5. Functions and Libraries
5. Functions and Libraries

- Basic concepts
- Case study: Digital audio
- Application: Gaussian distribution
- Modular programming and libraries
Context: basic building blocks for programming

any program you might want to write

objects

functions and libraries

graphics, sound, and image I/O

arrays

conditionals and loops

Math

text I/O

primitive data types

assignment statements

This lecture: Reuse code to build big programs from small pieces
Functions, libraries, and modules

Modular programming
• Organize programs as independent modules that do a job together.
• Why? Easier to share and reuse code to build bigger programs.

Facts of life
• Support of modular programming has been a holy grail for decades.
• Ideas can conflict and get highly technical in the real world.

Def. A library is a set of functions. for purposes of this lecture

Def. A module is a .java file. for purposes of this course

For now. Libraries and modules are the same thing: .java files containing sets of functions.

Later. Modules implement data types (stay tuned).
Java function ("aka static method")
- Takes zero or more input arguments.
- Returns zero or one output value.
- May cause side effects (e.g., output to standard draw).

Applications
- Scientists use mathematical functions to calculate formulas.
- Programmers use functions to build modular programs.
- You use functions for both.

Examples seen so far
- Built-in functions: Math.random(), Math.abs(), Integer.parseInt().
- Our I/O libraries: StdIn.readInt(), StdDraw.line(), StdAudio.play().
- User-defined functions: main().
Anatomy of a Java static method

To implement a function (static method)

- Create a name.
- Declare type and name of argument(s).
- Specify type for return value.
- Implement body of method.
- Finish with return statement.

```java
public static double sqrt(double c, double eps) {
    if (c < 0) return Double.NaN;
    double t = c;
    while (Math.abs(t - c/t) > eps * t) {
        t = (c/t + t) / 2.0;
    }
    return t;
}
```
Anatomy of a Java library

A **library** is a set of functions.

```java
public class Newton {
    public static double sqrt(double c, double eps) {
        if (c < 0) return Double.NaN;
        double t = c;
        while (Math.abs(t - c/t) > eps * t)
            t = (c/t + t) / 2.0;
        return t;
    }

    public static void main(String[] args) {
        double[] a = new double[args.length];
        for (int i = 0; i < args.length; i++)
            a[i] = Double.parseDouble(args[i]);
        for (int i = 0; i < a.length; i++)
            StdOut.println(sqrt(a[i], 1e-3));
    }
}
```

**Note:** We are using our `sqrt()` from Lecture 2 here to illustrate the basics with a familiar function.

Our focus is on control flow here. See Lecture 2 for technical details.

You can use `Math.sqrt()`.

**Key point.** Functions provide a *new way* to control the flow of execution.
Scope

**Def.** The *scope* of a variable is the code that can refer to it by name.

```java
public class Newton {
    public static double sqrt(double c, double eps) {
        if (c < 0) return Double.NaN;
        double t = c;
        while (Math.abs(t - c/t) > eps * t) {
            t = (c/t + t) / 2.0;
        }
        return t;
    }

    public static void main(String[] args) {
        double[] a = new double[args.length];
        for (int i = 0; i < args.length; i++) {
            a[i] = Double.parseDouble(args[i]);
        }
        for (int i = 0; i < a.length; i++) {
            StdOut.println(sqrt(a[i], 1e-3));
        }
    }
}
```

In a Java library, a variable's scope is the code following its declaration, in the same block.

