1.1 Basic Programming Model – 1.2 Data Abstraction

http://www.csc.villanova.edu/~map/2053
This presentation is adapted from slides accompanying *Algorithms 4/e* by Robert Sedgewick and Kevin Wayne and from *Introduction to Programming in Java: An Interdisciplinary Approach* by Robert Sedgewick and Kevin Wayne
1.5 Input and Output
2.2 Libraries and Clients
3.1 Using Data Types
1.1 Basic Programming Model – 1.2 Data Abstraction

http://www.csc.villanova.edu/~map/2053
This presentation is adapted from slides accompanying Algorithms 4/e by Robert Sedgwick and Kevin Wayne and from Introduction to Programming in Java: An Interdisciplinary Approach by Robert Sedgwick and Kevin Wayne
**Library.** A module whose methods are primarily intended for use by many other programs.

**Client.** Program that calls a library.

**API.** Contract between client and implementation.

**Implementation.** Program that implements the methods in an API.
public class RandomPoints {
    public static void main(String args[]) {
        int N = Integer.parseInt(args[0]);
        for (int i = 0; i < N; i++) {
            double x = StdRandom.gaussian(0.5, 0.2);
            double y = StdRandom.gaussian(0.5, 0.2);
            StdDraw.point(x, y);
        }
    }
}

% javac RandomPoints.java
% java RandomPoints 10000
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.** 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
Bird's Eye View

standard input

command-line arguments

standard output

standard audio

standard drawing
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.

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void print(String s)</code></td>
<td>print <code>s</code></td>
</tr>
<tr>
<td><code>void println(String s)</code></td>
<td>print <code>s</code>, followed by newline</td>
</tr>
<tr>
<td><code>void println()</code></td>
<td>print a new line</td>
</tr>
<tr>
<td><code>void printf(String f, ...)</code></td>
<td>formatted print</td>
</tr>
</tbody>
</table>
Standard Drawing

Standard drawing. **StdDraw** is a 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) reset x range to (x0, x1)
void setYscale(double y0, double y1) reset y range to (y0, y1)
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 .png file

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

http://introcs.cs.princeton.edu/java/15inout/USA.txt
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
}
```
Bird's Eye View (Revisited)

- **input streams**
- **standard input**
- **command-line arguments**
- **pictures**
- **drawings**
- **output streams**
- **standard output**
Non-Standard Input

Standard input. Read from terminal window.
Goal. Read from several different input streams.

In data type. Read text from stdin, a file, a web site, or network.

Ex: Are two text files identical?

```java
public class Diff {
    public static void main(String[] args) {
        In in0 = new In(args[0]); ← read from one file
        In in1 = new In(args[1]); ← read from another file
        String s = in0.readLine();
        String t = in1.readLine();
        StdOut.println(s.equals(t));
    }
}
```
Goal. Find current stock price of Google.

http://finance.yahoo.com/q?s=goog

NYSE symbol
Screen Scraping

**Goal.** Find current stock price of Google.

- `s.indexOf(t, i)`: index of first occurrence of pattern `t` in string `s`, starting at offset `i`.
- Find first string delimited by `<b>` and `</b>` after *Last Trade*.

```java
public class StockQuote {
    public static void main(String[] args) {
        String name = "http://finance.yahoo.com/q?s=";
        In in = new In(name + args[0]);
        String input = in.readAll();
        int start = input.indexOf("Last Trade:", 0);
        int from = input.indexOf("<b>", start);
        int to = input.indexOf("</b>", from);
        String price = input.substring(from + 3, to);
        StdOut.println(price);
    }
}
```

% java StockQuote goog
576.50
Try this.

- Find updated version of `StockQuote.java` online: [http://introcs.cs.princeton.edu/java/31datatype/StockQuote.java](http://introcs.cs.princeton.edu/java/31datatype/StockQuote.java)
- Run it from the command line to check some stock prices, eg:
  ```java
  java-algs4 StockQuote aapl
  ```
- Modify it to prompt and get input from user using `StdIn` and `StdOut` (instead of command line args[0] and `System.out.println()`)
- Put in a loop so it keeps asking for more stock prices
  - use `StdIn.isEmpty()` to test for end of input
  - test the program: indicate the end of input using Ctrl-C
Day Trader

(Continued)

- If you have been using an IDE, run the program again from command line
- Run from command line, but this time redirect Standard Input to come from a text file:
  - Create a small text file with some stock tickers in it, save it as tickers.txt
  - Run StockQuote with input from this file:
    ```
    java-algs4 StockQuote <tickers.txt
    ```
- You can also redirect Standard output:
  ```
  java-algs4 StockQuote >tickers.out
  ```
  (creates file tickers.out with output from program – you still need to type in your input, but the prompts and other output is going to tickers.out).
- or both!
  ```
  java-algs4 StockQuote <tickers.txt >tickers.out
 .java-algs4 StockQuote >tickers.out <tickers.txt
  ```
  (either order works)
Add bells and whistles at own risk...

- Plot price in real-time.
- Notify user if price dips below a certain price.
- Embed logic to determine when to buy and sell.
- Automatically send buy and sell orders to trading firm.

**Warning.** Please, please use at your own financial risk.
Statistics
Ex. Library to compute statistics on an array of real numbers.

