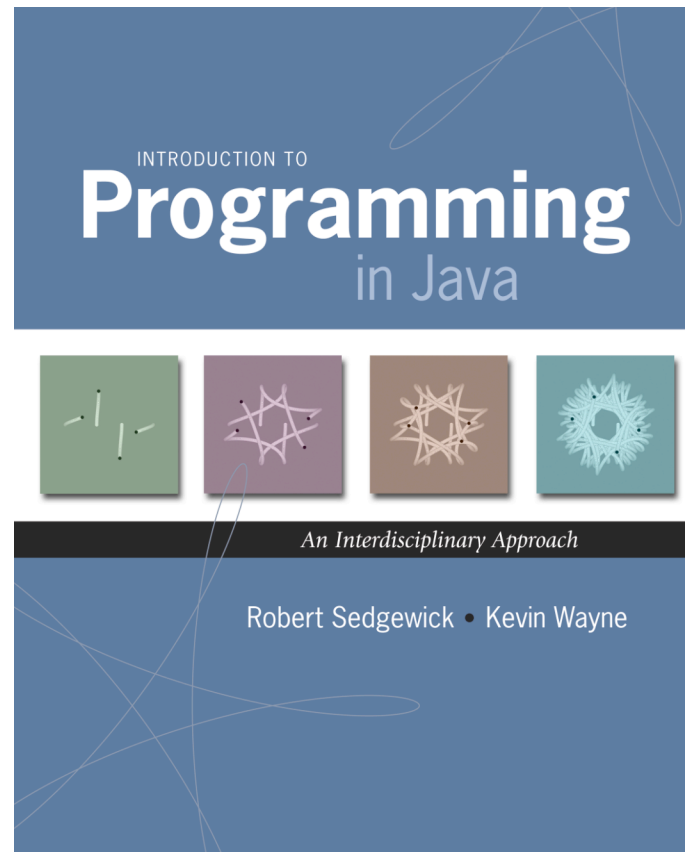
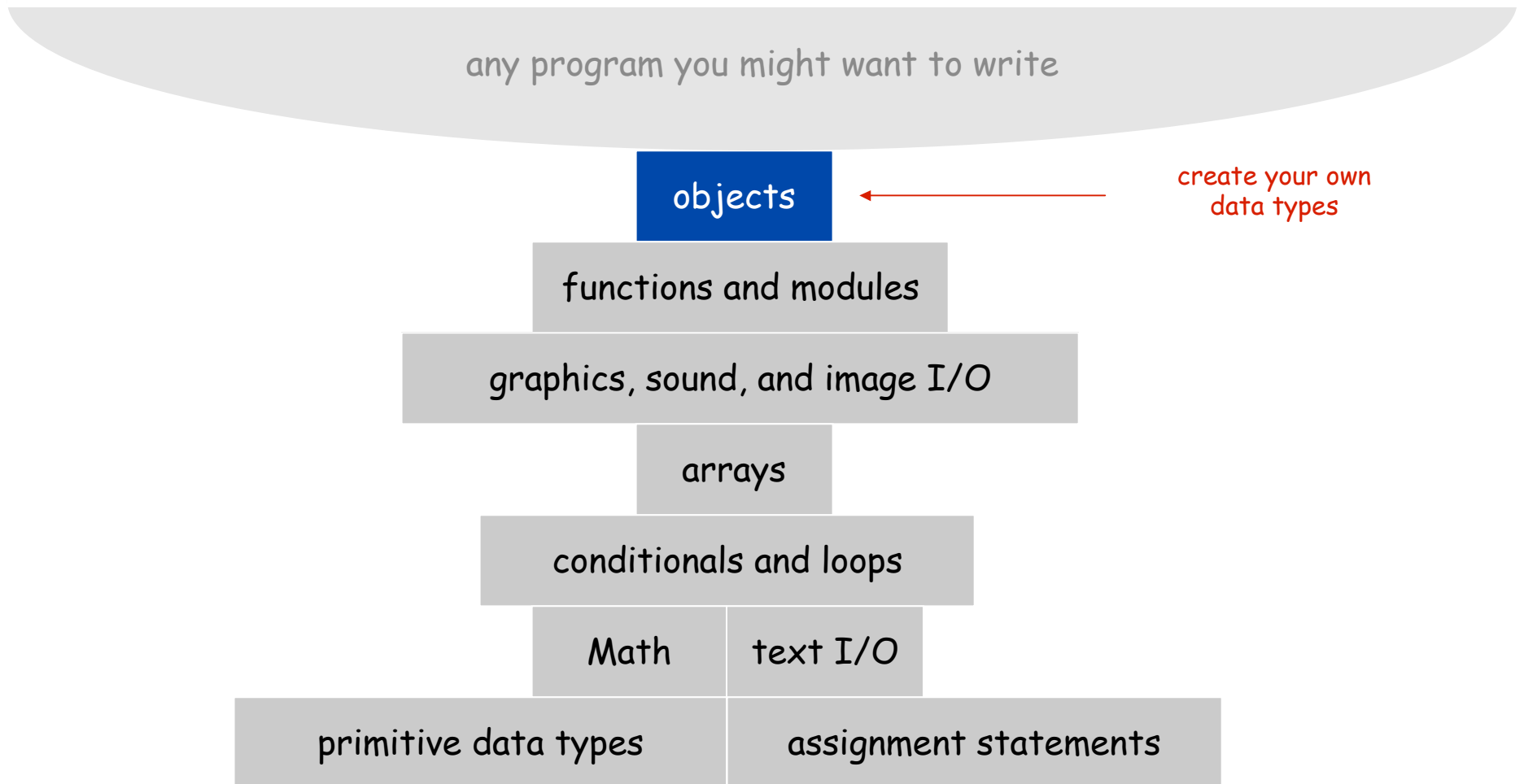


