1.5 Input and Output

INTRODUCTION TO

Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick • Kevin Wayne

Input and Output

Input devices.

| Keyboard | Mouse | Hard drive | Network | Digital camera | Microphone |

Output devices.

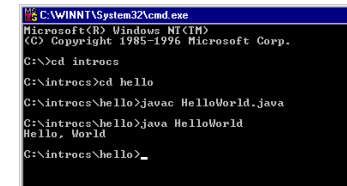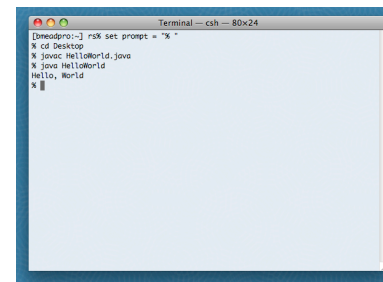| Display | Speakers | Hard drive | Network | Printer | MP3 Player |

Goal. Java programs that interact with the outside world.

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.

3

Terminal

Terminal. Application for typing commands to control the operating system.

```
[bmeadpro:~] rs% set prompt = "% "
% cd Desktop
% javac HelloWorld.java
% java HelloWorld
Hello, World
%
```

```
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

4

Command-Line Input and Standard Output

Command-line input. Read an integer N as command-line argument.
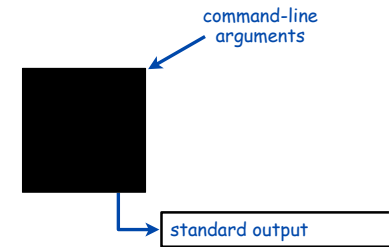
Standard output.
- Flexible OS abstraction for output.
- In Java, output from `System.out.println()` goes to standard output.
- By default, standard output is sent to Terminal.

```java
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
```
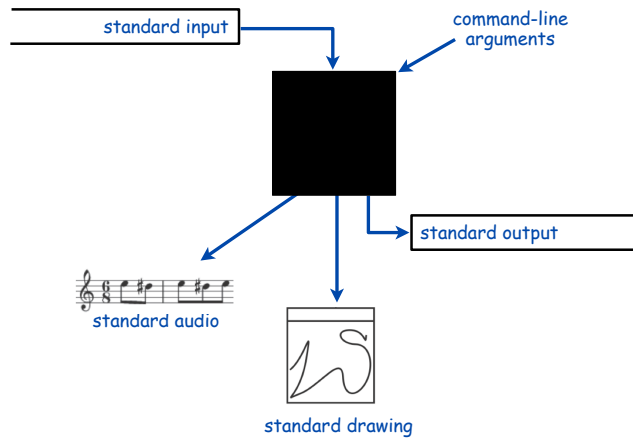
Old Bird's Eye View

command-line arguments

standard output

New Bird's Eye View

standard input

command-line arguments

standard output

standard audio

standard drawing

Standard Input and Output

Command-Line Input vs. Standard Input

Command-line inputs.
- Useful for providing a *few* user values (arguments) to a program.
- Not practical for a large or unspecified number of user inputs.
- Input entered *before* program begins execution.

Standard input.
- Flexible OS abstraction for input.
- Useful for providing an *unlimited amount* of data to a program.
- By default, standard input is received from Terminal window.
- Input entered *while* program is executing.

Standard Input and Output

Standard input. `StdIn` library has methods to read text input.

Standard output. `StdOut` library has methods to write text output.

```
public class StdIn
    boolean isEmpty()          true if no more values, false otherwise
        int readInt()          read a value of type int
     double readDouble()       read a value of type double
       long readLong()         read a value of type long
    boolean readBoolean()      read a value of type boolean
       char readChar()         read a value of type char
     String readString()       read a value of type String
     String readLine()         read the rest of the line
     String readAll()          read the rest of the text
```

```
public class StdOut
       void print(String s)        print s
       void println(String s)      print s, followed by a newline
       void println()              print a new line
       void printf(String f, ...)  formatted print
```

libraries developed for this course (and also broadly useful)

Programming

Standard IO Warmup

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
```

Standard IO Example: 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
```

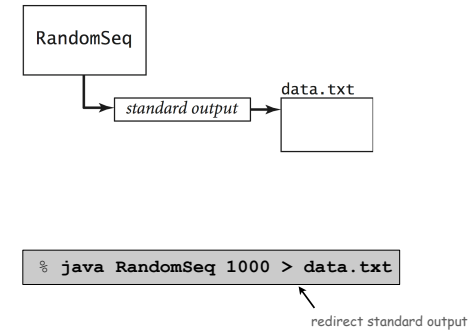**Key point.** Program does not limit amount of data.

<Ctrl-d> is OS X/Linux/Unix/DrJava EOF
<Ctrl-z> is Windows analog

Redirection and Piping

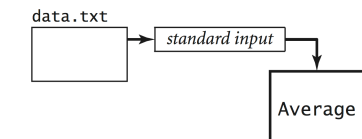## Redirecting Standard Output

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 standard output

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
…

                         standard input

% java Average < data.txt
0.4947655567740991
```

Connecting Programs

Piping. Use OS directive to make the standard output of one program become the standard input of another.
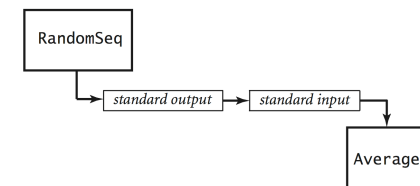
```
% java RandomSeq 1000000 | java Average
0.4997970473016028

% java RandomSeq 1000000 | java Average
0.5002071875644842
```

Key point. Program does not limit amount of data.

