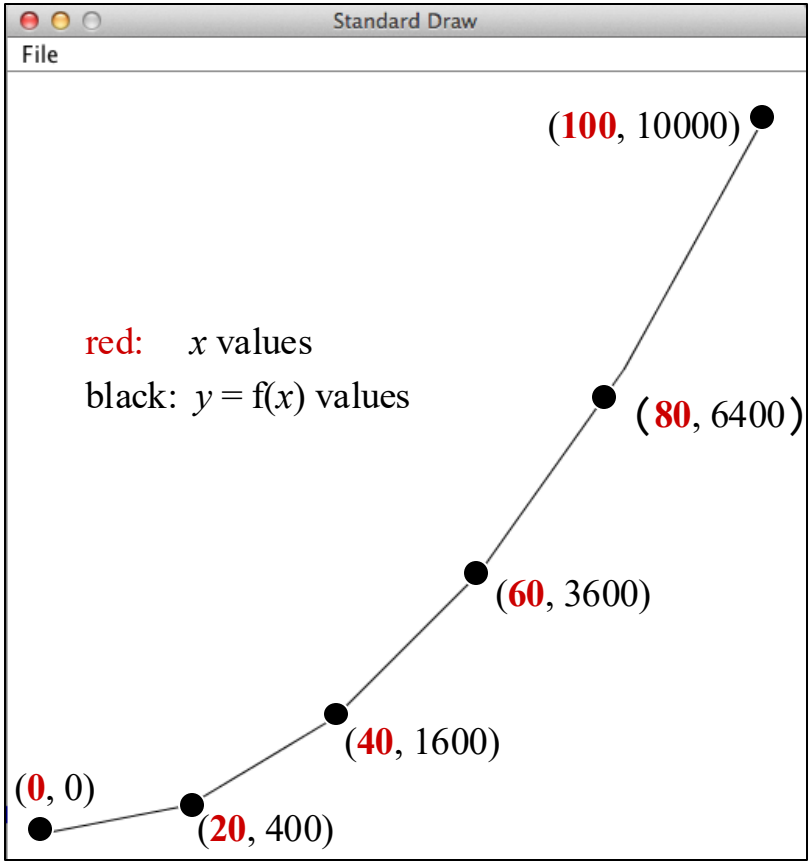
Algorithm

- Input: *xMin*, *xMax*, *N*
- Input (fixed in the program): f(x)
- Create an $N+1$ elements array containing the x values
- Create an $N+1$ elements array containing the f(x) values
- Use line drawing to connect the resulting $N+1$ (x, y) points.