Best practice. Declare variables so as to *limit* their scope.
Flow of control

public class Newton
{
    public static double sqrt(double c, double eps)
    {
        if (c < 0) return Double.NaN;
        double t = c;
        while (Math.abs(t - c/t) > eps * t)
            t = (c/t + t) / 2.0;
        return t;
    }

    public static void main(String[] args)
    {
        double[] a = new double[args.length];
        for (int i = 0; i < args.length; i++)
            a[i] = Double.parseDouble(args[i]);
        for (int i = 0; i < a.length; i++)
        {
            double x = sqrt(a[i], 1e-3);
            StdOut.println(x);
        }
    }
}
Function call flow of control trace

```java
public class Newton {
    public static double sqrt(double c, double eps) {
        if (c < 0) return Double.NaN;
        double t = c;
        while (Math.abs(t - c/t) > eps * t) {
            t = (c/t + t) / 2.0;
            return t;
        }
    }
    public static void main(String[] args) {
        double[] a = new double[args.length];
        for (int i = 0; i < args.length; i++)
            a[i] = Double.parseDouble(args[i]);
        for (int i = 0; i < a.length; i++) {
            double x = sqrt(a[i], 1e-3);
            StdOut.println(x);
        }
    }
}
```

<table>
<thead>
<tr>
<th>c</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>1.75</td>
<td></td>
</tr>
<tr>
<td>1.732</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>i</th>
<th>a[i]</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0</td>
<td>1.000</td>
</tr>
<tr>
<td>1</td>
<td>2.0</td>
<td>1.414</td>
</tr>
<tr>
<td>2</td>
<td>3.0</td>
<td>1.732</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

% java Newton 1 2 3
1.000
1.414
1.732
Pop quiz 1a on functions

Q. What happens when you compile and run the following code?

```java
class PQfunctions1a {
    public static int cube(int i) {
        int j = i * i * i;
        return j;
    }

    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```
Pop quiz 1a on functions

Q. What happens when you compile and run the following code?

```java
public class PQfunctions1a {
    public static int cube(int i)
    {
        int j = i * i * i;
        return j;
    }
    public static void main(String[] args)
    {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```

A. Takes N from the command line, then prints cubes of integers from 1 to N

```
% javac PQfunctions1a.java
% java PQfunctions1a 6
1 1
2 8
3 27
4 64
5 125
6 216
```
Pop quiz 1b on functions

Q. What happens when you compile and run the following code?

```java
public class PQfunctions1b {
    public static int cube(int i) {
        int i = i * i * i;
        return i;
    }
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```
Pop quiz 1b on functions

Q. What happens when you compile and run the following code?

```java
public class PQfunctions1b {
    public static int cube(int i) {
        int i = i * i * i;
        return i;
    }
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```

A. Won't compile. Argument variable i is declared and initialized for function block, so the name cannot be reused.

```
javac PQfunctions1b.java
PQfunctions1b.java:5: i is already defined in cube(int)
    int i = i * i * i;
    ^
1 error
```
Pop quiz 1c on functions

Q. What happens when you compile and run the following code?

```java
public class PQfunctions1c {
    public static int cube(int i) {
        i = i * i * i;
    }
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```
Pop quiz 1c on functions

Q. What happens when you compile and run the following code?

```java
public class PQ6_1c {
    public static int cube(int i) {
        i = i * i * i;
    }
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```

A. Won't compile. Need return statement.

```
% javac PQfunctions1c.java
PQfunctions1c.java:6: missing return statement
    }
^ 1 error
```
Pop quiz 1d on functions

Q. What happens when you compile and run the following code?

```java
public class PQfunctions1d {
    public static int cube(int i) {
        i = i * i * i;
        return i;
    }
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```
Pop quiz 1d on functions

Q. What happens when you compile and run the following code?

```java
public class PQfunctions1d {
    public static int cube(int i) {
        i = i * i * i;
        return i;
    }
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```

A. Works. The `i` in `cube()` is
- Declared and initialized as an argument.
- Different from the `i` in `main()`.

BUT changing values of function arguments is sufficiently confusing to be deemed bad style for this course.

```
% javac PQfunctions1d.java
% java PQfunctions1d 6
1 1
2 8
3 27
4 64
5 125
6 216
```
Q. What happens when you compile and run the following code?

```java
public class PQfunctions1e {
    public static int cube(int i) {
        return i * i * i;
    }
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```
Pop quiz 1e on functions

Q. What happens when you compile and run the following code?

