CS 112 Introduction to Programming
(Spring 2012)

Lectures #10-11: Input and Output

Zhong Shao

Department of Computer Science
Yale University
Office: 314 Watson

http://flint.cs.yale.edu/cs112

Acknowledgements: some slides used in this class are taken directly or adapted from those accompanying the textbook: Introduction to Programming in Java: An Interdisciplinary Approach by Robert Sedgewick and Kevin Wayne (Copyright 2002-2010)
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.
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 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.

```java
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 newline
    void println() // print a new line
    void printf(String f, ...) // formatted print
}
```

Libraries developed for this course (also broadly useful)
To use. Download `StdIn.java` and `StdOut.java` from booksite, and put in working directory (or use classpath).

```java
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.

```java
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);
    }
}
```

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

% java Average
10.0 5.0 6.0
3.0 7.0 32.0

<Ctrl-d> for OS X/Linux/Unix/DrJava
<Ctrl-z> for Windows
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. 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
```
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`
Redirecting Standard Output to a Toast Printer

% java HelloWorld > /dev/toaster
Standard Drawing
Standard Drawing

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

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

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

**Standard drawing.** We provide library `StdDraw` to plot graphics. To use, download `StdDraw.java` and put in working directory.

```java
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

![Diagram of a triangle with vertices at (0, 0), (1, 0), and (½, ½√3).]
Data Visualization

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

```java
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);
        }
    }
}
```
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

coordinates of 13,509 US cities

bounding box
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]);

\[
y = \sin 4x + \sin 20x, \ x \in [0, \pi]
\]
**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 ...
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);
        }
    }
}
Chaos Game

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

```
% java Chaos 10000
```

Sierpinski triangle
Commercial Break

HAPPY VALENTINE'S DAY.

-xKCD

http://xkcd.com/543
Commercial Break
Barnsley Fern. Play chaos game with different rules.

<table>
<thead>
<tr>
<th>probability</th>
<th>new x</th>
<th>new y</th>
</tr>
</thead>
<tbody>
<tr>
<td>2%</td>
<td>.50</td>
<td>.27y</td>
</tr>
<tr>
<td>15%</td>
<td>-.14x + .26y + .57</td>
<td>.25x + .22y - .04</td>
</tr>
<tr>
<td>13%</td>
<td>.17x - .21y + .41</td>
<td>.22x + .18y + .09</td>
</tr>
<tr>
<td>70%</td>
<td>.78x + .03y + .11</td>
<td>-.03x + .74y + .27</td>
</tr>
</tbody>
</table>

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.

```
(-1, -1)  (+1, +1)
```

\[
\begin{align*}
(vx, vy) & \\
(rx, ry) & \\
\end{align*}
\]
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); // rescale coordinates

        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); // draw the ball
            StdDraw.setPenColor(StdDraw.BLACK);
            StdDraw.filledCircle(rx, ry, radius);
            StdDraw.show(20); // turn on animation mode: display and pause for 20ms
        }
    }
}
Bouncing Ball Demo

```bash
% 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 \texttt{StdDraw.filledSquare()} once before loop (instead of inside)?

```
% java DeluxeBouncingBall
```
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.

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

440×2^{7/12}

*Notes, numbers, and waves*
Digital Audio

Sampling. Represent curve by sampling it at regular intervals.

\[
y(i) = \sin \left( \frac{2 \pi \cdot i \cdot 440}{44,100} \right)
\]
Digital Audio in Java

Standard audio. Library for playing digital audio.

```java
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)
Musical Tone

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

\[ a(i) = \sin \left( \frac{2\pi \cdot i \cdot hz}{sampleRate} \right) \]

declare double hz = 440.0;
declare double seconds = 1.5;

declare int SAMPLE_RATE = 44100;
declare int N = (int) (seconds * SAMPLE_RATE);
declare 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. 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
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);
        }
    }
}
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" is Java's GUI.

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

Believe it or not, the Swing Graphical User Interface API is not too difficult to use

```java
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
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.

<table>
<thead>
<tr>
<th>Frame 1</th>
<th>Frame 2</th>
<th>Frame 3</th>
<th>Frame 4</th>
<th>Frame 5</th>
<th>Frame 6</th>
<th>Frame 7</th>
<th>Frame 8</th>
<th>Frame 9</th>
<th>Frame 10</th>
<th>Frame 11</th>
<th>Frame 12</th>
<th>Frame 13</th>
<th>Frame 14</th>
<th>Frame 15</th>
<th>Frame 16</th>
<th>Frame 17</th>
</tr>
</thead>
</table>

http://java.sun.com/docs/books/tutorial
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);
        }
    }
}
# Operating System Specific Details

## Common OS abstractions.

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

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.

```java
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. $2^{20} = 1$ million
Digital Michelangelo Project

**Goal.** Precise 3D description of the David.
- Laser rangefinder.
- 5,000 hours of scanning, 32 Gigabytes!