# 3.1 Using Data Types

---



# A Foundation for Programming



# Data Types

**Data type.** Set of values and operations on those values.

**Primitive types.** Ops directly translate to machine instructions.

| Data Type | Set of Values                  | Operations              |
|-----------|--------------------------------|-------------------------|
| boolean   | true, false                    | not, and, or, xor       |
| int       | $-2^{31}$ to $2^{31} - 1$      | add, subtract, multiply |
| double    | any of $2^{64}$ possible reals | add, subtract, multiply |

**We want to write programs that process other types of data.**

- Colors, pictures, strings, input streams, ...
- Complex numbers, vectors, matrices, polynomials, ...
- Points, polygons, charged particles, celestial bodies, ...

# Objects

**Object.** Holds a data type value; variable name refers to object.

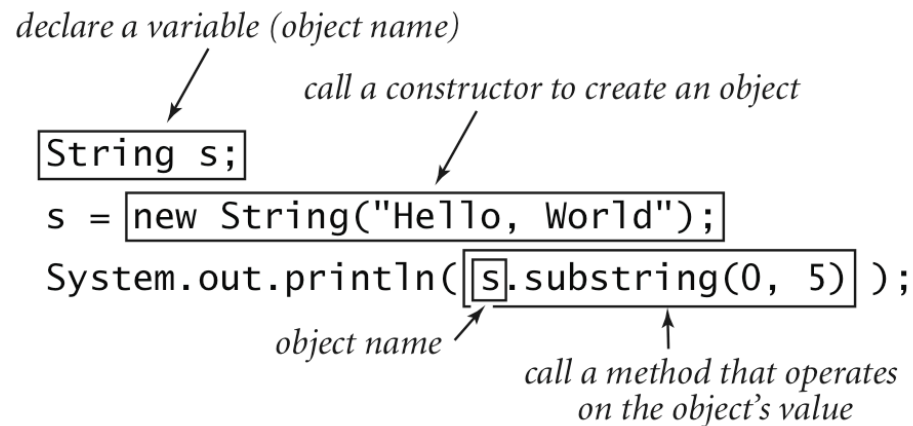
**Impact.** Enables us to create our own data types; define operations on them; and integrate into our programs.

| Data Type | Set of Values          | Operations                    |
|-----------|------------------------|-------------------------------|
| Color     | 24 bits                | get red component, brighten   |
| Picture   | 2D array of colors     | get/set color of pixel (i, j) |
| String    | sequence of characters | length, substring, compare    |

# Constructors and Methods

To construct a new object: Use keyword `new` and name of data type.

