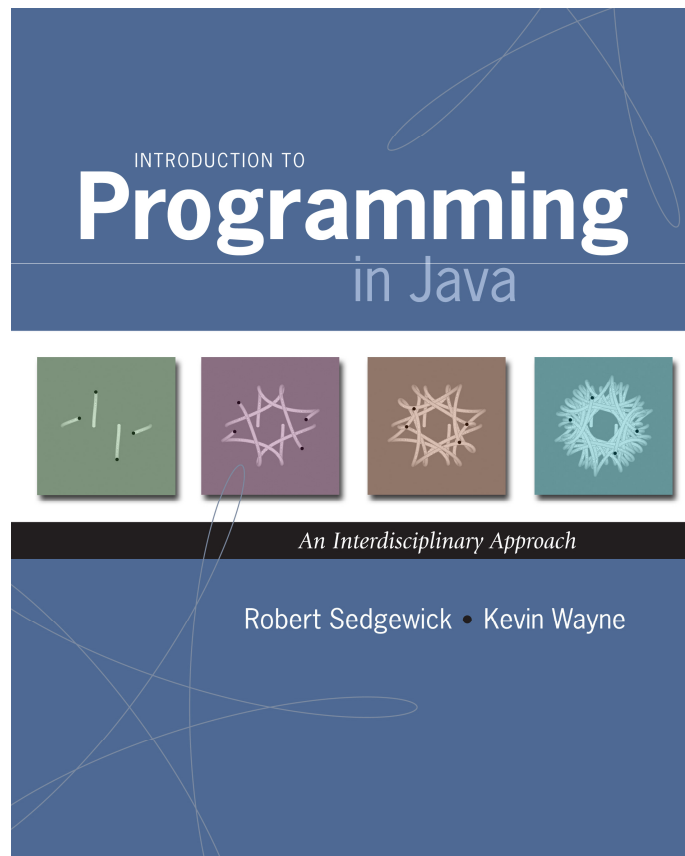


1.5 Input and Output



Input and Output

Input devices.



Keyboard



Mouse



Hard drive



Network



Digital camera



Microphone

Output devices.



Display



Speakers



Hard drive



Network



Printer



MP3 Player

Goal. Java programs that interact with the outside world.

Input and Output

Input devices.



Keyboard



Mouse



Hard drive



Network



Digital camera



Microphone

Output devices.



Display



Speakers



Hard drive



Network



Printer



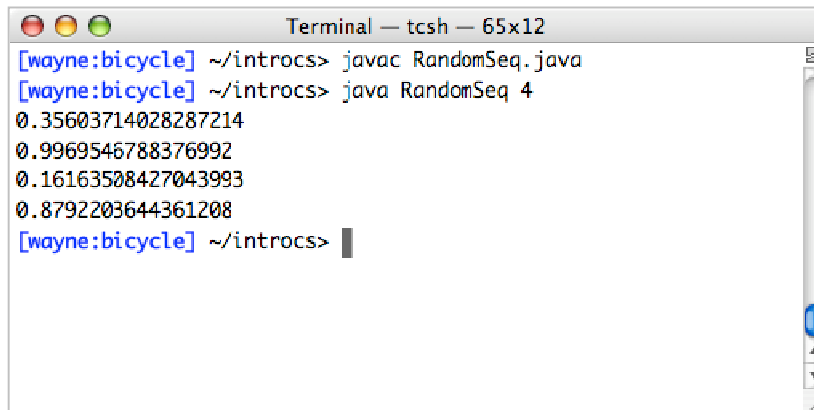
MP3 Player

Our approach.

- Define Java libraries of functions for input and output.
- Use operating system (OS) to connect Java programs to: file system, each other, keyboard, mouse, display, speakers.

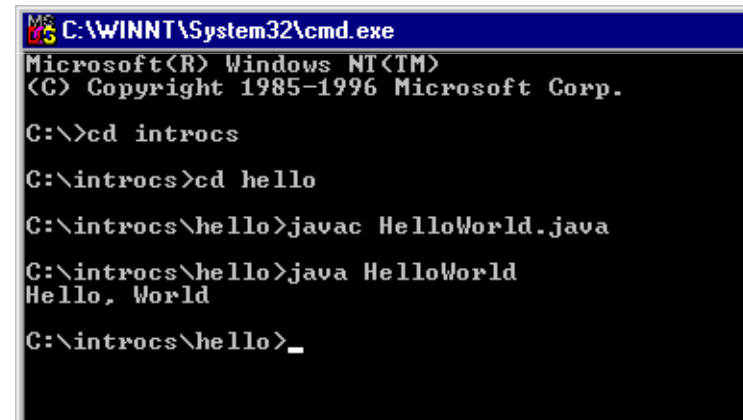
Terminal

Terminal. Application where you can type commands to control the operating system.

A screenshot of a Mac OS X Terminal window. The title bar reads "Terminal — tcsh — 65x12". The prompt is "[wayne:bicycle] ~/introc>". The user has entered "javac RandomSeq.java" and "java RandomSeq 4". The output shows four lines of floating-point numbers: "0.35603714028287214", "0.9969546788376992", "0.16163508427043993", and "0.8792203644361208". The prompt is now "[wayne:bicycle] ~/introc>".

```
Terminal — tcsh — 65x12
[wayne:bicycle] ~/introc> javac RandomSeq.java
[wayne:bicycle] ~/introc> java RandomSeq 4
0.35603714028287214
0.9969546788376992
0.16163508427043993
0.8792203644361208
[wayne:bicycle] ~/introc> |
```

Mac OS X

A screenshot of a Microsoft Windows Command Prompt window. The title bar reads "C:\WINNT\System32\cmd.exe". The prompt is "Microsoft(R) Windows NT(TM) (C) Copyright 1985-1996 Microsoft Corp.". The user has entered "cd introcs", "cd hello", "javac HelloWorld.java", and "java HelloWorld". The output shows "Hello, World". The prompt is now "C:\introcs\hello>".

```
C:\WINNT\System32\cmd.exe
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.
C:\>cd introcs
C:\introcs>cd hello
C:\introcs\hello>javac HelloWorld.java
C:\introcs\hello>java HelloWorld
Hello, World
C:\introcs\hello>_
```

Microsoft Windows

Command-Line Input and Standard Output

Command-line input. Read an integer N as command-line argument.

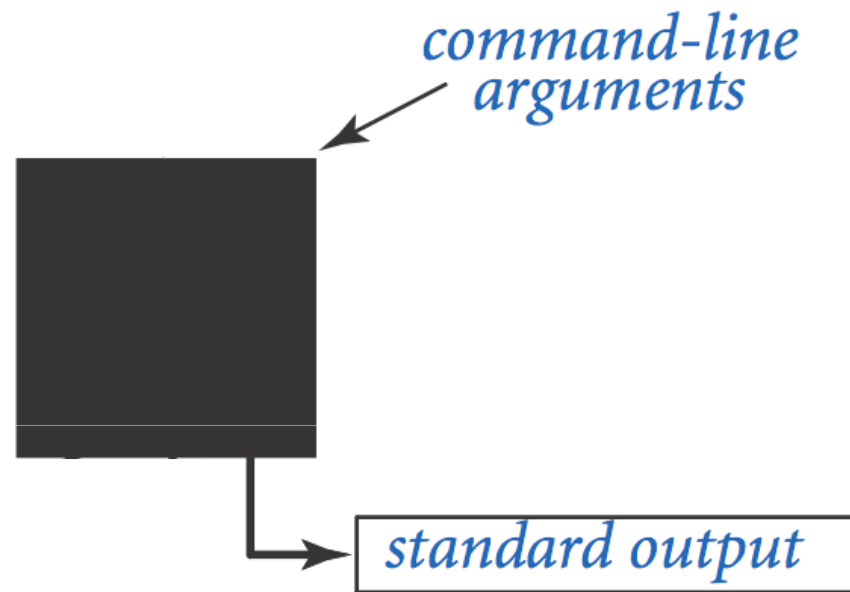
Standard output.

- Flexible OS abstraction for output.
- In Java, output from `System.out.println()` goes to standard output.
- By default, standard output is sent to Terminal.