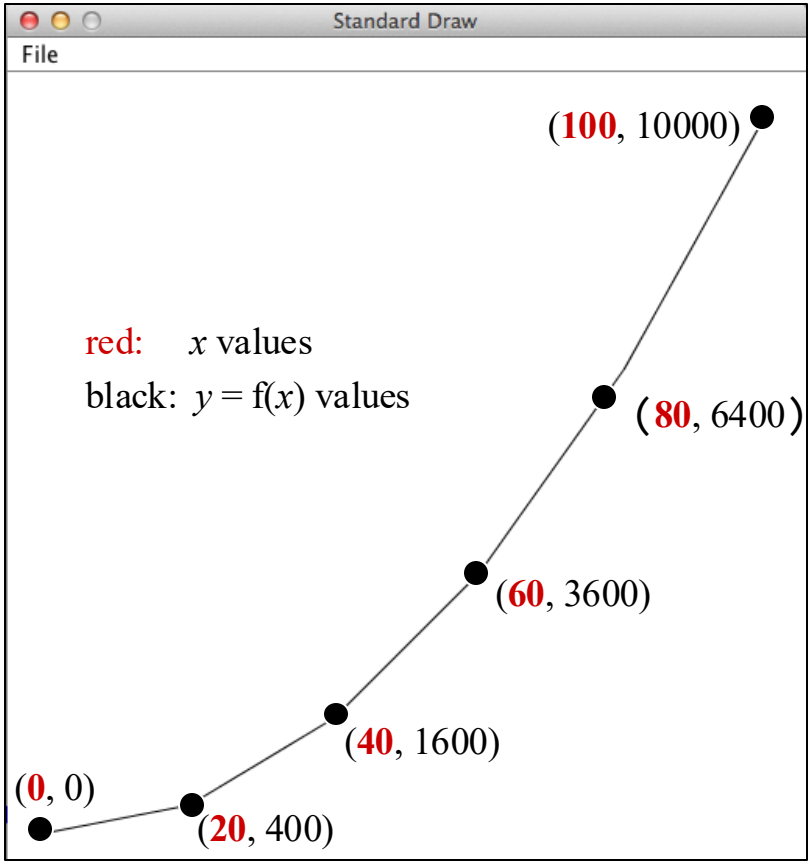
Function plotting

```
/** Plots functions. */
public class PlotFunction {

    // The function to plot: f(x) = x * x
    public static double f(double x) {
        return x * x;
    }

    public static void main(String[] args) {
        double xMin = Double.parseDouble(args[0]);
        double xMax = Double.parseDouble(args[1]);
        int N = Integer.parseInt(args[2]);

        // Creates arrays for the x values and the y values
        double x[] = xArr(xMin, xMax, N);
        double y[] = f(x);

        // Scales the canvas
        StdDraw.setXscale(xMin, xMax);
        StdDraw.setYscale(min(y), max(y));

        // Connects the (x,y) points
        for (int i = 0; i < N; i++) {
            StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
        }
    }
    //// More functions: Coming up
}
```



Function plotting

```
/** Plots functions. */
public class PlotFunction {

    // The function to plot: f(x) = x * x
    public static double f(double x) {
        return x * x;
    }

    public static void main(String[] args) {
        double xMin = Double.parseDouble(args[0]);
        double xMax = Double.parseDouble(args[1]);
        int N = Integer.parseInt(args[2]);

        // Creates arrays for the x values and the y values
        double x[] = xArr(xMin, xMax, N);
        double y[] = f(x);

        // Scales the canvas
        StdDraw.setXscale(xMin, xMax);
        StdDraw.setYscale(min(y), max(y));

        // Connects the (x,y) points
        for (int i = 0; i < N; i++) {
            StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
        }
    }
    //// More functions: Coming up
}
```



Function plotting

```
/** Plots functions. */
public class PlotFunction {

    // The function to plot: f(x) = x * x
    public static double f(double x) {
        return x * x;
    }

    public static void main(String[] args) {
        double xMin = Double.parseDouble(args[0]);
        double xMax = Double.parseDouble(args[1]);
        int N = Integer.parseInt(args[2]);

        // Creates arrays for the x values and y values
        double x[] = xArr(xMin, xMax, N);
        double y[] = f(x);

        // Scales the canvas
        StdDraw.setXscale(xMin, xMax);
        StdDraw.setYscale(min(y), max(y));

        // Connects the (x,y) points
        for (int i = 0; i < N; i++) {
            StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
        }
    }
    //// More functions: On the right of this slide
}
```

```
// Returns an array that represents the x-axis (x -ציר ה):
// N equally-spaced points between a and b
public static double[] xArr(double a, double b, int N) {
    double[] x = new double[N + 1];
    double interval = (b - a) / N;
    for (int i = 0; i <= N; i++) {
        x[i] = a + (i * interval);
    }
    return x;
}

// Returns the array f(x[])
public static double[] f(double x[]) {
    int N = x.length;
    double[] y = new double[N];
    for (int i = 0; i < N; i++) {
        y[i] = f(x[i]);   // computes y = f(x)
    }
    return y;
}

// Returns the minimum value in the given array (code omitted)
    public static double min(double arr[]) {
}

// Returns the maximum value in the given array (code omitted)
    public static double max(double arr[]) {
}
```