To apply an operation: Use name of object, the **dot operator**, and the name of the **method**.



# Image Processing

---

## Color Data Type

**Color.** A sensation in the eye from electromagnetic radiation.

**Set of values.** [RGB representation]  $256^3$  possible values, which quantify the amount of red, green, and blue, each on a scale of 0 to 255.

| R   | G   | B   | Color   |
|-----|-----|-----|---------|
| 255 | 0   | 0   | Red     |
| 0   | 255 | 0   | Green   |
| 0   | 0   | 255 | Blue    |
| 255 | 255 | 255 | White   |
| 0   | 0   | 0   | Black   |
| 255 | 0   | 255 | Magenta |
| 105 | 105 | 105 | Grey    |

# Color Data Type

**Color.** A sensation in the eye from electromagnetic radiation.

**Set of values.** [RGB representation]  $256^3$  possible values, which quantify the amount of red, green, and blue, each on a scale of 0 to 255.

**API.** Application Programming Interface.

```
public class java.awt.Color
```

---

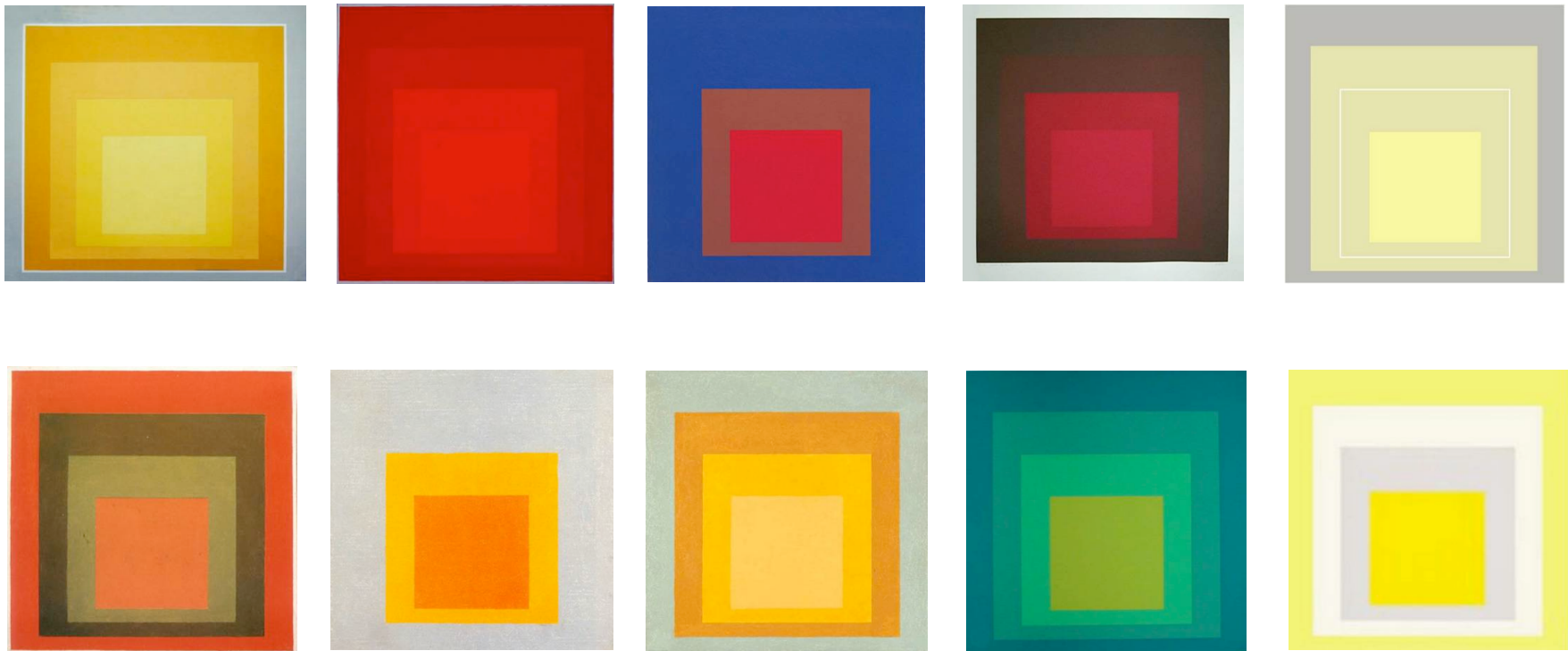
```
    Color(int r, int g, int b)
    int  getRed()           red intensity
    int  getGreen()        green intensity
    int  getBlue()         blue intensity
    Color brighter()       brighter version of this color
    Color darker()         darker version of this color
    String toString()      string representation of this color
```