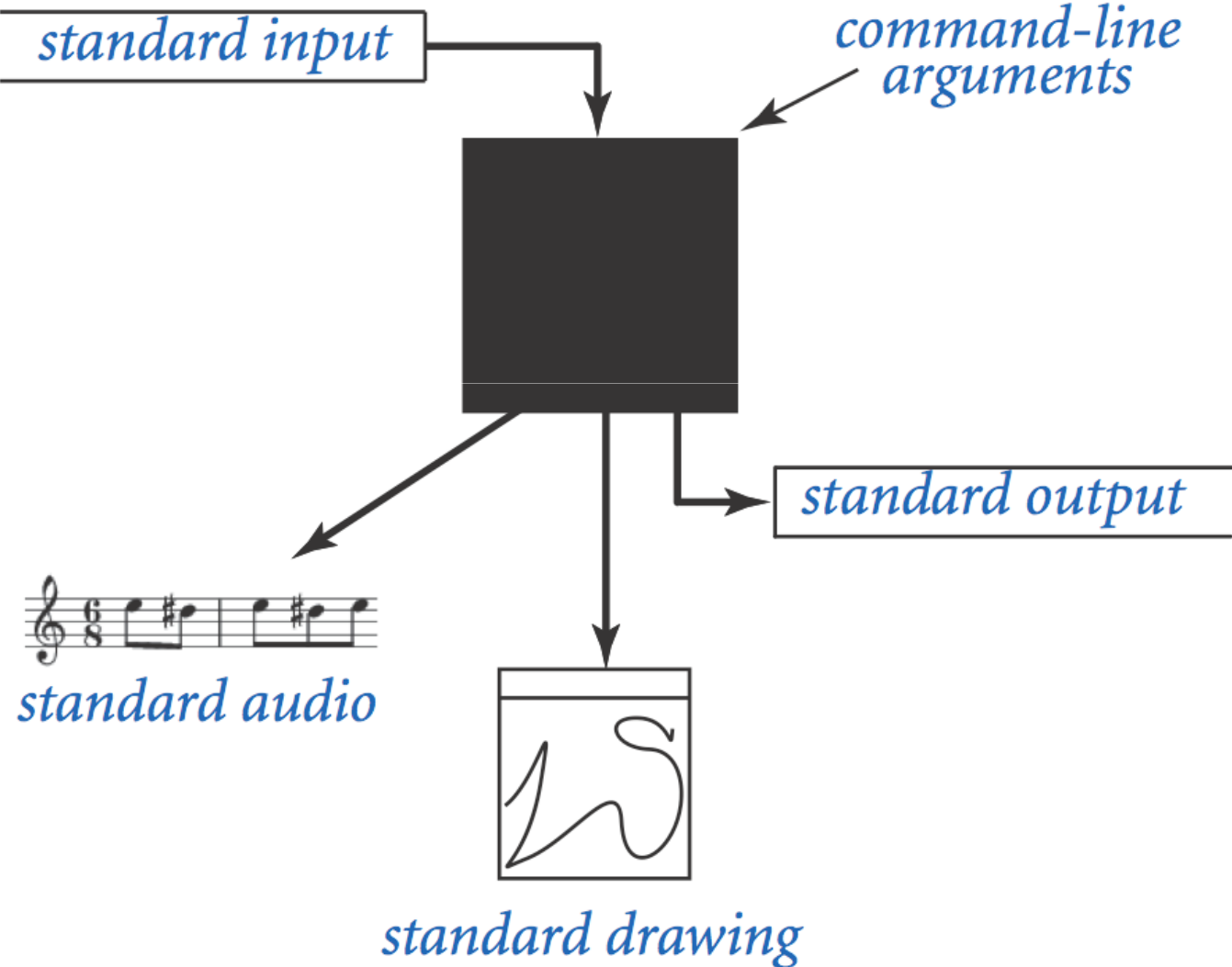
```
public class RandomSeq {  
    public static void main(String[] args) {  
        int N = Integer.parseInt(args[0]);  
        for (int i = 0; i < N; i++) {  
            System.out.println(Math.random());  
        }  
    }  
}
```

```
% java RandomSeq 4  
0.9320744627218469  
0.4279508713950715  
0.08994615071160994  
0.6579792663546435
```

Old Bird's Eye View



New Bird's Eye View



Standard Input and Output

Command-Line Input vs. Standard Input

Command-line input.

- Use command-line input to read in a **few** user values.
- Not practical for many user inputs.
- Input entered **before** program begins execution.

Standard input.

- Flexible OS abstraction for input.
- By default, standard input is received from Terminal window.
- Input entered **while** program is executing.

Standard Input and Output

Standard input. `stdIn` is library for reading text input.

Standard output. `stdOut` is library for writing text output.

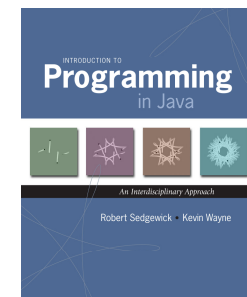
```
public class StdIn
```

<code>boolean isEmpty()</code>	<i>true if no more values, false otherwise</i>
<code>int readInt()</code>	<i>read a value of type int</i>
<code>double readDouble()</code>	<i>read a value of type double</i>
<code>long readLong()</code>	<i>read a value of type long</i>
<code>boolean readBoolean()</code>	<i>read a value of type boolean</i>
<code>char readChar()</code>	<i>read a value of type char</i>
<code>String readString()</code>	<i>read a value of type String</i>
<code>String readLine()</code>	<i>read the rest of the line</i>
<code>String readAll()</code>	<i>read the rest of the text</i>

```
public class StdOut
```

<code>void print(String s)</code>	<i>print s</i>
<code>void println(String s)</code>	<i>print s, followed by newline</i>
<code>void println()</code>	<i>print a new line</i>
<code>void printf(String f, ...)</code>	<i>formatted print</i>

libraries developed
for this course
(also broadly useful)



Standard Input and Output

To use. Download `stdIn.java` and `stdOut.java` from booksite, and put in working directory (or use classpath).

see booksite

```
public class Add {  
    public static void main(String[] args) {  
        StdOut.print("Type the first integer: ");  
        int x = StdIn.readInt();  
        StdOut.print("Type the second integer: ");  
        int y = StdIn.readInt();  
        int sum = x + y;  
        StdOut.println("Their sum is " + sum);  
    }  
}
```

```
% java Add  
Type the first integer: 1  
Type the second integer: 2  
Their sum is 3
```

Averaging A Stream of Numbers

Average. Read in a stream of numbers, and print their average.

```
public class Average {
    public static void main(String[] args) {
        double sum = 0.0; // cumulative total
        int n = 0; // number of values

        while (!StdIn.isEmpty()) {
            double x = StdIn.readDouble();
            sum = sum + x;
            n++;
        }

        StdOut.println(sum / n);
    }
}
```

```
% java Average
10.0 5.0 6.0
3.0 7.0 32.0
<Ctrl-d>
10.5
```

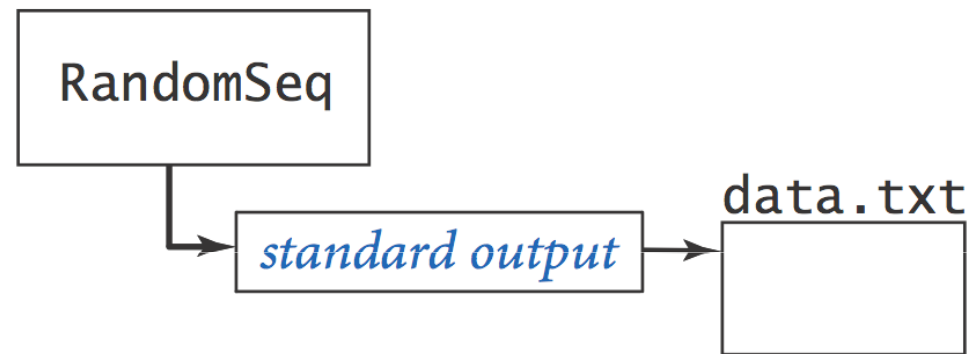
<Ctrl-d> for OS X/Linux/Unix/DrJava
<Ctrl-z> for Windows

Key point. Program does not limit the amount of data.

Redirection and Piping

Redirecting Standard Output

Redirecting standard output. Use OS directive to send standard output to a file for permanent storage (instead of terminal window).

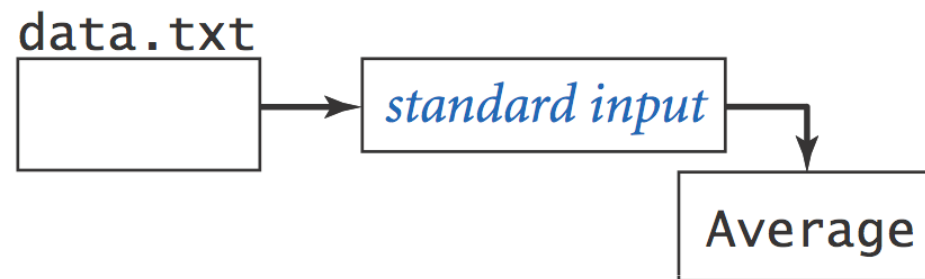


```
% java RandomSeq 1000 > data.txt
```

redirect stdout

Redirecting Standard Input

Redirecting standard input. Use OS directive to read standard input from a file (instead of terminal window).