Function plotting

```
/** Plots functions. */
public class PlotFunction {

    // The function to plot: f(x) = x * x
    public static double f(double x) {
        return x * x;
    }

    public static void main(String[] args) {
        double xMin = Double.parseDouble(args[0]);
        double xMax = Double.parseDouble(args[1]);
        int N = Integer.parseInt(args[2]);

        // Creates arrays for the x values and y values
        double x[] = xArr(xMin, xMax, N);
        double y[] = f(x);

        // Scales the canvas
        StdDraw.setXscale(xMin, xMax);
        StdDraw.setYscale(min(y), max(y));

        // Connects the (x,y) points
        for (int i = 0; i < N; i++) {
            StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
        }
    }
    //// More functions: On the right of this slide
}
```

```
// Returns an array that represents the x-axis (x -ציר ה):
// N equally-spaced points between a and b
public static double[] xArr(double a, double b, int N) {
    double[] x = new double[N + 1];
    double interval = (b - a) / N;
    for (int i = 0; i <= N; i++) {
        x[i] = a + (i * interval);
    }
    return x;
}

// Returns the array f(x[])
public static double[] f(double x[]) {
    int N = x.length;
    double[] y = new double[N];
    for (int i = 0; i < N; i++) {
        y[i] = f(x[i]);   // computes y = f(x)
    }
    return y;
}

// Returns the minimum value in the given array (code omitted)
    public static double min(double arr[]) {
}

// Returns the maximum value in the given array (code omitted)
    public static double max(double arr[]) {
}
```

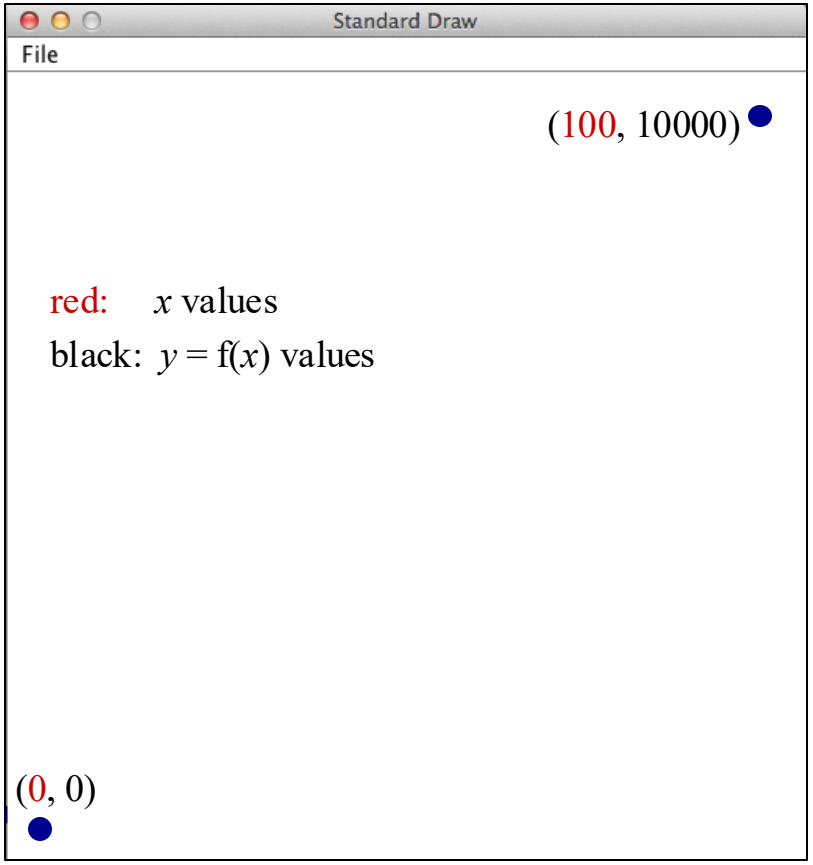
Function plotting

```
/** Plots functions. */
public class PlotFunction {

    // The function to plot: f(x) = x * x
    public static double f(double x) {
        return x * x;
    }

    public static void main(String[] args) {
        double xMin = Double.parseDouble(args[0]);
        double xMax = Double.parseDouble(args[1]);
        int N = Integer.parseInt(args[2]);

        // Creates arrays for the x values and y values
        double x[] = xArr(xMin, xMax, N);
        double y[] = f(x);

        // Scales the canvas
        StdDraw.setXscale(xMin, xMax);
        StdDraw.setYscale(min(y), max(y));

        // Connects the (x,y) points
        for (int i = 0; i < N; i++) {
            StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
        }
    }
    //// More functions: On the right of this slide
}
```

```
// Returns an array that represents the x-axis (x -ציר ה):
// N equally-spaced points between a and b
public static double[] xArr(double a, double b, int N) {
    double[] x = new double[N + 1];
    double interval = (b - a) / N;
    for (int i = 0; i <= N; i++) {
        x[i] = a + (i * interval);
    }
    return x;
}

// Returns the array f(x[])
public static double[] f(double x[]) {
    int N = x.length;
    double[] y = new double[N];
    for (int i = 0; i < N; i++) {
        y[i] = f(x[i]);   // computes y = f(x)
    }
    return y;
}

// Returns the minimum value in the given array (code omitted)
    public static double min(double arr[]) {
}

// Returns the maximum value in the given array (code omitted)
    public static double max(double arr[]) {
}
```