```java
public class PQfunctions1e
{
    public static int cube(int i)
    {
        return i * i * i;
    }
    public static void main(String[] args)
    {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + " " + cube(i));
    }
}
```

A. Works fine. Preferred (compact) code.

```bash
% javac PQfunctions1e.java
% java PQfunctions1e 6
1 1
2 8
3 27
4 64
5 125
6 216
```
Image sources

http://upload.wikimedia.org/wikipedia/commons/b/ba/Working_Together_Teamwork_Puzzle_Concept.jpg
http://upload.wikimedia.org/wikipedia/commons/e/ef/Ben_Jigsaw_Puzzle_Puzzle_Puzzle.png
5. Functions and Libraries

- Basic concepts
- Case study: Digital audio
- Application: Gaussian distribution
- Modular programming
Crash course in sound

**Sound** is the perception of the vibration of molecules.

A **musical tone** is a steady periodic sound.

A **pure tone** is a sinusoidal waveform.

### Western musical scale
- Concert A is 440 Hz.
- 12 notes, logarithmic scale.

<table>
<thead>
<tr>
<th>pitch</th>
<th>i</th>
<th>frequency (440*2^i/12)</th>
<th>sinusodial waveform</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>440</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
<tr>
<td>A# / B♭</td>
<td>1</td>
<td>466.16</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>493.88</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>523.25</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
<tr>
<td>C# / D♭</td>
<td>4</td>
<td>554.37</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>587.33</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
<tr>
<td>D# / E♭</td>
<td>6</td>
<td>622.25</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
<tr>
<td>E</td>
<td>7</td>
<td>659.26</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
<tr>
<td>F</td>
<td>8</td>
<td>698.46</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
<tr>
<td>F# / G♭</td>
<td>9</td>
<td>739.99</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
<tr>
<td>G</td>
<td>10</td>
<td>783.99</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
<tr>
<td>G# / A♭</td>
<td>11</td>
<td>830.61</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
<tr>
<td>A</td>
<td>12</td>
<td>880</td>
<td><img src="https://via.placeholder.com/150" alt="Waveform" /></td>
</tr>
</tbody>
</table>
Digital audio

To represent a wave, *sample* at regular intervals and save the values in an array.

<table>
<thead>
<tr>
<th>samples/sec</th>
<th>samples</th>
<th>sampled waveform</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,512</td>
<td>137</td>
<td><img src="image" alt="Sample waveform" /></td>
</tr>
<tr>
<td>11,025</td>
<td>275</td>
<td><img src="image" alt="Sample waveform" /></td>
</tr>
<tr>
<td>22,050</td>
<td>551</td>
<td><img src="image" alt="Sample waveform" /></td>
</tr>
<tr>
<td>44,100</td>
<td>1102</td>
<td><img src="image" alt="Sample waveform" /></td>
</tr>
</tbody>
</table>

1/40 second of concert A

CD standard

**Bottom line.** You can *write programs* to manipulate sound (arrays of double values).
**StdAudio library**

*Developed for this course, also broadly useful*

- Play a sound wave (array of double values) on your computer’s audio output.
- Convert to and from standard .wav file format.

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public class StdAudio</td>
<td></td>
</tr>
<tr>
<td>void play(String file)</td>
<td><em>play the given .wav file</em></td>
</tr>
<tr>
<td>void play(double[] a)</td>
<td><em>play the given sound wave</em></td>
</tr>
<tr>
<td>void play(double x)</td>
<td><em>play the sample for 1/44100 second</em></td>
</tr>
<tr>
<td>void save(String file, double[] a)</td>
<td><em>save to a .wav file</em></td>
</tr>
<tr>
<td>double[] read(String file)</td>
<td><em>read from a .wav file</em></td>
</tr>
</tbody>
</table>

Enables you to *hear the results* of your programs that manipulate sound.
public class PlayThatNote
{
    public static double[] tone(double hz, double duration)
    {
        int N = (int) (44100 * duration);
        double[] a = new double[N+1];
        for (int i = 0; i <= N; i++)
            a[i] = Math.sin(2 * Math.PI * i * hz / 44100);
        return a;
    }
    public static void main(String[] args)
    {
        double hz = Double.parseDouble(args[0]);
        double duration = Double.parseDouble(args[1]);
        double[] a = tone(hz, duration);
        StdAudio.play(a);
    }
}
Play that tune

Read a list of tones and durations from standard input to **play a tune.**