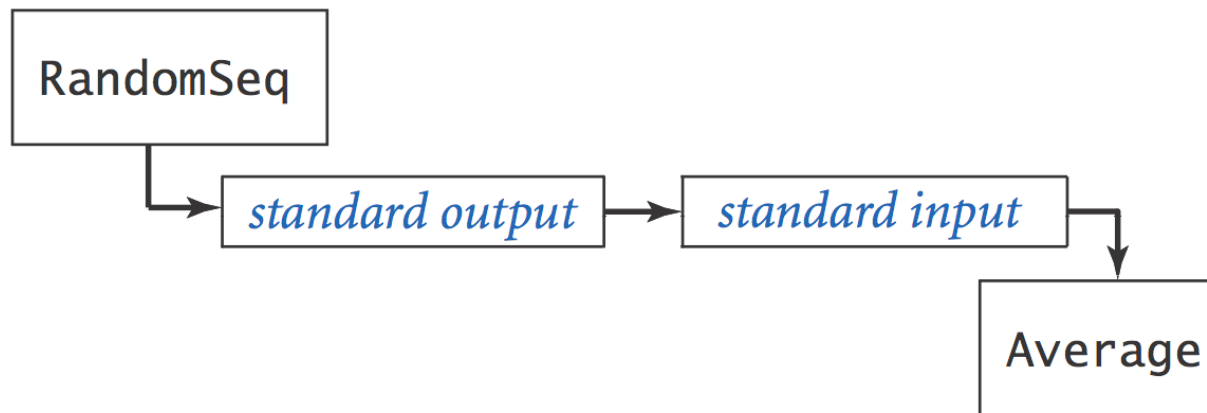


```
% more < data.txt
0.5475375782884312
0.4971087292684019
0.23123808041753813
...
% java Average < data.txt
0.4947655567740991
```

A red arrow points from the text "redirect stdin" to the "<" symbol in the second command.

Connecting Programs

Piping. Use OS directive to make the standard output of one program become the standard input of another.



pipe stdout of **RandomSeq**
to stdin of **Average**

```
% java RandomSeq 1000000 | java Average  
0.4997970473016028  
  
% java RandomSeq 1000000 | java Average  
0.5002071875644842
```


Redirecting Standard Output to a Toast Printer

```
% java HelloWorld > /dev/toaster
```



Standard Drawing

Standard Drawing

Standard drawing. `StdDraw` is library for producing graphical output.

```
public class StdDraw
```

```
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)
```

```
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)
```

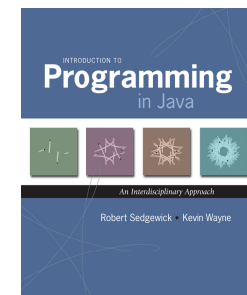
show all; pause dt milliseconds

```
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.

library developed
for this course
(also broadly useful)

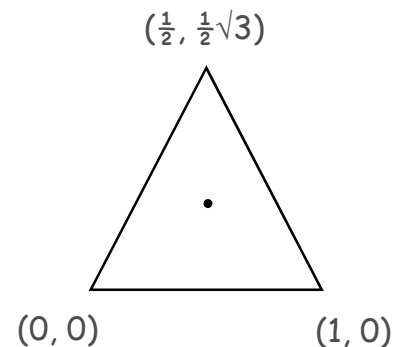


Standard Draw

Standard drawing. We provide library `stdDraw` to plot graphics.
To use. Download `stdDraw.java` and put in working directory.

```
public class Triangle {
    public static void main(String[] args) {
        double t = Math.sqrt(3.0) / 2.0;
        StdDraw.line(0.0, 0.0, 1.0, 0.0);
        StdDraw.line(1.0, 0.0, 0.5, t);
        StdDraw.line(0.5, t, 0.0, 0.0);
        StdDraw.point(0.5, t/3.0);
    }
}
```

```
% java Triangle
```



Data Visualization

Plot filter. Read in a sequence of (x, y) coordinates from standard input, and plot using standard drawing.

```
public class PlotFilter {
    public static void main(String[] args) {
        double xmin = StdIn.readDouble();
        double ymin = StdIn.readDouble();
        double xmax = StdIn.readDouble();
        double ymax = StdIn.readDouble();
        StdDraw.setXscale(xmin, xmax);
        StdDraw.setYscale(ymin, ymax);

        while (!StdIn.isEmpty()) {
            double x = StdIn.readDouble();
            double y = StdIn.readDouble();
            StdDraw.point(x, y);
        }
    }
}
```

← rescale coordinate system

← read in points, and plot them

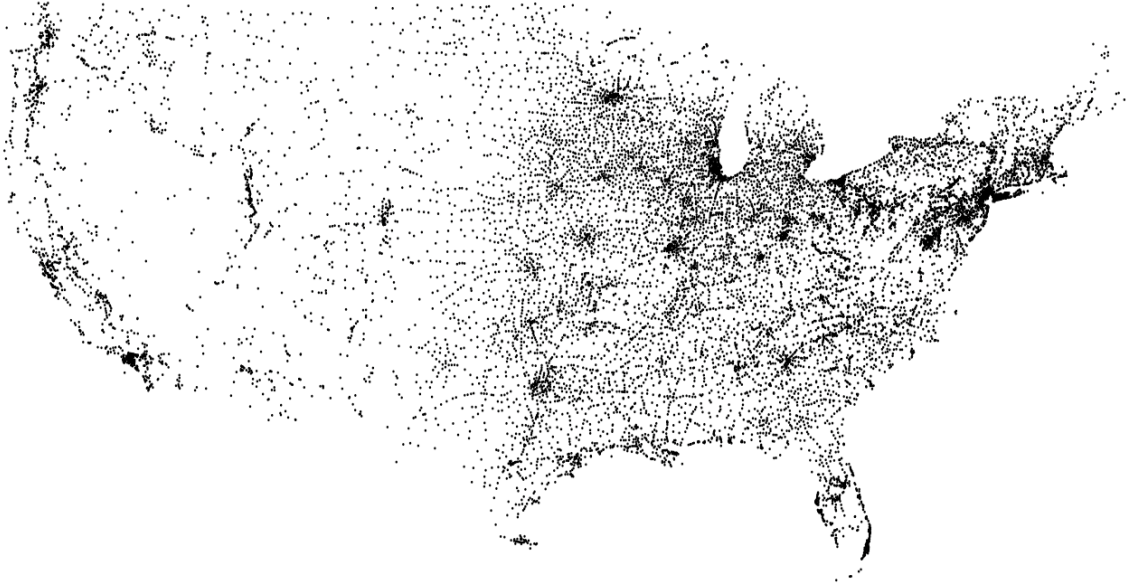
Data Visualization

```
% more < USA.txt
669905.0 247205.0 1244962.0 490000.0
1097038.8890 245552.7780
1103961.1110 247133.3330
1104677.7780 247205.5560
...

% java PlotFilter < USA.txt
```

bounding box

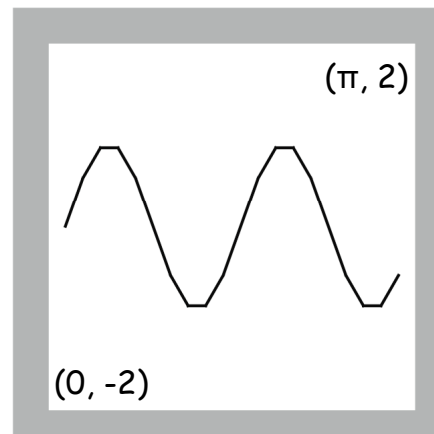
coordinates of
13,509 US cities



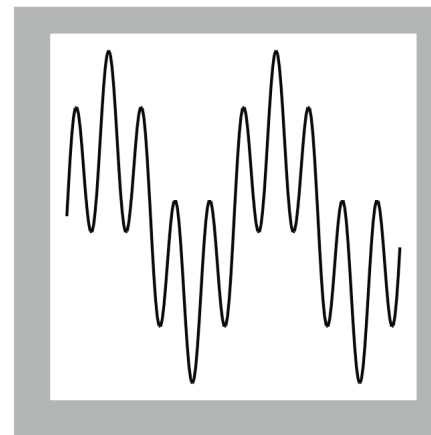
Plotting a Function