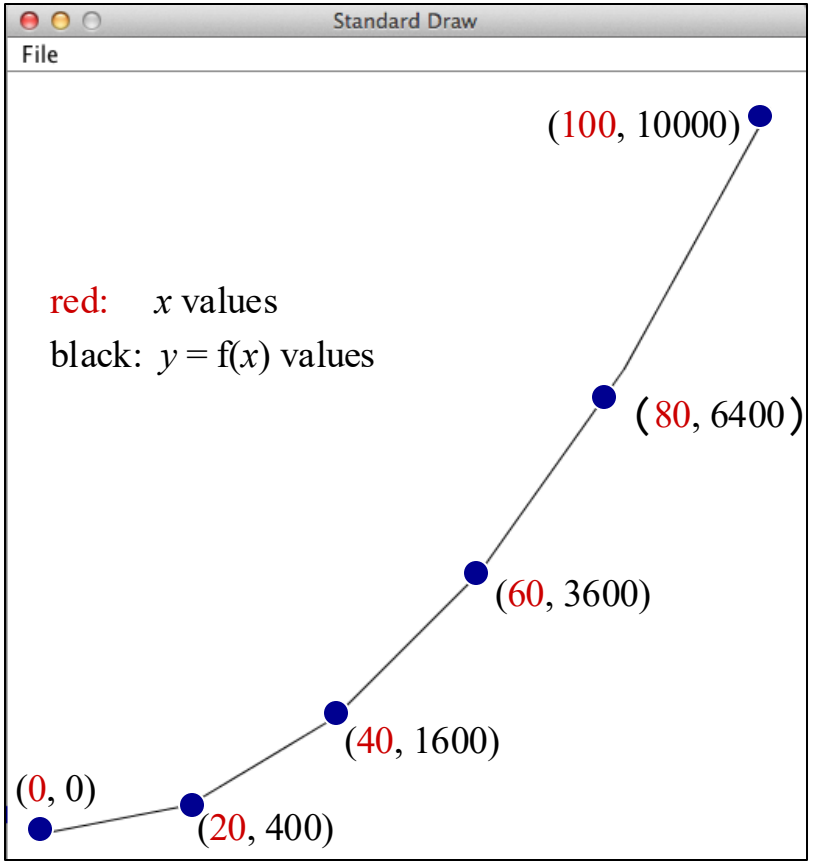
Function plotting

```
/** Plots functions. */
public class PlotFunction {

    // The function to plot: f(x) = x * x
    public static double f(double x) {
        return x * x;
    }

    public static void main(String[] args) {
        double xMin = Double.parseDouble(args[0]);
        double xMax = Double.parseDouble(args[1]);
        int N = Integer.parseInt(args[2]);

        // Creates arrays for the x values and y values
        double x[] = xArr(xMin, xMax, N);
        double y[] = f(x);

        // Scales the canvas
        StdDraw.setXscale(xMin, xMax);
        StdDraw.setYscale(min(y), max(y));

        // Connects the (x,y) points
        for (int i = 0; i < N; i++) {
            StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
        }
    }
    //// More functions: On the right of this slide
}
```