```java
public class PlayThatTune {
    public static void main(String[] args) {
        double tempo = Double.parseDouble(args[0]);
        while (!StdIn.isEmpty()) {
            int pitch = StdIn.readInt();
            double duration = StdIn.readDouble() * tempo;
            double hz = 440 * Math.pow(2, pitch / 12.0);
            double[] a = PlayThatNote.tone(hz, duration);
            StdAudio.play(a);
        }
        StdAudio.close();
    }
}
```
Pop quiz 2 on functions

**Q.** What sound does the following program produce?

```java
public class PQfunctions2
{
   public static void main(String[] args)
   {
      int N = (int) (44100 * 11);
      double[] a = new double[N+1];
      for (int i = 0; i <= N; i++)
         a[i] = Math.random();
      StdAudio.play(a);
   }
}
```
Pop quiz 2 on functions

Q. What sound does the following program produce?

```java
public class PQfunctions2 {
    public static void main(String[] args) {
        int N = (int) (44100 * 11.0);
        double[] a = new double[N+1];
        for (int i = 0; i <= N; i++)
            a[i] = Math.random();
        StdAudio.play(a);
    }
}
```

A. 11 seconds of pure noise.
Play that chord

Produce chords by *averaging* waveforms.

```java
public static double[] avg(double[] a, double[] b) {
    double[] c = new double[a.length];
    for (int i = 0; i < a.length; i++)
        c[i] = a[i]/2.0 + b[i]/2.0;
    return c;
}
```

All values stay between 0 and 1
public class PlayThatChord
{
    public static double[] avg(double[] a, double[] b)
    { /* See previous slide. */ }

    public static double[] chord(int pitch1, int pitch2, double d)
    {
        double hz1 = 440.0 * Math.pow(2, pitch1 / 12.0);
        double hz2 = 440.0 * Math.pow(2, pitch2 / 12.0);
        double[] a = PlayThatNote.tone(hz1, d);
        double[] b = PlayThatNote.tone(hz2, d);
        return avg(a, b);
    }

    public static void main(String[] args)
    {
        int pitch1 = Integer.parseInt(args[0]);
        int pitch2 = Integer.parseInt(args[1]);
        double duration = Double.parseDouble(args[2]);
        double[] a = chord(pitch1, pitch2, duration);
        StdAudio.play(a);
    }
}
Play that tune (deluxe version)

Add harmonics to PlayThatTune to produce a more realistic sound.

**Function to add harmonics to a tone**

```java
public static double[] note(int pitch, double duration) {
    double hz = 440.0 * Math.pow(2, pitch / 12.0);
    double[] a = tone(1.0 * hz, duration);
    double[] hi = tone(2.0 * hz, duration);
    double[] lo = tone(0.5 * hz, duration);
    double[] harmonic = sum(hi, lo);
    return avg(a, harmonic);
}
```

% java PlayThatTune 1.5 < elise.txt

Program 2.1.4 in text (with tempo added)
Bottom line. *You* can write programs to manipulate sound.
This lecture: Case study of the utility of functions.
Upcoming assignment: Fun with musical tones.
Image sources

http://commons.wikimedia.org/wiki/File:1405_Sound_Waves_and_the_Ear.jpg
http://cantorion.org/music/497/The-Entertainer-Original-version
5. Functions and Libraries

• Basic concepts
• Case study: Digital audio
• Application: Gaussian distribution
• Modular programming
Gaussian distribution.

- A mathematical model used successfully for centuries.
- "Bell curve" fits experimental observations in many contexts.

Gaussian probability density function (pdf)

\[ \phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \]

Example: SAT scores in 20xx (verbal + math)

\[ \phi(x, \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)} \]

mean (\(\mu\)): 1019
stddev (\(\sigma\)): 209

mean (\(\mu\)): 0
stddev (\(\sigma\)): 1
Gaussian distribution in the wild

Polystyrene particles in glycerol

Calibration of optical tweezers

Laser beam propagation

Predicted US oil production

German money

Polarized W bosons from top-quark decay

Cytochrome oxidase patches in macaque primary visual cortex

Population mean tuning curves

Spikes per second

Difference from peak orientation (degrees)

Patch, n = 177

Interpatch, n = 419

Vermaan money

Money
Defining a library of functions

Q. Is the Gaussian pdf $\Phi$ implemented in Java's Math library?  
A. No.

Q. Why not?  
A. Maybe because it is so easy for you to do it yourself.