Standard Drawing

Standard drawing. `StdDraw` library has methods to produce 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
    void setYscale(double y0, double y1)    reset y range
    void setPenRadius(double r)
    void setPenColor(Color c)
    void setFont(Font f)
    void setCanvasSize(int w, int h)
    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 .jpg or .png file
```
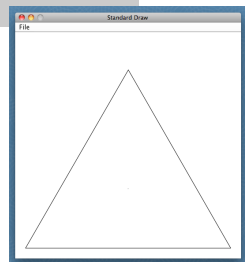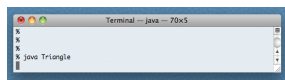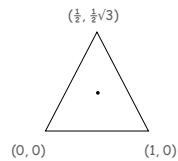
library developed
for this course
(and also broadly useful)

Programming
in Java

18

"Hello World" for Standard Draw

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);
    }
}
```

$(\frac{1}{2}, \frac{1}{2}\sqrt{3})$

$(0, 0)$ $(1, 0)$

Standard Draw
File

Terminal — java — 70×5
%
%
% java Triangle

19

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

20

Data Visualization

```
% more < USA.txt
669905.0 247205.0 1244962.0 490000.0          ← bounding box
 1097038.8890   245552.7780                     ← coordinates of
 1103961.1110   247133.3330                       13,509 US cities
 1104677.7780   247205.5560
 ...


% java PlotFilter < USA.txt
```

Plotting a Function with `StdDraw`

```java
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]);
```

$N = 20$ $N = 200$
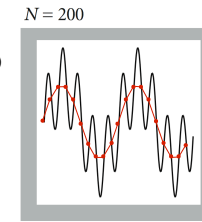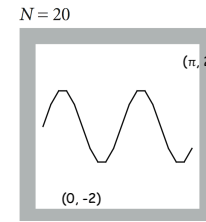
Lesson 1: Plotting is simple.

$(\pi, 2)$

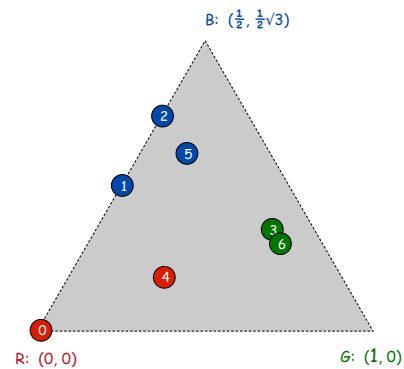Lesson 2: If you don't plot enough points, you might miss something!

$(0, -2)$

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

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 …

B: $(\frac{1}{2}, \frac{1}{2}\sqrt{3})$

R: $(0, 0)$ G: $(1, 0)$

Example: Chaos Game

```java
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);
        }
    }
}
```
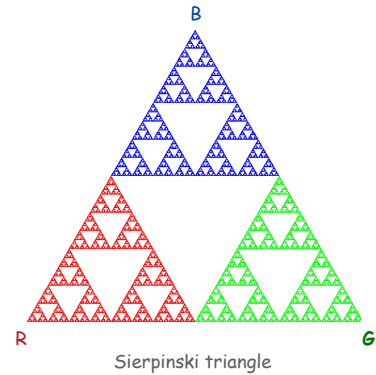
$\frac{1}{2}\sqrt{3}$
(best to avoid hardwired constants like this)

result: 0, 1 , or 2

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

`% java Chaos 10000`

Sierpinski triangle

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?

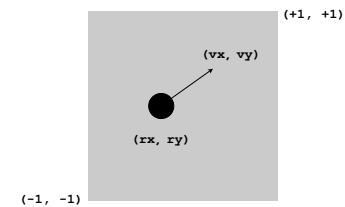20th century sciences. Formulas.

21st century sciences. Algorithms?

Animation 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.

