CMSC 150
INTRODUCTION TO COMPUTING

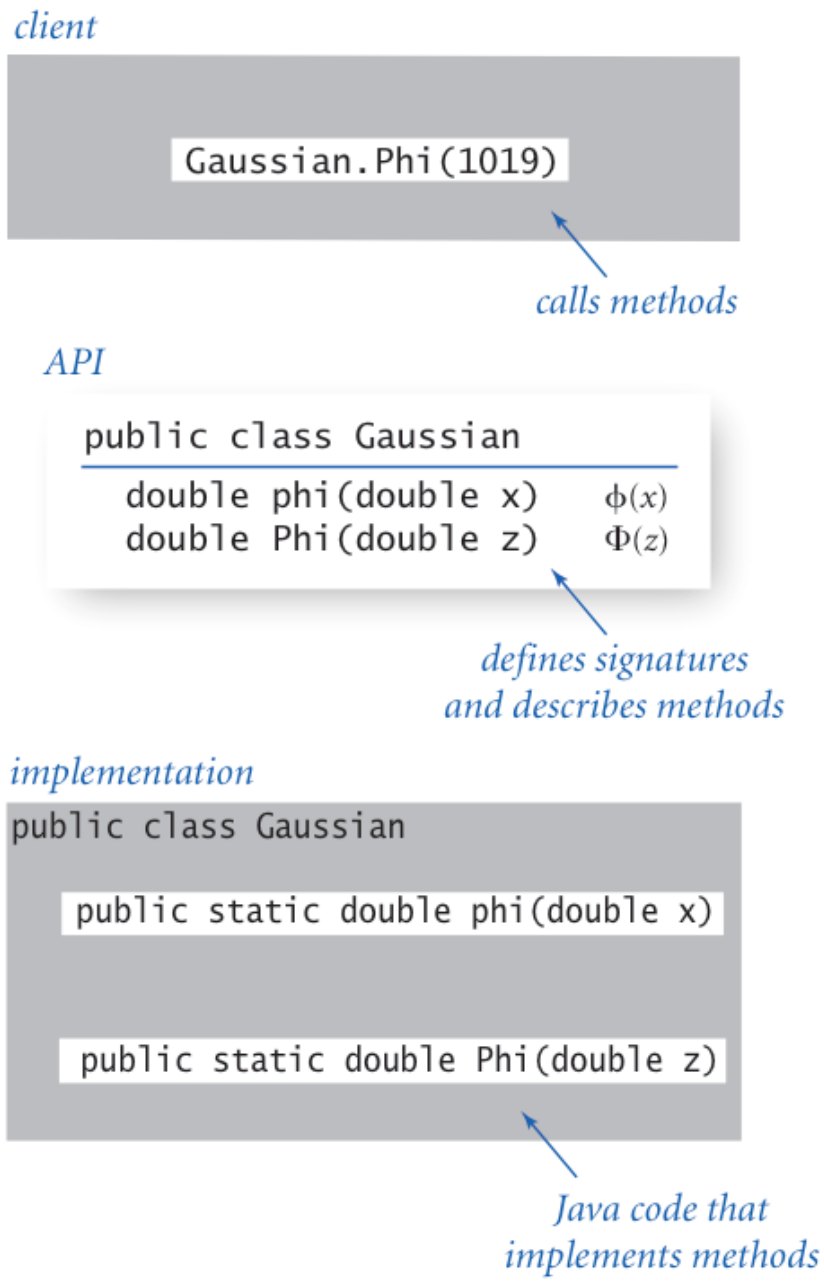
ACKNOWLEDGEMENT: THESE SLIDES ARE ADAPTED FROM SLIDES PROVIDED WITH INTRODUCTION TO PROGRAMMING IN JAVA: AN INTERDISCIPLINARY APPROACH, SEDGEWICK AND WAYNE (PEARSON ADDISON-WESLEY 2007)

LECTURE 6

- MODULES
- LIBRARIES AND CLIENTS

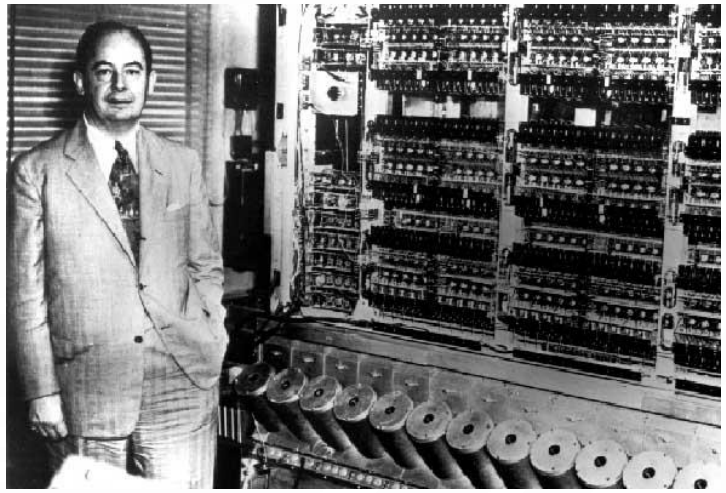
LIBRARIES

- **Library** – A module whose methods are primarily intended for use by many other programs.
- **Client** – Program that calls a library.
- **Application Program Interface (API)** – Contract between client and implementation.
- **Implementation** – Program that implements the methods in an API.



RANDOM NUMBERS

> *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)

STANDARD RANDOM API

- Standard random. Author's library to generate pseudo-random numbers.