```java
public class Gaussian {
    public static double pdf(double x) {
        return Math.exp(-x*x / 2) / Math.sqrt(2 * Math.PI);
    }

    public static double pdf(double x, double mu, double sigma) {
        return pdf((x - mu) / sigma) / sigma;
    }

    // Stay tuned for more functions.
}
```

Functions and libraries provide an easy way for any user to extend the Java system.
Gaussian cumulative distribution function

Q. What percentage of the total is less than or equal to $z$?
Q. (equivalent). What is the area under the curve to the left of $z$?
A. Gaussian cumulative distribution function.

\[ \Phi(x, \mu, \sigma) = \Phi\left(\frac{x - \mu}{\sigma}\right) \]

Gaussian probability density function (pdf)

\[ \phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \]

Gaussian cumulative distribution function (cdf)

\[ \Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-x^2/2} \, dx \]
Typical application: SAT scores

Q. In 20xx NCAA required at least 820 for Division I athletes. What fraction of test takers did not qualify??

A. About 17%, since \( \Phi(820, 1019, 209) = 0.17050966869132111... \)
Gaussian CDF implementation

**Q.** No closed form for Gaussian CDF $\Phi$. How to implement?

**A. Use Taylor series.** $\Phi(z) \equiv \int_{-\infty}^{z} \phi(x)dx = \frac{1}{2} + \phi(z)\left(z + \frac{z^3}{3} + \frac{z^5}{3 \cdot 5} + \frac{z^7}{3 \cdot 5 \cdot 7} + \ldots\right)$

```java
public static double cdf(double z)
{
    if (z < -8.0) return 0.0;
    if (z >  8.0) return 1.0;
    double sum = 0.0, term = z;
    for (int i = 3; sum + term != sum; i += 2) {
        sum = sum + term;
        term = term * z * z / i;
    }
    return 0.5 + sum * pdf(z); ← accurate to 15 places
}
```

```java
public static double cdf(double z, double mu, double sigma)
{
    return cdf((z - mu) / sigma);
}
```

**Bottom line.** 1,000 years of mathematical formulas at your fingertips.
Summary: a library for Gaussian distribution functions

Best practice
- Test all code at least once in main().
- Also have it do something useful.

Q. What fraction of SAT test takers did not qualify for NCAA participation in 20xx?

% java Gaussian 820 1019 209
0.17050966869132111

Fun fact
We use cdf() to evaluate randomness in submitted programs.

Bottom line
YOU can build a layer of abstraction to use in any future program.

```java
public class Gaussian {
    public static double pdf(double x) {
        return Math.exp(-x*x / 2) / Math.sqrt(2 * Math.PI);
    }
    public static double pdf(double x, double mu, double sigma) {
        return pdf((x - mu) / sigma) / sigma;
    }
    public static double cdf(double z) {
        if (z < -8.0) return 0.0;
        if (z > 8.0) return 1.0;
        double sum = 0.0, term = z;
        for (int i = 3; sum + term != sum; i += 2) {
            sum = sum + term;
            term = term * z * z / i;
        }
        return 0.5 + sum * pdf(z);
    }
    public static double cdf(double z, double mu, double sigma) {
        return cdf((z - mu) / sigma);
    }
    public static void main(String[] args) {
        double z = Double.parseDouble(args[0]);
        double mu = Double.parseDouble(args[1]);
        double sigma = Double.parseDouble(args[2]);
        StdOut.println(cdf(z, mu, sigma));
    }
}
```
Using a library

To use these methods in another program

- Put a copy of Gaussian.java in your working directory.
- Call Gaussian.pdf() or Gaussian.cdf() from any other module in that directory.

Example. Draw a plot of $\phi(x, 0, 1)$ in $(-4, 4)$