```
double[] x = new double[N+1];
double[] y = new double[N+1];
for (int i = 0; i <= N; i++) {
    x[i] = Math.PI * i / N;
    y[i] = Math.sin(4*x[i]) + Math.sin(20*x[i]);
}
StdDraw.setXscale(0, Math.PI);
StdDraw.setYscale(-2.0, +2.0);
for (int i = 0; i < N; i++)
    StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
```

$N = 20$



$N = 200$



$$y = \sin 4x + \sin 20x, x \in [0, \pi]$$

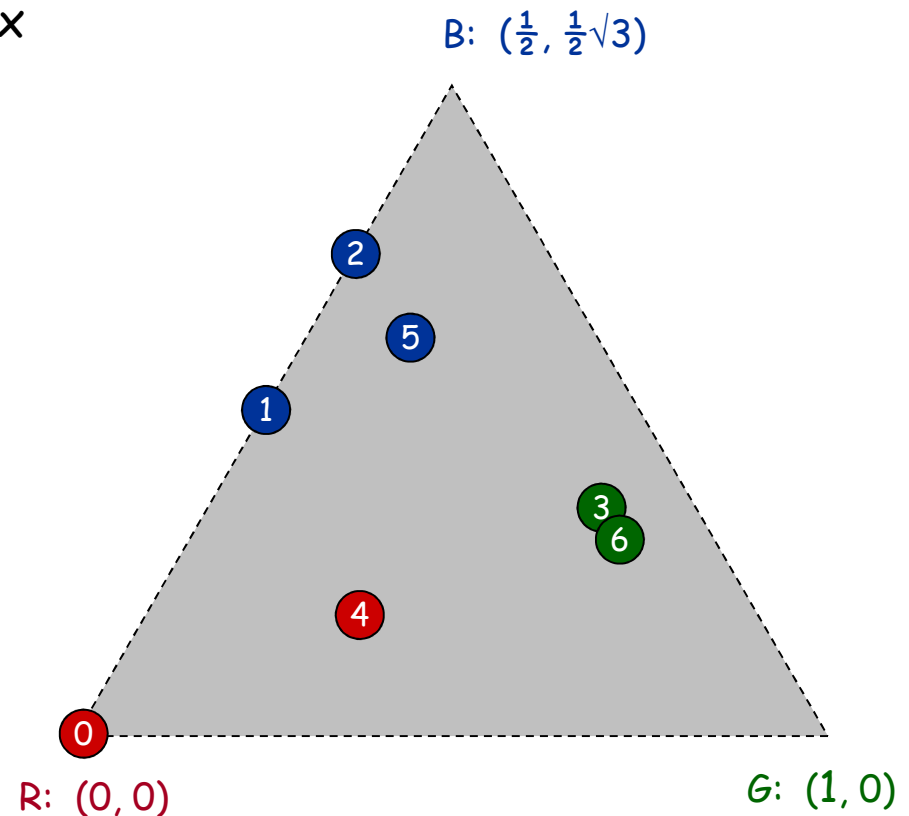
Chaos Game

Chaos game. Play on equilateral triangle, with vertices R, G, B.

- Start at R.
- Repeat the following n times:
 - pick a random vertex
 - move halfway between current point and vertex
 - draw a point in color of vertex

Q. What picture emerges?

B B G R B G ...



Chaos Game

```
public class Chaos {  
    public static void main(String[] args) {  
        int T = Integer.parseInt(args[0]);  
        double[] cx = { 0.000, 1.000, 0.500 };  
        double[] cy = { 0.000, 0.000, 0.866 };  
  
        double x = 0.0, y = 0.0;  
        for (int t = 0; t < T; t++) {  
            int r = (int) (Math.random() * 3);  
            x = (x + cx[r]) / 2.0;  
            y = (y + cy[r]) / 2.0;  
            StdDraw.point(x, y);  
        }  
    }  
}
```

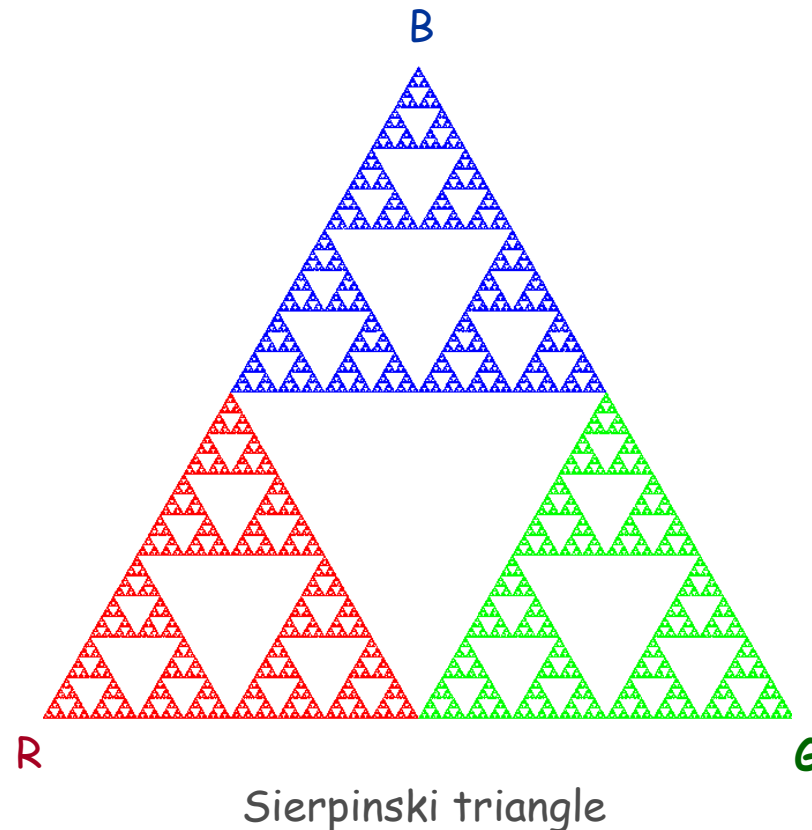
$\frac{1}{2}\sqrt{3}$
(avoid hardwired
constants like this)

between 0 and 2

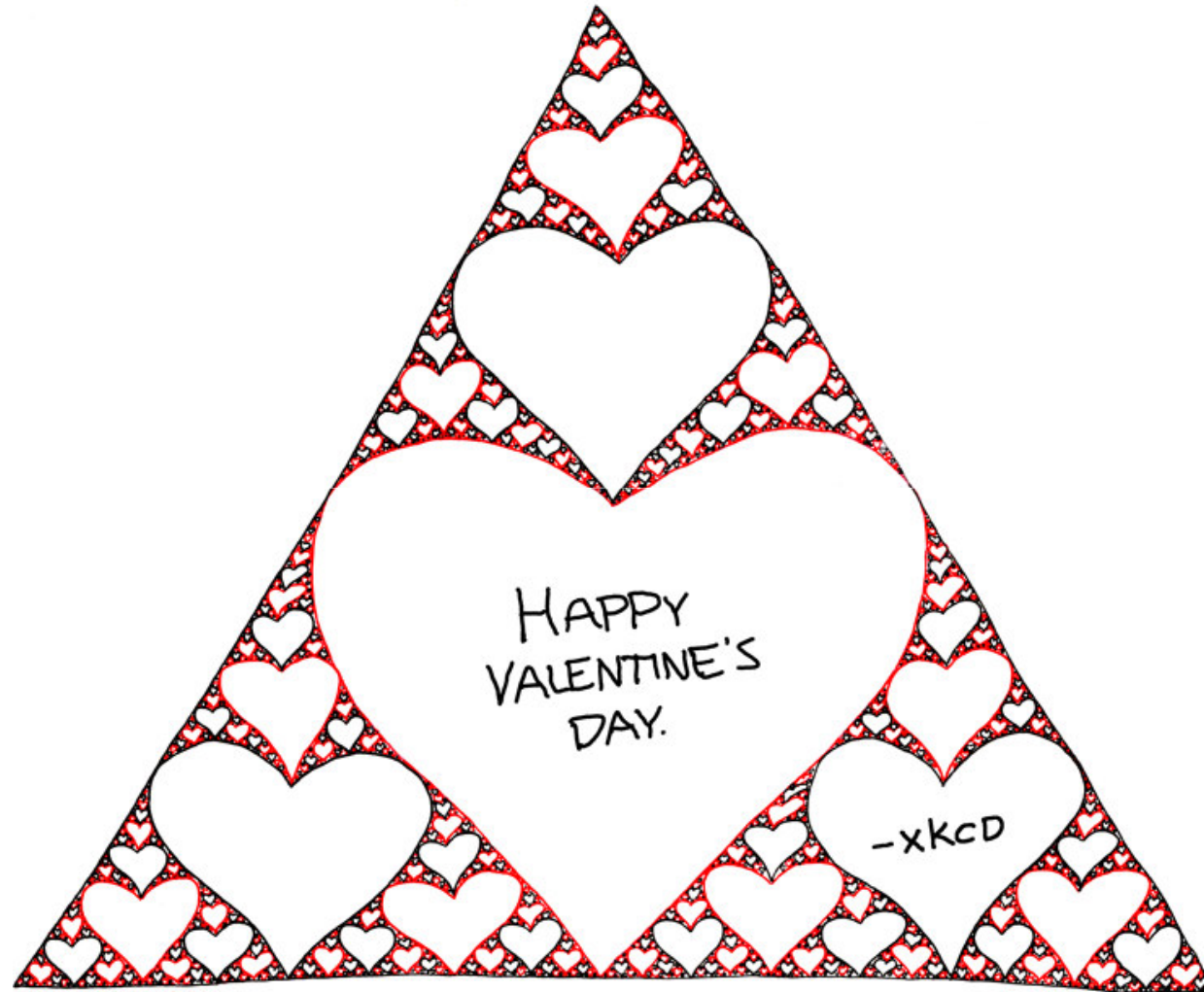
Chaos Game

Easy modification. Color point according to random vertex chosen using `StdDraw.setPenColor(StdDraw.RED)` to change the pen color.