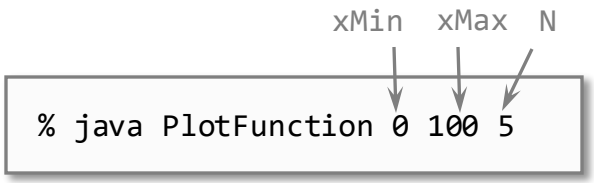
Function plotting

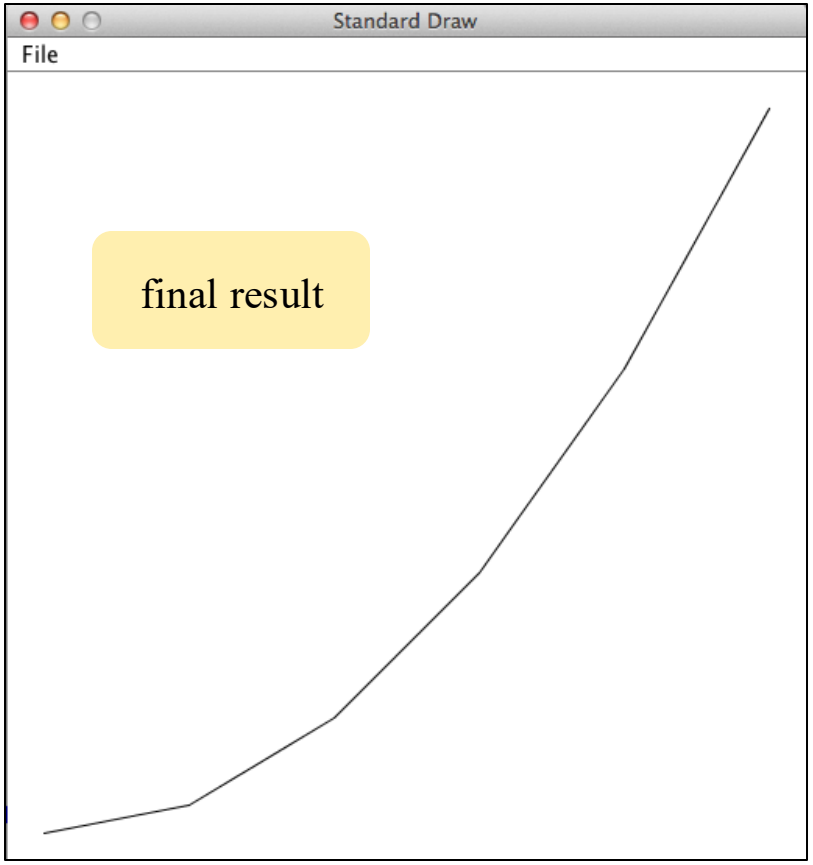
```
/** Plots functions. */
public class PlotFunction {

    // The function to plot: f(x) = x * x
    public static double f(double x) {
        return x * x;
    }

    public static void main(String[] args) {
        double xMin = Double.parseDouble(args[0]);
        double xMax = Double.parseDouble(args[1]);
        int N = Integer.parseInt(args[2]);

        // Creates arrays for the x values and y values
        double x[] = xArr(xMin, xMax, N);
        double y[] = f(x);

        // Scales the canvas
        StdDraw.setXscale(xMin, xMax);
        StdDraw.setYscale(min(y), max(y));

        // Connects the (x,y) points
        for (int i = 0; i < N; i++) {
            StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
        }
    }
    //// More functions: On the right of this slide
}
```



