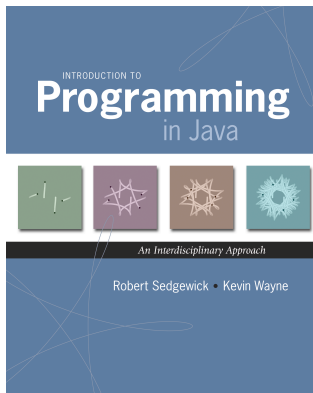
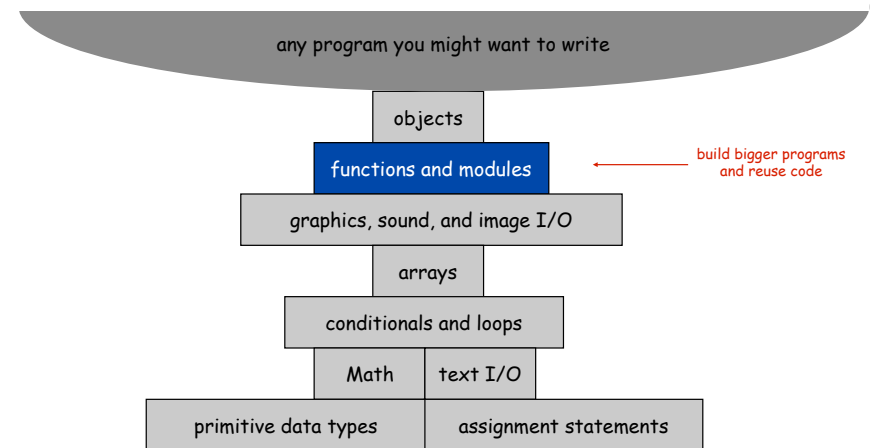


2.1 Functions

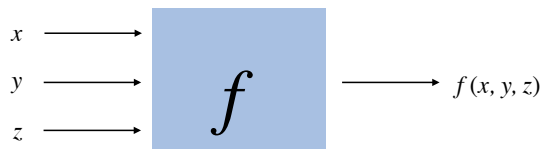


Introduction to Programming in Java: An Interdisciplinary Approach · Robert Sedgewick and Kevin Wayne · Copyright © 2008 · * *



2

2.1 Functions



Functions (Static Methods)

Java function.

- Takes zero or more input arguments.
- Returns one output value (and/or has *side effects*—stay tuned).

Applications.

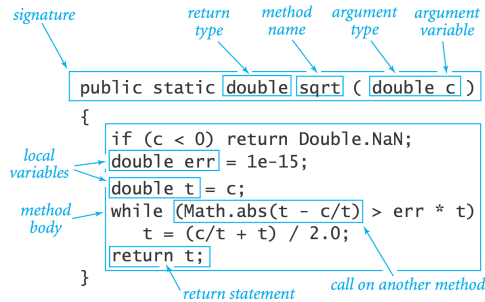
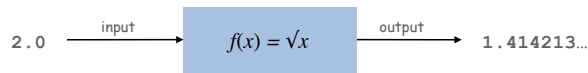
- Scientists use mathematical functions to calculate formulas.
- Programmers use functions to build modular programs.
- **You** use functions for both.

Examples.

- **Built-in functions:** `Math.random()`, `Math.abs()`, `Integer.parseInt()`.
- **Our I/O libraries:** `StdIn.readInt()`, `StdDraw.line()`, `StdAudio.play()`.
- **User-defined functions:** `main()`.

4

Java functions. Easy to write your own.



```
public class Gambler {
    public static void main(String[] args) {
        int stake = Integer.parseInt(args[0]);
        int goal = Integer.parseInt(args[1]);
        int trials = Integer.parseInt(args[2]);
        . . .
        . . .
    }
}
```

main is a function !
 - that has one argument (String array args)
 - that returns no value (hence void)

Scope

Flow of Control

Scope (of a name). The code that can refer to that name.

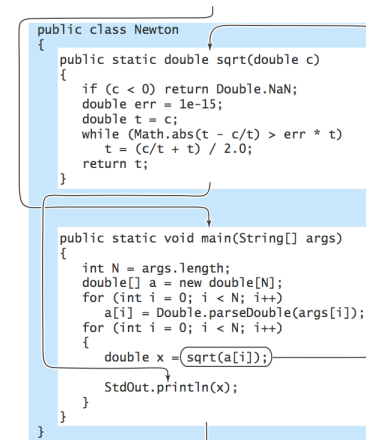
Ex. A variable's scope is code following the declaration in the block.

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

```
public class Newton {
    public static double sqrt(double c) {
        double epsilon = 1e-15;
        if (c < 0) return Double.NaN;
        double t = c;
        while (Math.abs(t - c/t) > epsilon * 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]);
            StdOut.println(x);
        }
    }
}
```

Annotations for scope: "scope of c" (points to `double c`), "scope of epsilon" (points to `double epsilon`), "scope of t" (points to `double t`), "scope of i" (points to `int i` in the first loop), "scope of another i" (points to `int i` in the second loop), and "two different variables with the same name i" (points to both `i` declarations).