```
% java Chaos 10000
```



Commercial Break



<http://xkcd.com/543>

Barnsley Fern

Barnsley fern. Play chaos game with different rules.

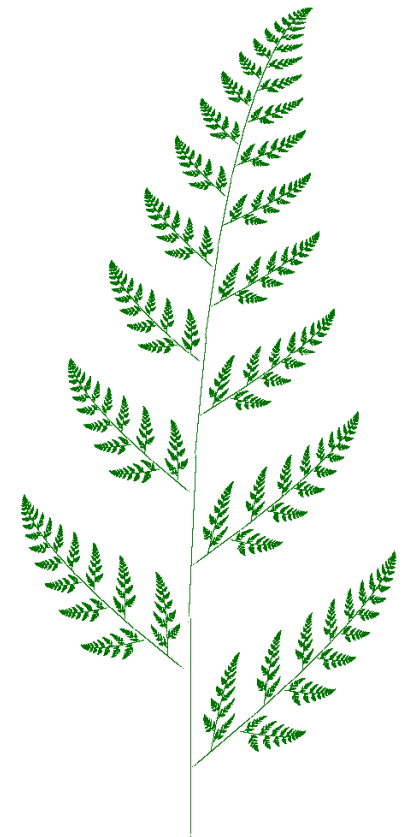
probability	new x	new y
2%	.50	.27y
15%	$-.14x + .26y + .57$	$.25x + .22y - .04$
13%	$.17x - .21y + .41$	$.22x + .18y + .09$
70%	$.78x + .03y + .11$	$-.03x + .74y + .27$

Q. What does computation tell us about nature?

Q. What does nature tell us about computation?

20th century sciences. Formulas.

21st century sciences. Algorithms?



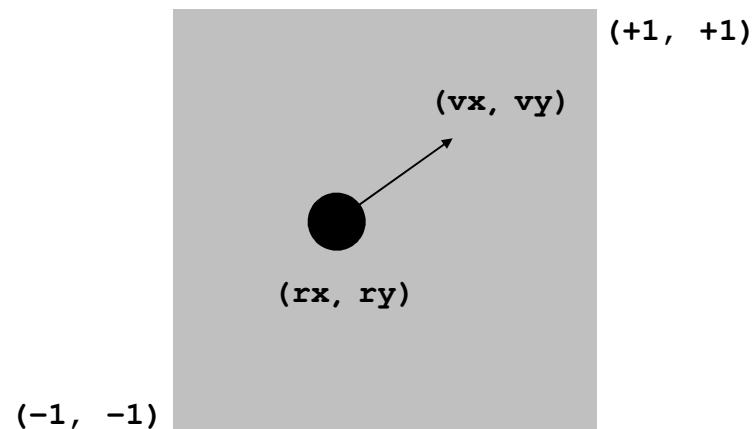
Animation

Animation loop. Repeat the following:

- Clear the screen.
- Move the object.
- Draw the object.
- Display and pause for a short while.

Ex. Bouncing ball.

- Ball has position (rx, ry) and constant velocity (vx, vy) .
- Detect collision with wall and reverse velocity.



Bouncing Ball

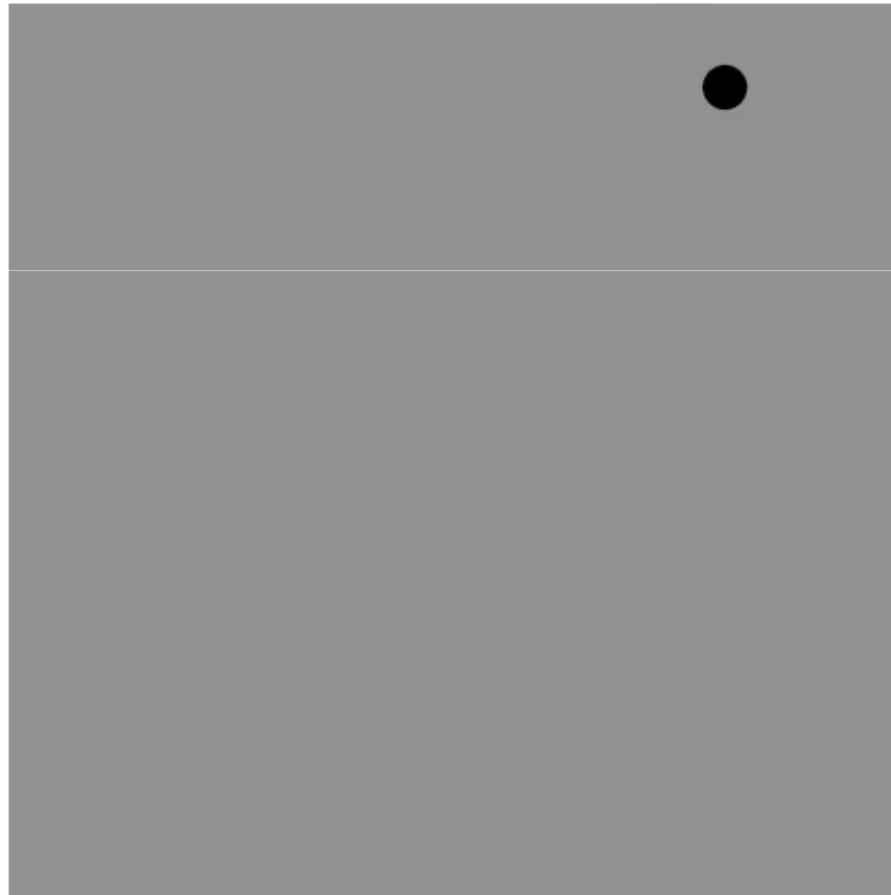
```
public class BouncingBall {  
    public static void main(String[] args) {  
        double rx = .480, ry = .860;           position  
        double vx = .015, vy = .023;         constant velocity  
        double radius = .05;                  radius  
  
        StdDraw.setXscale(-1.0, +1.0);        rescale coordinates  
        StdDraw.setYscale(-1.0, +1.0);
```

```
        while(true) {  
            bounce  
            if (Math.abs(rx + vx) + radius > 1.0) vx = -vx;  
            if (Math.abs(ry + vy) + radius > 1.0) vy = -vy;  
  
            rx = rx + vx;                       update position  
            ry = ry + vy;  
  
            StdDraw.setPenColor(StdDraw.GRAY);  clear background  
            StdDraw.filledSquare(0.0, 0.0, 1.0);  
            StdDraw.setPenColor(StdDraw.BLACK); draw the ball  
            StdDraw.filledCircle(rx, ry, radius);  
            StdDraw.show(20);  
        }  
    }  
}
```

turn on animation mode:
display and pause for 20ms

Bouncing Ball Demo

```
% java BouncingBall
```



Special Effects

Images. Put `.gif`, `.png`, or `.jpg` file in the working directory and use `StdDraw.picture()` to draw it.

Sound effects. Put `.wav`, `.mid`, or `.au` file in the working directory and use `StdAudio.play()` to play it.

Ex. Modify `BouncingBall` to display image and play sound upon collision.

- Replace `StdDraw.filledCircle()` with:

```
StdDraw.picture(rx, ry, "earth.gif");
```



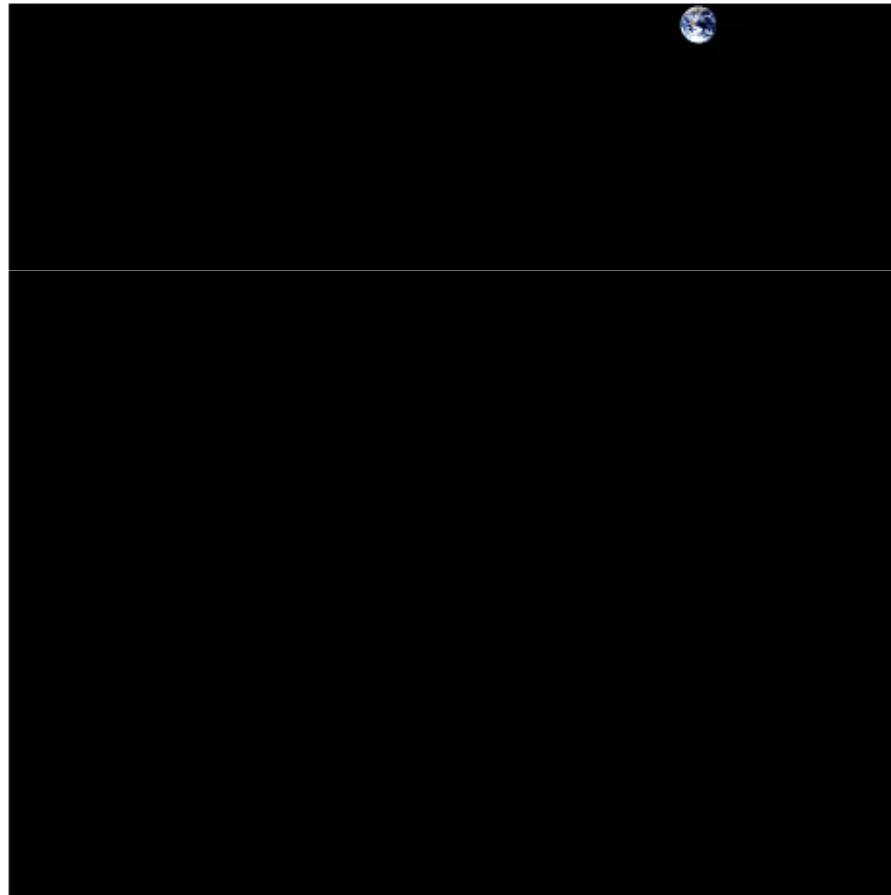
- Add following code upon collision with vertical wall:

```
StdAudio.play("laser.wav");
```



Deluxe Bouncing Ball Demo

```
% java DeluxeBouncingBall
```



Bouncing Ball Challenge

Q. What happens if you call `StdDraw.filledSquare()` once before loop (instead of inside)?

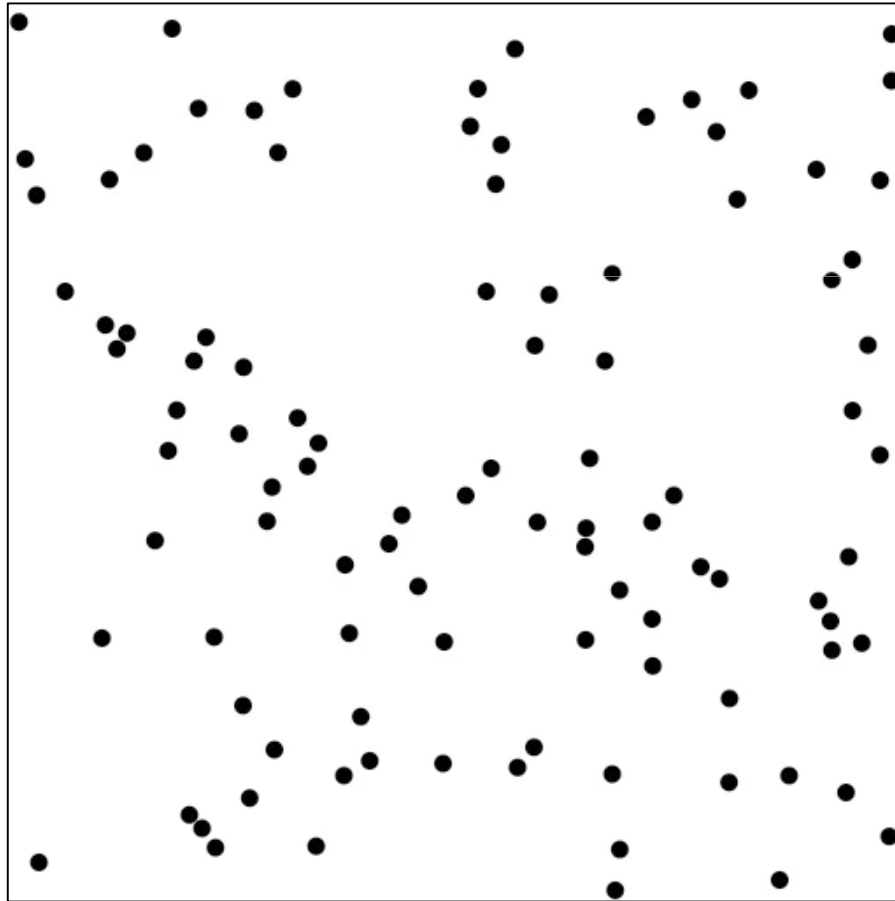
```
% java DeluxeBouncingBall
```



Colliding Balls

Challenge. Add elastic collisions.

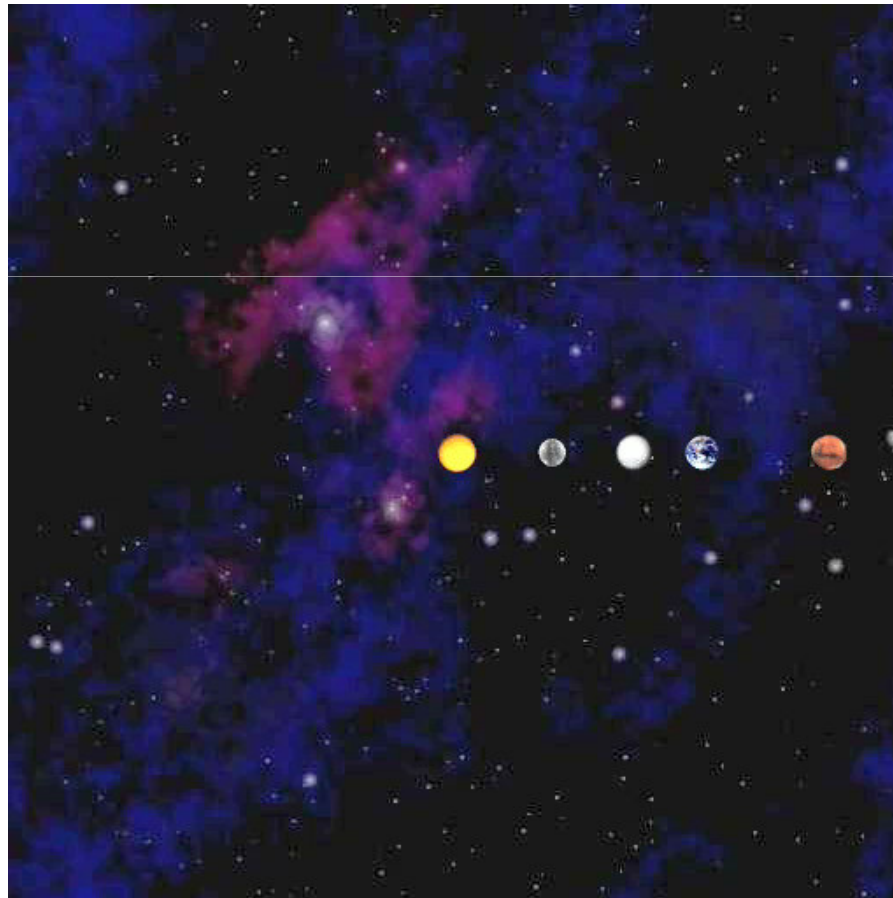
```
% java CollidingBalls 100
```



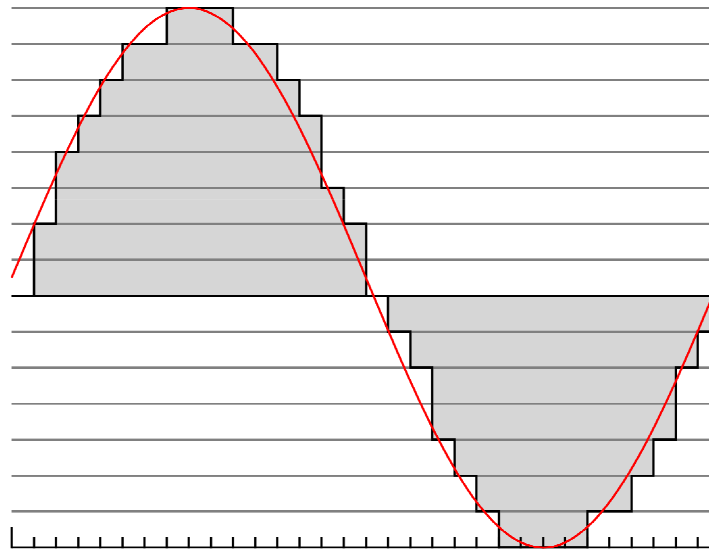
N-body Simulation

Challenge. Add gravity.