Function plotting

```
/** Plots functions. */
public class PlotFunction {

    // The function to plot: f(x) = x * x
    public static double f(double x) {
        return x * x;
    }

    public static void main(String[] args) {
        double xMin = Double.parseDouble(args[0]);
        double xMax = Double.parseDouble(args[1]);
        int N = Integer.parseInt(args[2]);

        // Creates arrays for the x values and y values
        double x[] = xArr(xMin, xMax, N);
        double y[] = f(x);

        // Scales the canvas
        StdDraw.setXscale(xMin, xMax);
        StdDraw.setYscale(min(y), max(y));

        // Connects the (x,y) points
        for (int i = 0; i < N; i++) {
            StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
        }
    }
    //// More functions: On the right of this slide
}
```

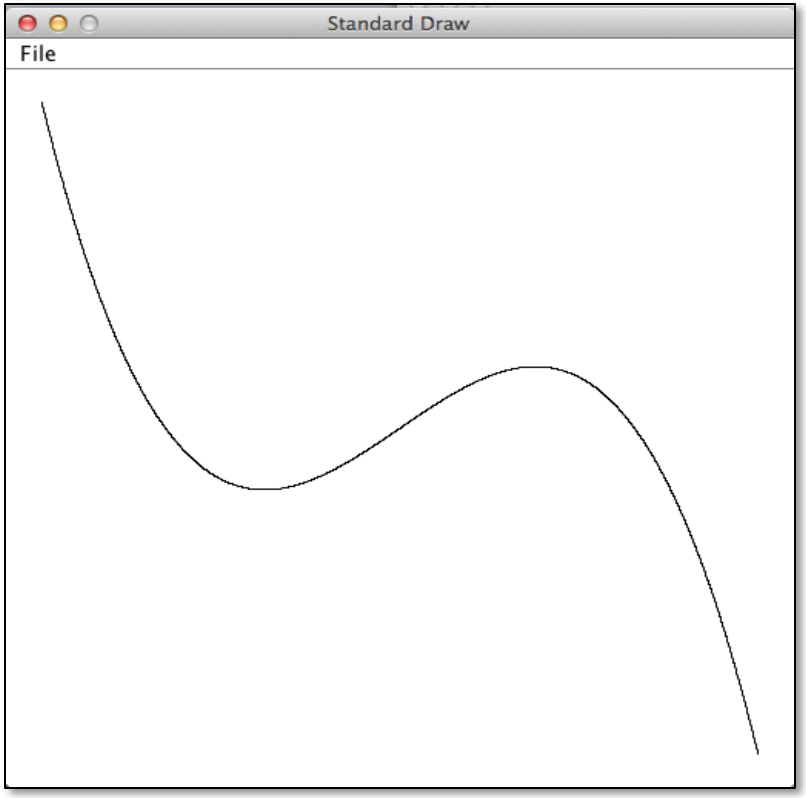