Best practice: declare variables to limit their scope.

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

Summary of what happens when a function is called:

- Control transfers to the function code.
- Argument variables are assigned the values given in the call.
- Function code is executed.
- Return value is assigned in place of the function name in calling code.
- Control transfers back to the calling code.

Note. This is known as "pass by value."

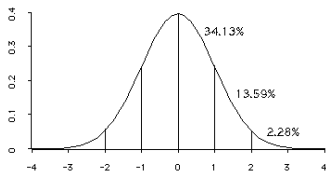
Gaussian Distribution

Gaussian Distribution

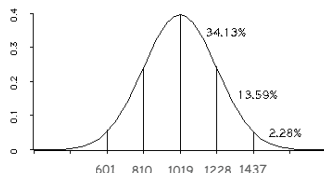
Standard Gaussian distribution.

- "Bell curve."
- Basis of most statistical analysis in social and physical sciences.

Ex. 2000 SAT scores follow a Gaussian distribution with mean $\mu = 1019$, stddev $\sigma = 209$.



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



$$\begin{aligned} \phi(x, \mu, \sigma) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 / 2\sigma^2} \\ &= \phi\left(\frac{x-\mu}{\sigma}\right) / \sigma \end{aligned}$$

Java Function for $\phi(x)$

Mathematical functions. Use built-in functions when possible; build your own when not available.

```
public class Gaussian {
    public static double phi(double x) {
        return Math.exp(-x*x / 2) / Math.sqrt(2 * Math.PI);
    }
    public static double phi(double x, double mu, double sigma) {
        return phi((x - mu) / sigma) / sigma;
    }
}
```

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

$$\phi(x, \mu, \sigma) = \phi\left(\frac{x-\mu}{\sigma}\right) / \sigma$$

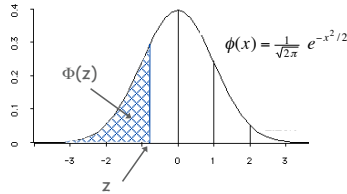
Overloading. Functions with different signatures are different.
Multiple arguments. Functions can take any number of arguments.
Calling other functions. Functions can call other functions.

library or user-defined

Gaussian Cumulative Distribution Function

Goal. Compute Gaussian cdf $\Phi(z)$.

Challenge. No "closed form" expression and not in Java library.



$$\Phi(z) = \int_{-\infty}^z \phi(x) dx \quad \text{Taylor series}$$

$$= \frac{1}{2} + \phi(z) \left(z + \frac{z^3}{3} + \frac{z^5}{3 \cdot 5} + \frac{z^7}{3 \cdot 5 \cdot 7} + \dots \right)$$

Bottom line. 1,000 years of mathematical formulas at your fingertips.

Java function for $\Phi(z)$

```
public class Gaussian {
    public static double phi(double x)
        // as before

    public static double Phi(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 * phi(z);
    }

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

accurate with absolute error less than $8 \cdot 10^{-16}$

$$\Phi(z, \mu, \sigma) = \int_{-\infty}^z \phi(z, \mu, \sigma) = \Phi((z - \mu) / \sigma)$$

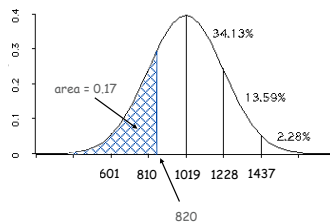
13

14

SAT Scores

Q. NCAA requires at least 820 for Division I athletes. What fraction of test takers in 2000 do not qualify?

A. $\Phi(820, \mu, \sigma) \approx 0.17051$. [approximately 17%]