(+1, +1)

(vx, vy)

(rx, ry)

(-1, -1)

Bouncing Ball

```
public class BouncingBall
{
   public static void main(String[] args)
   {
      double rx = .480, ry = .860;
      double vx = .015, vy = .023;
      double radius = .05;

      StdDraw.setXscale(-1.0, +1.0);
      StdDraw.setYscale(-1.0, +1.0);

      while(true)
      {
         if (Math.abs(rx + vx) + radius > 1.0) vx = -vx;    bounce
         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);
         StdDraw.filledCircle(rx, ry, radius);             draw the ball
         StdDraw.show(20);
      }                           turn on animation mode:
   }                              display and pause for 50ms
}
```
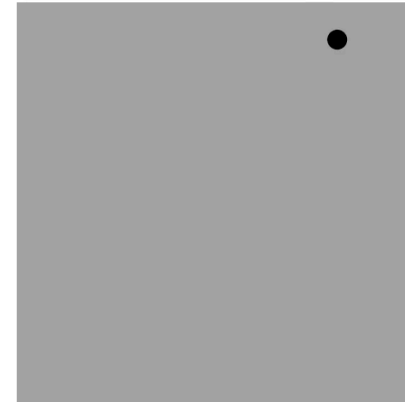
Bouncing Ball Demo

`% 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.   stay tuned for more on `StdAudio`

Ex. Modify `BouncingBall` to display image and play sound upon collision.
  • Replace `StdDraw.filledCircle()` with:

```
StdDraw.picture(rx, ry, "earth.gif");
```
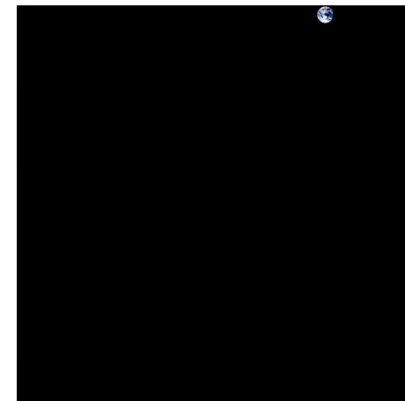
  • Add following code upon collision with wall:

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

Deluxe Bouncing Ball Demo

`% java DeluxeBouncingBall`

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

```
public class BouncingBall
{
    public static void main(String[] args)
    {
        double rx = .480, ry = .860;
        double vx = .015, vy = .023;
        double radius = .05;

        StdDraw.setXscale(-1.0, +1.0);
        StdDraw.setYscale(-1.0, +1.0);

        while(true)
        {
            if (Math.abs(rx + vx) + radius > 1.0) vx = -vx;
            if (Math.abs(ry + vy) + radius > 1.0) vy = -vy;

            rx = rx + vx;
            ry = ry + vy;

            StdDraw.setPenColor(StdDraw.GRAY);
            StdDraw.filledSquare(0.0, 0.0. 1.0);
            StdDraw.setPenColor(StdDraw.BLACK);
            StdDraw.filledCircle(rx, ry, radius);
            StdDraw.show(20);
        }
    }
}
```

```
public class BouncingBall
{
    public static void main(String[] args)
    {
        double rx = .480, ry = .860;
        double vx = .015, vy = .023;
        double radius = .05;

        StdDraw.setXscale(-1.0, +1.0);
        StdDraw.setYscale(-1.0, +1.0);
        StdDraw.filledSquare(0.0, 0.0. 1.0);

        while(true)
        {
            if (Math.abs(rx + vx) + radius > 1.0) vx = -vx;
            if (Math.abs(ry + vy) + radius > 1.0) vy = -vy;

            rx = rx + vx;
            ry = ry + vy;

            StdDraw.setPenColor(StdDraw.GRAY);
            StdDraw.setPenColor(StdDraw.BLACK);
            StdDraw.filledCircle(rx, ry, radius);
            StdDraw.show(20);
        }
    }
}
```
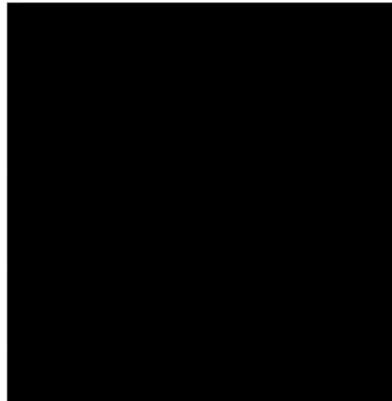
33

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

```
% java DeluxeBouncingBall
```


34

Digital Audio in Java

Standard audio. Library for playing digital audio.

```
public class StdAudio

    void play(String file)           play the given .wav file
    void play(double[] a)            play the given sound wave
    void play(double x)              play sample for 1/44100 second
    void save(String file, double[] a)   save to a .wav file
double[] read(String file)           read from a .wav file
```
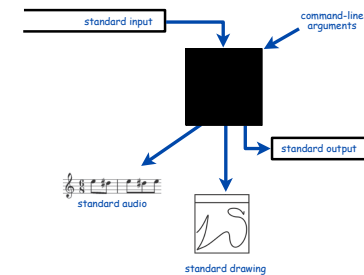
library developed
for this course
(also broadly useful)

Stay tuned. Example client in next lecture.

Programming
in Java

35

Input/Output Summary

standard input

command-line
arguments

standard output

standard audio

standard drawing

Command-line arguments. Parameters to control your program.

Standard input. Data for your program to process.

Standard output. Results of your program, or data for another program.

Standard drawing. Graphical output.

Standard audio. Sound output.

36