Function plotting

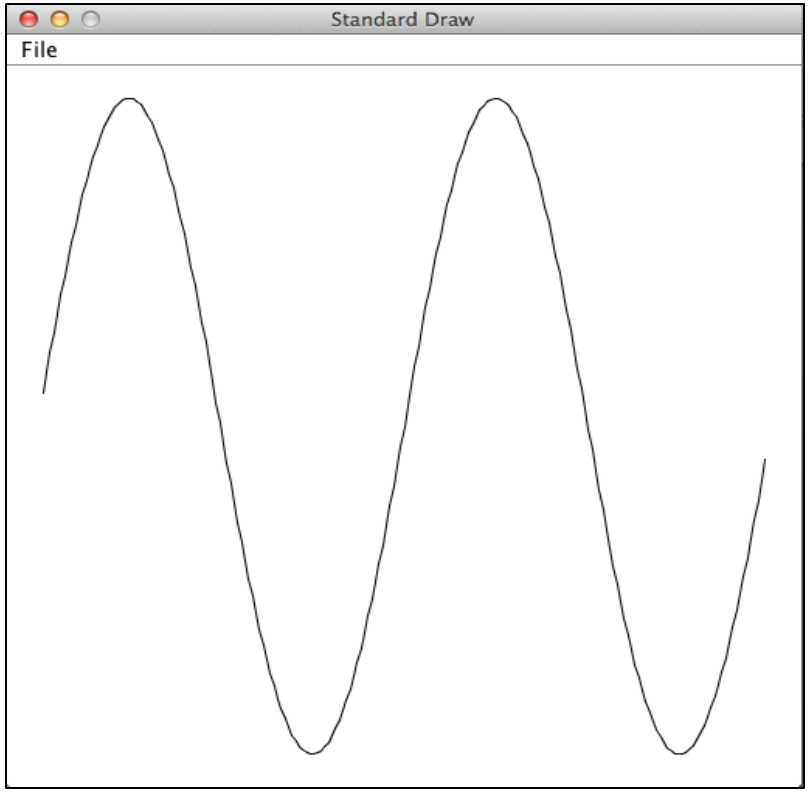
Plot $f(x) = -7x^3 + 3x + 2$

```
% java PlotFunction -2 2 100
```



Plot $f(x) = sin(x)$ in the range $(-2\pi, +2\pi)$

```
% java PlotFunction -6.28 6.28 100
```

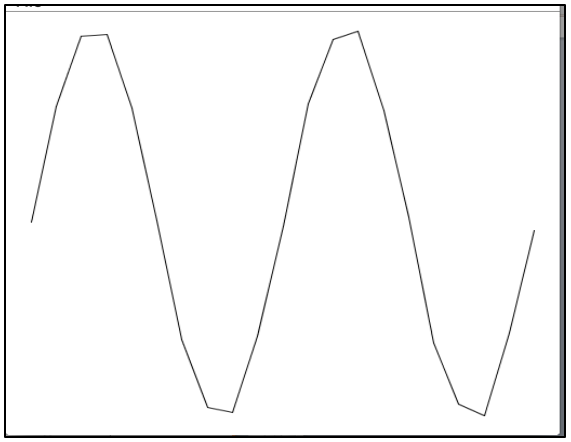


Function plotting

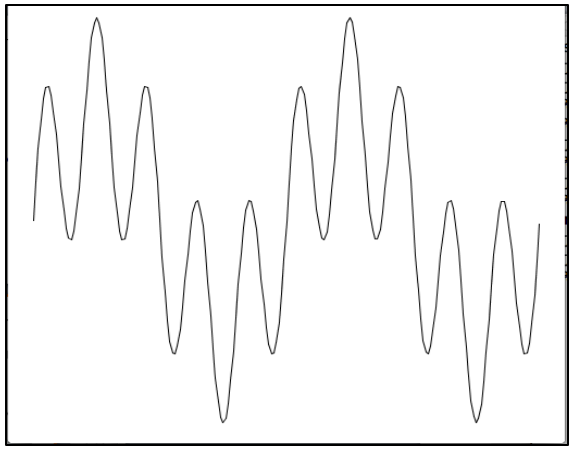
Plot $f(x) = \sin(4x) + \sin(20x), x \in [0, \pi]$

```
public class PlotFunction {
    // The function to plot: f(x) = sin(4x) + sin(20x)
    public static double f(double x) {
        return Math.sin(4*x) + Math.sin(20*x);
    }
    public static void main(String[] args) {
    ...
```

```
% java PlotFunction 0 3.14 20
```



```
% java PlotFunction 0 3.14 200
```



One lesson...

visual imaging of data (like linear interpolation) can be tricky;

when shown a data-driven visual, always demand to see the data.