```java
public class GaussianPlot {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        StdDraw.setXscale(-4.0, +4.0);
        StdDraw.setYscale(0, .5);
        StdDraw.setPenRadius(0.01);
        double[] x = new double[N+1];
        double[] y = new double[N+1];
        for (int i = 0; i <= N; i++) {
            x[i] = -4.0 + 8.0 * i / N;
            y[i] = Gaussian.pdf(x[i]);
        }
        for (int i = 0; i < N; i++)
            StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
    }
}
```

Libraries of functions provide an easy way for any user (you) to extend the Java system.
Image sources

http://www.intechopen.com/books/rheology-new-concepts-applications-and-methods/a-practical-review-of-microrheological-techniques
http://tweezpal.natan.si
http://www.albartlett.org/articles/art2000jan.html
http://laser.physics.sunysb.edu/~vwang/beam-propagation/
http://www.nature.com/neuro/journal/v14/n12/full/nn.2958.html
http://www-cdf.fnal.gov/physics/new/top/2012/WHe19ifb/
5. Functions and Libraries

- Basic concepts
- Case study: Digital audio
- Application: Gaussian distribution
- Modular programming
Fundamental abstractions for modular programming

**Client**
Module that calls a library's methods.

**Applications programming interface (API)**
Defines signatures, describes methods.

**Implementation**
Module containing library's Java code.

```java
public class GaussianPlot {
    ...
    y[i] = Gaussian.pdf(x[i]);
    ...
}
```

```java
public class Gaussian {
    public static double pdf(double x) {
        double val = Math.exp(-x*x / 2);
        val /= Math.sqrt(2 * Math.PI);
        return val;
    }
    ...
}
```
**Example: StdRandom library**

Developed for this course, but broadly useful
- Implement methods for generating random numbers of various types.
- Available for download at booksite (and included in introcs software).

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int uniform(int N)</td>
<td>integer between 0 and N-1</td>
</tr>
<tr>
<td>double uniform(double lo, double hi)</td>
<td>real between lo and hi</td>
</tr>
<tr>
<td>boolean bernoulli(double p)</td>
<td>true with probability p</td>
</tr>
<tr>
<td>double gaussian()</td>
<td>normal with mean 0, stddev 1</td>
</tr>
<tr>
<td>double gaussian(double m, double s)</td>
<td>normal with mean m, stddev s</td>
</tr>
<tr>
<td>int discrete(double[] a)</td>
<td>i with probability a[i]</td>
</tr>
<tr>
<td>void shuffle(double[] a)</td>
<td>randomly shuffle the array a[]</td>
</tr>
</tbody>
</table>

First step in developing a library: **Articulate the API!**
StdRandom details

Implementation

```java
public class StdRandom {
    public static double uniform(double a, double b) {
        return a + Math.random() * (b-a); }
    public static int uniform(int N) {
        return (int) (Math.random() * N); }
    public static boolean bernoulli(double p) {
        return Math.random() < p; }
    public static double gaussian() /* see Exercise 1.2.27 */ {
        return mean + (stddev * gaussian()); }
    public static double gaussian(double m, double s) {
        return mean + (stddev * gaussian()); }
    ...
}
```

Typical client

```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);
        }
    }
}
```

You could implement many of these methods, but now you don't have to!

% java RandomPoints 10000
Best practices

Small modules
• Separate and classify small tasks.
• Implement a layer of abstraction.

Independent development
• Code client before coding implementation.
• Anticipate needs of future clients.

Test clients
• Include main() test client in each module.
• Do more extensive testing in a separate module.