```
public class StdStats {
    double max(double[] a) { /* largest value */ }
    double min(double[] a) { /* smallest value */ }
    double mean(double[] a) { /* average */ }
    double var(double[] a) { /* sample variance */ }
    double stddev(double[] a) { /* sample standard deviation */ }
    double median(double[] a) { /* median */ }
    void plotPoints(double[] a) { /* plot points at (i, a[i]) */ }
    void plotLines(double[] a) { /* plot lines connecting points at (i, a[i]) */ }
    void plotBars(double[] a) { /* plot bars to points at (i, a[i]) */ }
}
```

\[
\mu = \frac{a_0 + a_1 + \cdots + a_{n-1}}{n}, \quad \sigma^2 = \frac{(a_0 - \mu)^2 + (a_1 - \mu)^2 + \cdots + (a_{n-1} - \mu)^2}{n - 1}
\]

*mean* \quad *sample variance*
Ex. Library to compute statistics on an array of real numbers.

```java
public class StdStats {

    public static double max(double[] a) {
        double max = Double.NEGATIVE_INFINITY;
        for (int i = 0; i < a.length; i++)
            if (a[i] > max) max = a[i];
        return max;
    }

    public static double mean(double[] a) {
        double sum = 0.0;
        for (int i = 0; i < a.length; i++)
            sum = sum + a[i];
        return sum / a.length;
    }

    public static double stddev(double[] a) {
        // see text
    }
}
```
(Continued – back for some stats!)

- Store the prices in an array, as they are being accessed.  
  (if necessary, use java.util.Arrays.copyOf() to trim the array to size after all  
  the values are there)

- Use StdStats.mean(prices) and StdStats.stddev(prices) to obtain the mean and  
  standard deviation of some stocks

- try some of the other StdStats methods.
The generation of random numbers is far too important to leave to chance. Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.

Jon von Neumann (left), ENIAC (right)
Random Numbers

A weighted random number generator just produced a new batch of numbers.

Let's use them to build narratives!

All sports commentary
Standard Random

**Standard random.** Our library to generate pseudo-random numbers.

```java
public class StdRandom {
    int uniform(int N) // integer between 0 and N-1
    double uniform(double lo, double hi) // real between lo and hi
    boolean bernoulli(double p) // true with probability p
    double gaussian() // normal, mean 0, standard deviation 1
    double gaussian(double m, double s) // normal, mean m, standard deviation s
    int discrete(double[] a) // i with probability a[i]
    void shuffle(double[] a) // randomly shuffle the array a[]
}
```

```java
int getRandomNumber() {
    return 4; // chosen by fair dice roll.
    // guaranteed to be random.
}
```
public class StdRandom {

    // between a and b
    public static double uniform(double a, double b) {
        return a + Math.random() * (b-a);
    }

    // between 0 and N-1
    public static int uniform(int N) {
        return (int) (Math.random() * N);
    }

    // true with probability p
    public static boolean bernoulli(double p) {
        return Math.random() < p;
    }

    // gaussian with mean = 0, stddev = 1
    public static double gaussian() {
        /* see Exercise 1.2.27 */
    }

    // gaussian with given mean and stddev
    public static double gaussian(double mean, double stddev) {
        return mean + (stddev * gaussian());
    }

    ...
}

Standard Random
Earlier example: Using a Library

```java
public class RandomPoints {
    public static void main(String args[]) {
        int N = Integer.parseInt(args[0]);
        for (int i = 0; i < N; i++) {
            double x = StdRandom.gaussian(0.5, 0.2);
            double y = StdRandom.gaussian(0.5, 0.2);
            StdDraw.point(x, y);
        }
    }
}
```

% javac RandomPoints.java
% java RandomPoints 10000
Unit Testing

**Unit test.** Include `main()` to test each library.

```java
public class StdRandom {
    
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 0; i < N; i++) {
            StdOut.printf("%2d ", uniform(100));
            StdOut.printf("%8.5f ", uniform(10.0, 99.0));
            StdOut.printf("%5b ", bernoulli(.5));
            StdOut.printf("%7.5f ", gaussian(9.0, .2));
            StdOut.println();
        }
    }
}
```

```bash
% java StdRandom 5
61 21.76541  true  9.30910
57 43.64327  false  9.42369
31 30.86201  true  9.06366
92 39.59314  true  9.00896
36 28.27256  false  8.66800
```