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[]

STANDARD RANDOM IMPLEMENTATION

```
1.  public class StdRandom {
2.     // between a and b
3.     public static double uniform(double a,
       double b) {
4.        return a + Math.random() * (b-a);
5.     }
6.     // between 0 and N-1
7.     public static int uniform(int N) {
8.        return (int) (Math.random() * N);
9.     }
10.    // true with probability p
11.    public static boolean bernoulli(double p) {
12.       return Math.random() < p;
13.    }
14.    // gaussian with mean = 0, stddev = 1
15.    public static double gaussian()
16.       /* see Exercise 1.2.27 */
17.    // gaussian with given mean and stddev
18.    public static double gaussian(double
       mean, double stddev) {
19.       return mean + (stddev * gaussian());
20.    }
21.    ...
22. }
```

UNIT TESTING

- **Unit test** – Automated piece of code that invoked a "unit" of work and then checks a single assumption of its behavior. Use main() to test each library.

```
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 \n", gaussian(9.0, .2));
    }
  }
}
```

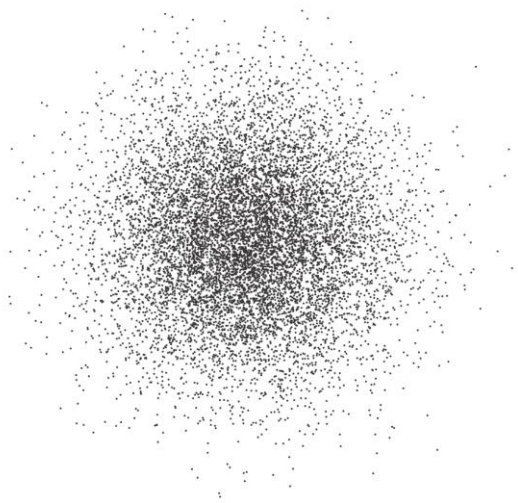
```
% java StdRandom 5
 61 21.76541  true 9.30910
 57 43.64327 false 9.42369
 31 30.86201  true 9.06366
 92 39.59314  true 9.00896
 36 28.27256 false 8.66800
```

USING A LIBRARY

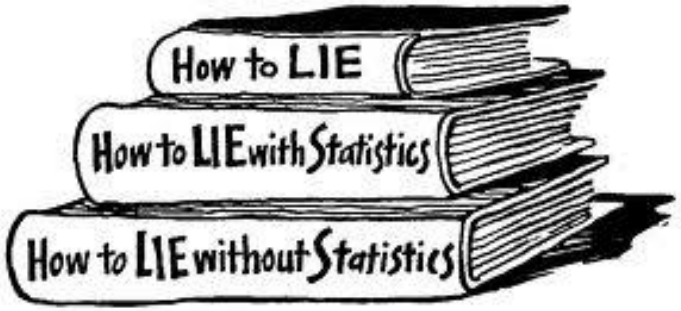
```
1.  public class RandomPoints {
2.    public static void main(String args[]) {
3.      int N = Integer.parseInt(args[0]);
4.      for (int i = 0; i < N; i++) {
5.        double x = StdRandom.gaussian(0.5, 0.2);
6.        double y = StdRandom.gaussian(0.5, 0.2);
7.        StdDraw.point(x, y);
8.      }
9.    }
10. }
```

use library name to invoke method

```
% javac RandomPoints.java
% java RandomPoints 10000
```



EXAMPLE STATISTICS



STANDARD STATISTICS API

- 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])

STANDARD STATISTICS IMPLEMENTATION

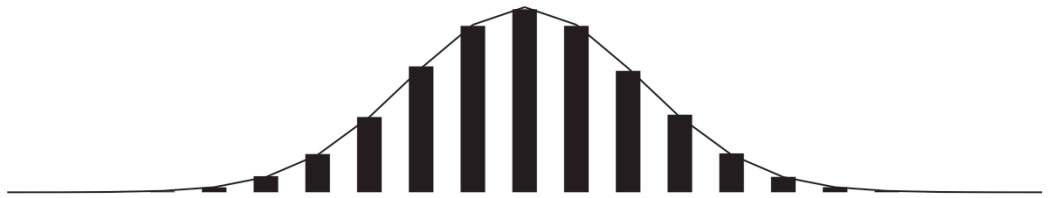
```
1.  public class StdStats {
2.
3.     public static double max(double[] a) {
4.        double max = Double.NEGATIVE_INFINITY;
5.        for (int i = 0; i < a.length; i++)
6.           if (a[i] > max) max = a[i];
7.        return max;
8.     }
9.
10.    public static double mean(double[] a) {
11.       double sum = 0.0;
12.       for (int i = 0; i < a.length; i++)
13.          sum = sum + a[i];
14.       return sum / a.length;
15.    }
16.
17.    public static double stddev(double[] a)
18.       // see text
19. }
```

MODULAR PROGRAMMING

MODULAR PROGRAMMING

- Modular programming.
  - Divide program into self-contained pieces.
  - Test each piece individually.
  - Combine pieces to make program.
  - Allows larger and more complex programs
- Ex. Flip N coins. How many heads?
  - Read arguments from user.
  - Flip one fair coin.
  - Flip N fair coins and count number of heads.
  - Repeat simulation, counting number of times each outcome occurs.
  - Plot histogram of empirical results.
  - Compare with theoretical predictions.

```
% java Bernoulli 20 100000
```



SUMMARY

- Why use libraries?
  - Makes code easier to understand.
  - Makes code easier to debug.
  - Makes code easier to maintain and improve.
  - Makes code easier to reuse.