```
double fraction = Gaussian.Phi(820, 1019, 209);
```

15

Gaussian Distribution

Q. Why relevant in mathematics?

A. Central limit theorem: under very general conditions, average of a set of variables tends to the Gaussian distribution.

Q. Why relevant in the sciences?

A. Models a wide range of natural phenomena and random processes.

- Weights of humans, heights of trees in a forest.
- SAT scores, investment returns.

Caveat.

Everybody believes in the exponential law of errors: the experimenters, because they think it can be proved by mathematics; and the mathematicians, because they believe it has been established by observation. - M. Lippman in a letter to H. Poincaré

16

Building Functions

Functions enable you to build a new layer of abstraction.

- Takes you beyond pre-packaged libraries.
- You build the tools you need: `Gaussian.phi()`, ...

Process.

- Step 1: identify a useful feature.
 - Step 2: implement it.
 - Step 3: use it.
-
- Step 3': re-use it in **any** of your programs.

17

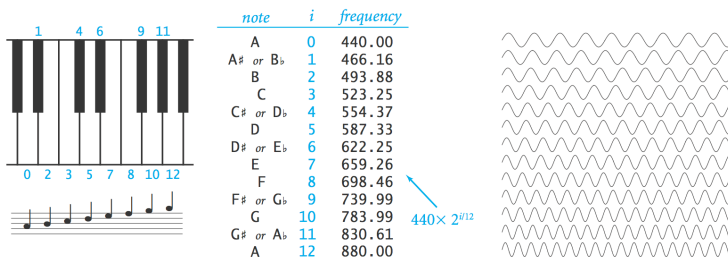
Digital Audio

Crash Course in Sound

Sound. Perception of the **vibration** of molecules in our eardrums.

Concert A. Sine wave, scaled to oscillate at 440Hz.

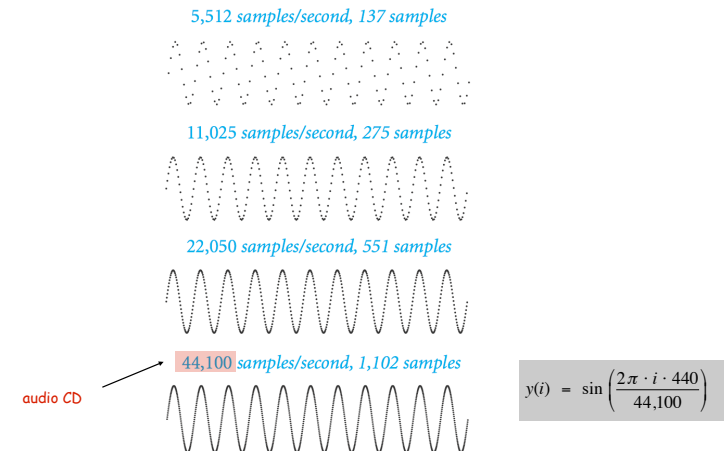
Other notes. 12 notes on chromatic scale, divided logarithmically.



Notes, numbers, and waves

Digital Audio

Sampling. Represent curve by sampling it at regular intervals.



19

20

Musical tone. Create a music tone of a given frequency and duration.

```
public static double[] tone(double hz, double seconds) {
    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);
    }
    return a;
}
```

$$y(i) = \sin\left(\frac{2\pi \cdot i \cdot \text{hz}}{44,100}\right)$$

Remark. Can use arrays as function return value and/or argument.

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
    void double[] read(String file) read from a .wav file
}
```

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

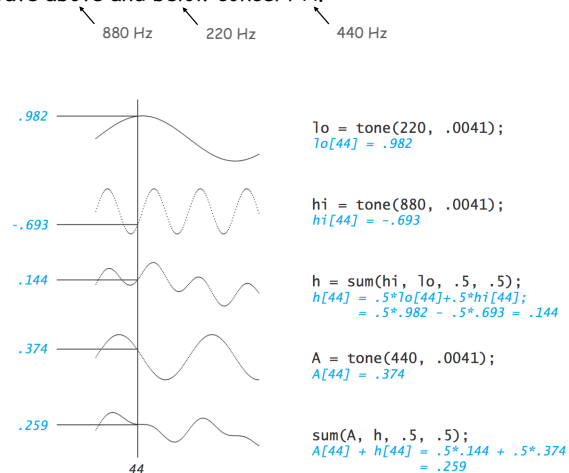
```
double[] a = tone(440, 1.5);
StdAudio.play(a);
```

21

22

Harmonics

Concert A with harmonics. Obtain richer sound by adding tones one octave above and below concert A.



23

Harmonics

```
public class PlayThatTune {
    // return weighted sum of two arrays
    public static double[] sum(double[] a, double[] b, double awt, double bwt) {
        double[] c = new double[a.length];
        for (int i = 0; i < a.length; i++)
            c[i] = a[i]*awt + b[i]*bwt;
        return c;
    }

    // return a note of given pitch and duration
    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[] h = sum(hi, lo, .5, .5);
        return sum(a, h, .5, .5);
    }

    public static double[] tone(double hz, double t)
    // see previous slide

    public static void main(String[] args)
    // see next slide
}
```

24

Harmonics

Play that tune. Read in pitches and durations from standard input, and play using standard audio.

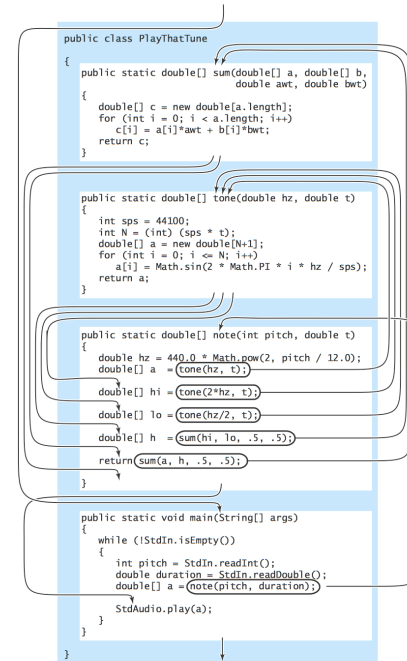
```
public static void main(String[] args) {
    while (!StdIn.isEmpty()) {
        int pitch = StdIn.readInt();
        double duration = StdIn.readDouble();
        double[] a = note(pitch, duration);
        StdAudio.play(a);
    }
}
```

```
% 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
```



25



26