```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("%8.5f ", bernoulli(.5));
            StdOut.printf("%8.5f ", gaussian(9.0, .2));
            StdOut.println();
        }
    }
}
```

% 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

run all code at least once!
**Example: StdStats library**

Developed for this course, but broadly useful

- Implement methods for computing statistics on arrays of real numbers.
- Available for download at book site (and included in introcs software).

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public class StdStats</td>
<td></td>
</tr>
<tr>
<td>double max(double[] a)</td>
<td>largest value</td>
</tr>
<tr>
<td>double min(double[] a)</td>
<td>smallest value</td>
</tr>
<tr>
<td>double mean(double[] a)</td>
<td>average</td>
</tr>
<tr>
<td>double var(double[] a)</td>
<td>sample variance</td>
</tr>
<tr>
<td>double stddev(double[] a)</td>
<td>sample standard deviation</td>
</tr>
<tr>
<td>double median(double[] a)</td>
<td>plot points at (i, a[i])</td>
</tr>
<tr>
<td>void plotPoints(double[] a)</td>
<td>plot points at (i, a[i])</td>
</tr>
<tr>
<td>void plotLines(double[] a)</td>
<td>plot lines connecting points at (i, a[i])</td>
</tr>
<tr>
<td>void plotBars(double[] a)</td>
<td>plot bars to points at (i, a[i])</td>
</tr>
</tbody>
</table>

Easy to implement, but easier to use! — one reason to develop a library: clarify client code
Example of modular programming: StdStats, StdRandom, and Gaussian client

**Experiment**
- Flip $N$ coins.
- How many heads?
- Prediction: Expect $N/2$.

**Prediction (more detailed)**
- Run experiment *trials* times.
- How many heads?

```java
public static int binomial(int N) {
    int heads = 0;
    for (int j = 0; j < N; j++)
        if (StdRandom.bernoulli(0.5))
            heads++;
    return heads;
}
```

**Goal.** Write a program to validate predictions.
Example of modular programming: Bernoulli trials

```java
public class Bernoulli {
    public static int binomial(int N) {
        // See previous slide.
        return 0; // Placeholder for binomial function
    }

    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        int trials = Integer.parseInt(args[1]);

        int[] freq = new int[N+1];
        for (int t = 0; t < trials; t++)
            freq[binomial(N)]++;

        double[] normalized = new double[N+1];
        for (int i = 0; i <= N; i++)
            normalized[i] = (double) freq[i] / trials;
        StdStats.plotBars(normalized);

        double mean = N / 2.0;
        double stddev = Math.sqrt(N) / 2.0;
        double[] phi = new double[N+1];
        for (int i = 0; i <= N; i++)
            phi[i] = Gaussian.pdf(i, mean, stddev);
        StdStats.plotLines(phi);
    }
}
```

Bernoulli simulation

- Get command-line arguments (trials experiments of N flips).
- Run experiments. Keep track of frequency of occurrence of each return value.
- Normalize to between 0 and 1. Plot histogram.
- Plot theoretical curve.

% java Bernoulli 20 10000
Modular programming enables development of complicated programs via simple independent modules.

Advantages. Code is easier to understand, debug, maintain, improve, and reuse.
Why modular programming?

Modular programming enables
• Independent development of small programs.
• Every programmer to develop and share layers of abstraction.
• Self-documenting code.

Fundamental characteristics
• Separation of client from implementation benefits all future clients.
• Contract between implementation and clients (API) benefits all past clients.

Challenges
• How to break task into independent modules?
• How to specify API?
Image sources

http://xkcd.com/221/
http://upload.wikimedia.org/wikipedia/commons/b/ba/Working_Together_Teamwork_Puzzle_Concept.jpg
http://upload.wikimedia.org/wikipedia/commons/e/ef/Ben_Jigsaw_Puzzle_Puzzle_Puzzle.png
5. Functions and Libraries