```
% java NBody < planets.txt
```



Standard Audio

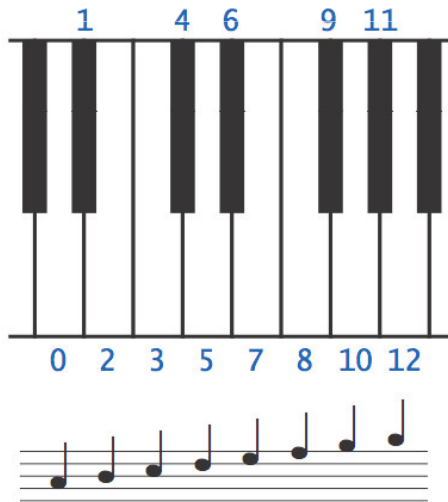


Crash Course in Sound

Sound. Perception of the **vibration** of molecules in our eardrums.

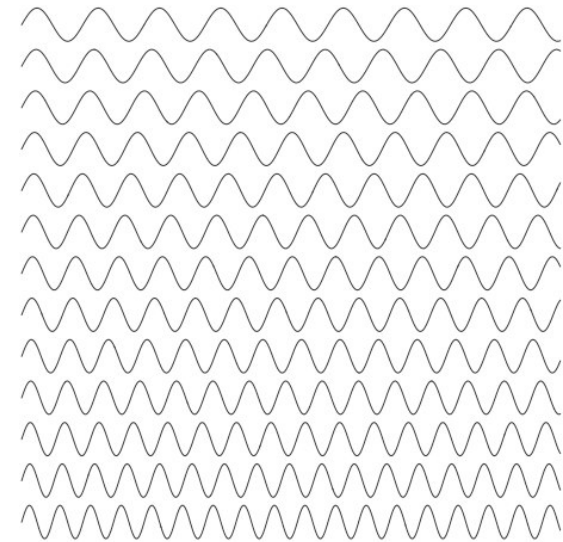
Concert A. Sine wave, scaled to oscillate at 440Hz.

Other notes. 12 notes on chromatic scale, divided logarithmically.



<i>note</i>	<i>i</i>	<i>frequency</i>
A	0	440.00
A# or B _b	1	466.16
B	2	493.88
C	3	523.25
C# or D _b	4	554.37
D	5	587.33
D# or E _b	6	622.25
E	7	659.26
F	8	698.46
F# or G _b	9	739.99
G	10	783.99
G# or A _b	11	830.61
A	12	880.00

$440 \times 2^{i/12}$



Notes, numbers, and waves

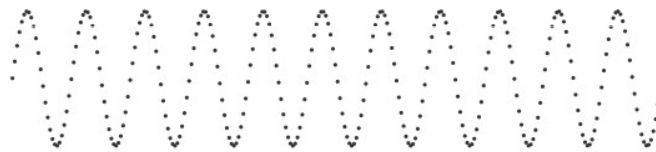
Digital Audio

Sampling. Represent curve by sampling it at regular intervals.

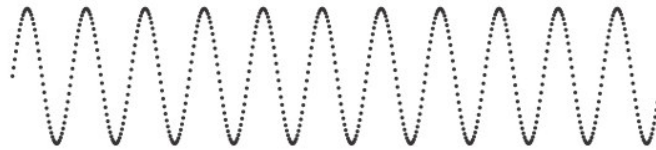
5,512 samples/second, 137 samples



11,025 samples/second, 275 samples

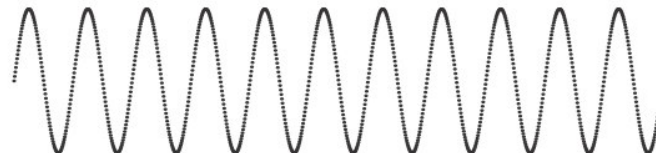


22,050 samples/second, 551 samples



44,100 samples/second, 1,102 samples

audio CD



$$y(i) = \sin\left(\frac{2\pi \cdot i \cdot 440}{44,100}\right)$$

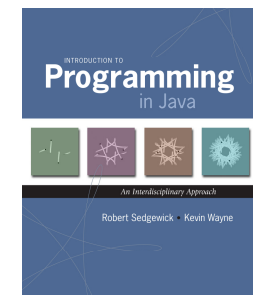
Digital Audio in Java

Standard audio. Library for playing digital audio.

```
public class StdAudio
```

<code>void play(String file)</code>	<i>play the given .wav file</i>
<code>void play(double[] a)</code>	<i>play the given sound wave</i>
<code>void play(double x)</code>	<i>play sample for 1/44100 second</i>
<code>void save(String file, double[] a)</code>	<i>save to a .wav file</i>
<code>double[] read(String file)</code>	<i>read from a .wav file</i>

library developed
for this course
(also broadly useful)



Musical Tone

Concert A. Play concert A for 1.5 seconds using StdAudio.

$$a(i) = \sin\left(\frac{2\pi \cdot i \cdot \text{hz}}{44,100}\right)$$

```
double hz = 440.0;
double seconds = 1.5;

int SAMPLE_RATE = 44100;
int N = (int) (seconds * SAMPLE_RATE);
double[] a = new double[N+1];
for (int i = 0; i <= N; i++) {
    a[i] = Math.sin(2 * Math.PI * i * hz / SAMPLE_RATE);
}
StdAudio.play(a);
```

Play That Tune

Play that tune. Read in pitches and durations from standard input; sonify using standard audio.

```
% more elise.txt  
7 .125  
6 .125  
7 .125  
6 .125  
7 .125  
2 .125  
5 .125  
3 .125  
0 .25
```

```
% java PlayThatTune < elise.txt
```



Play That Tune

```
public class PlayThatTune {
    public static void main(String[] args) {
        while (!StdIn.isEmpty()) {
            int pitch = StdIn.readInt();
            double seconds = StdIn.readDouble();
            double hz = 440.0 * Math.pow(2, pitch / 12.0);

            int SAMPLE_RATE = 44100;
            int N = (int) (seconds * SAMPLE_RATE);
            double[] a = new double[N+1];
            for (int i = 0; i <= N; i++) {
                a[i] = Math.sin(2 * Math.PI * i * hz / SAMPLE_RATE);
            }
            StdAudio.play(a);
        }
    }
}
```

code as before

1.5 Extra Slides

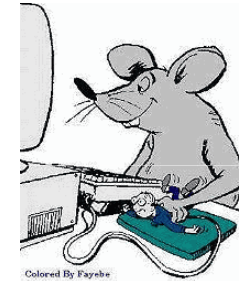
User Interfaces

Command line interface.

- User types commands at terminal.
- Easily customizable.
- Extends to complex command sequences.

Point and click.

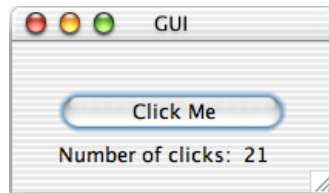
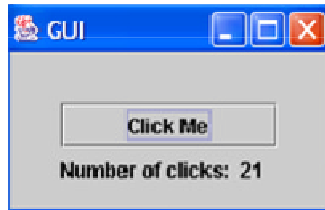
- User launches applications by clicking.
 - `File` → `Open` → `HelloWorld.java`
- Restricted to pre-packaged menu options.



Swing Graphical User Interface

"Swing" is Java's GUI.

- Buttons.
- Menus.
- Scrollbars.
- Toolbars.
- File choosers.



```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class GUI implements ActionListener {
    private int clicks = 0;
    private JFrame frame = new JFrame();
    private JLabel label = new JLabel("Number of clicks: 0");
    public GUI() {
        JButton button = new JButton("Click Me");
        button.addActionListener(this);
        JPanel panel = new JPanel();
        panel.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));
        panel.setLayout(new GridLayout(0, 1));
        panel.add(button);
        panel.add(label);
        frame.add(panel, BorderLayout.CENTER);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setTitle("GUI");
        frame.pack();
        frame.show();
    }

    public void actionPerformed(ActionEvent e) {
        clicks++;
        label.setText("Number of clicks: " + clicks);
    };

    public static void main(String[] args) {
        GUI gui = new GUI();
    }
}
```

a sample Swing application

Ignore details.

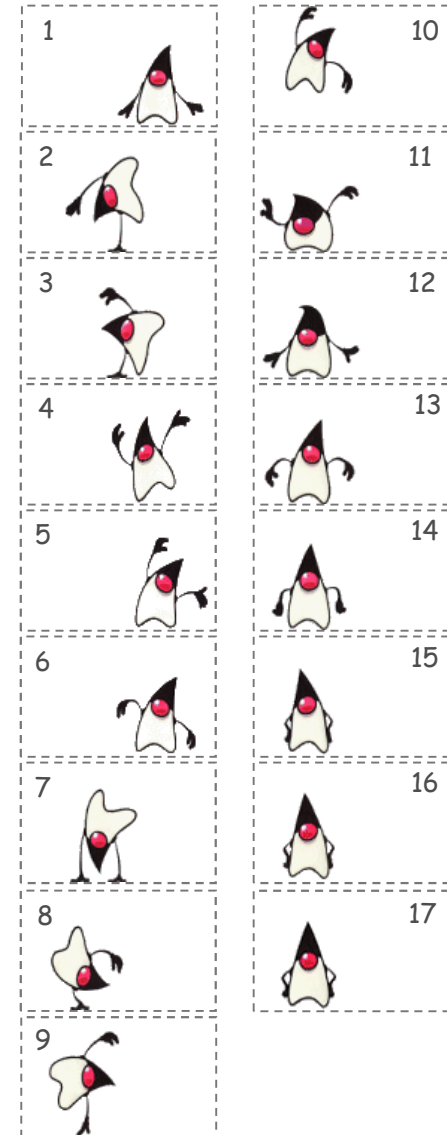
Computer Animation

Computer animation. Display a sequence of closely related images in rapid succession to produce the illusion of movement.

Frame rate. Use 15-70 frames per second to "trick" human eye and brain into seeing smooth motion.

Ex 1. Television and motion pictures.

Ex 2. Java mascot Duke cart-wheeling.



Java Implementation

```
public class Duke {  
    public static void main(String[] args) {  
        int images = 17;  
        int WIDTH = 130, HEIGHT = 80;  
        StdDraw.setCanvasSize(WIDTH, HEIGHT);  
  
        for (int t = 0; true; t++) {  
            int i = 1 + (t % images);  
            String file = "T" + i + ".gif";  
            StdDraw.picture(0.5, 0.5, file);  
            StdDraw.show(100);  
        }  
    }  
}
```

T1.gif - T17.gif



Operating System Specific Details

Common OS abstractions.

Operation	Windows XP	OS X	Unix
Cycle through recent command	Up, down arrows	Up, down arrows	Up, down arrows
File name completion	Tab	Tab	Tab
End of file	Ctrl-z	<Enter>Ctrl-d	Ctrl-d
Newline character	\r\n	\n or \r	\n
Scroll through text, one screenful at a time	more	more less	more less
List files in current directory	dir	ls	ls
Redirection, pipes	<, >,	<, >,	<, >,
File system	C:\introcs\Hi.java	/u/introcs/Hi.java	/u/introcs/Hi.java

Unix means Unix variants (Linux, Solaris, Aix)

Most Windows XP commands also supported in other version of Windows.

Twenty Questions

Twenty questions. User thinks of an integer between one and 1 million. Computer tries to guess it.

```
public class TwentyQuestions {
    public static void main(String[] args) {
        int lo = 1, hi = 1000000;
        while (lo < hi) {
            int mid = (lo + hi) / 2;
            StdOut.println("Is your number <= " + mid + "?");
            boolean response = StdIn.readBoolean();
            if (response) hi = mid;
            else          lo = mid + 1;
        }
        StdOut.println("Your number is " + lo);
    }
}
```

Binary search. Each question removes half of possible remaining values.

Consequence. Always succeeds after 20 questions.

$2^{20} \approx 1$ million

invariant: user's number
always between lo and hi

Using the standard libraries - stdlib.jar

The file `stdlib.jar` bundles together all of our standard libraries into one file. There are a number of ways to access the libraries:

Current directory. The easiest (but not the sanest) way to use the standard libraries is to download `stdlib.jar` and `unjar` it in your current working directory.

```
% jar xf stdlib.jar
```

Alternatively, you can download the individual `.java` files you need (such as `StdIn.java`) and put them in the same directory as the program you are writing. Then, compile and execute as usual.

```
% javac MyProgram.java  
% java MyProgram
```

This approach has the drawback that you need a copy of each `.java` file you need in each directory where you need it.

Using the standard libraries - stdlib.jar

Classpath. Put `stdlib.jar` in the same directory as the program you are writing (but do not unjar it). Then, compile and execute as follows:

Windows

```
% javac -cp .;stdlib.jar MyProgram.java
```

```
% java -cp .;stdlib.jar MyProgram
```

OS X / Linux

```
% javac -cp .:stdlib.jar MyProgram.java
```

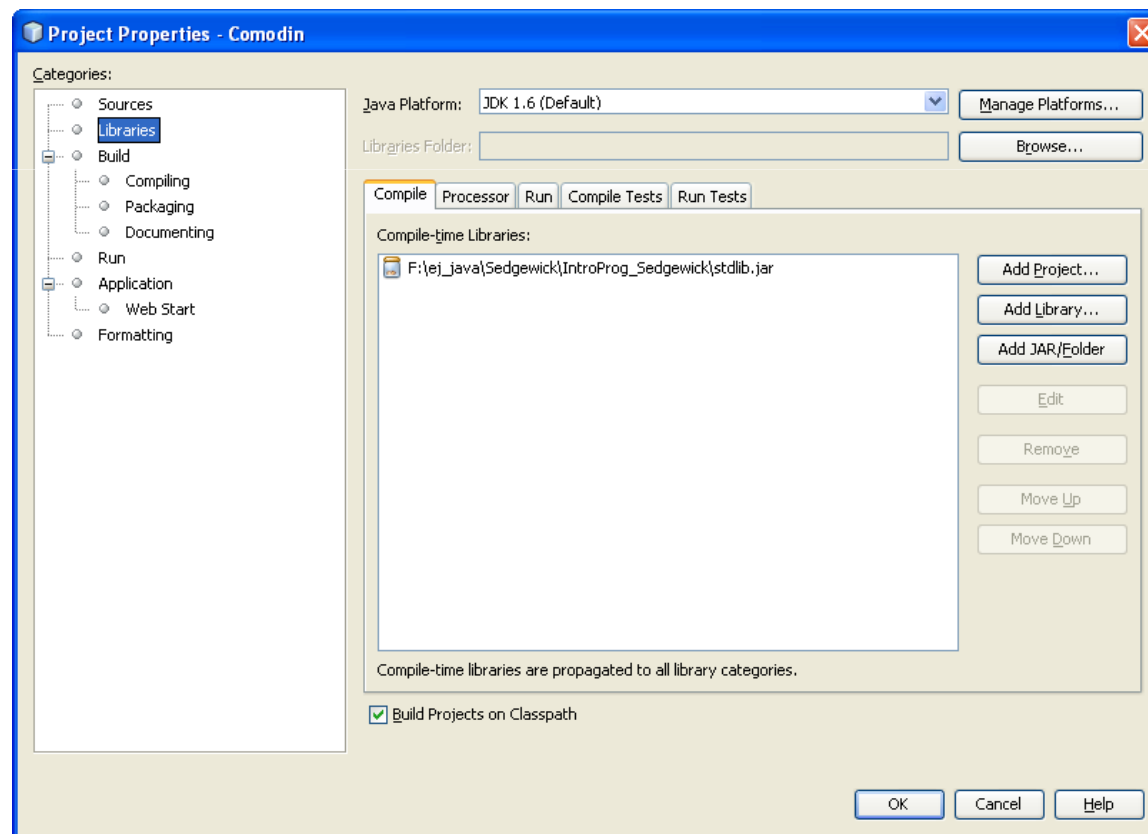
```
% java -cp .:stdlib.jar MyProgram
```

The `-cp` flag sets the classpath. The `.` tells Java to look in the current directory for `.java` and `.class` files (such as `MyProgram.java` and `MyProgram.class`). The `stdlib.jar` tells Java to also look in the `.jar` file. On OS X, the `:` separates directories in the classpath; on Windows the `;` separates directories.

Using the standard libraries - stdlib.jar

Configure your IDE.

- Right-click the project's node, choose Properties, select the Libraries category, and modify the listed classpath entries.
- Right-click the Libraries node in the Projects window and choose Add JAR/Folder.



Using the standard libraries - stdlib.jar

Setting the environment variable CLASSPATH.

- Similar procedure as configuration the environment variable PATH.