<http://java.sun.com/j2se/1.5.0/docs/api/java/awt/Color.html>



# Albers Squares

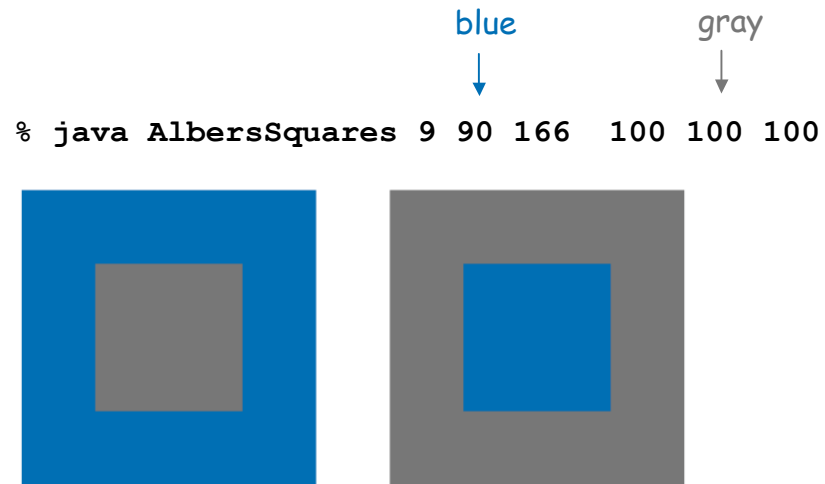
Josef Albers. Revolutionized the way people think about color.



*Homage to the Square* by Josef Albers (1949-1975)

# Albers Squares

Josef Albers. Revolutionized the way people think about color.



# Using Colors in Java

```
import java.awt.Color;
```

to access Color library

```
public class AlbersSquares {  
    public static void main(String[] args) {  
        int r1 = Integer.parseInt(args[0]);  
        int g1 = Integer.parseInt(args[1]);  
        int b1 = Integer.parseInt(args[2]);  
        Color c1 = new Color(r1, g1, b1);  
  
        int r2 = Integer.parseInt(args[3]);  
        int g2 = Integer.parseInt(args[4]);  
        int b2 = Integer.parseInt(args[5]);  
        Color c2 = new Color(r2, g2, b2);  
  
        StdDraw.setPenColor(c1);  
        StdDraw.filledSquare(.25, .5, .2);  
        StdDraw.setPenColor(c2);  
        StdDraw.filledSquare(.25, .5, .1);  
  
        StdDraw.setPenColor(c2);  
        StdDraw.filledSquare(.75, .5, .2);  
        StdDraw.setPenColor(c1);  
        StdDraw.filledSquare(.75, .5, .1);  
    }  
}
```

first color

second color

first square

second square

## Monochrome Luminance

Monochrome luminance. Effective brightness of a color.

NTSC formula.  $Y = 0.299r + 0.587g + 0.114b$ .

```
import java.awt.Color;

public class Luminance {

    public static double lum(Color c) {
        int r = c.getRed();
        int g = c.getGreen();
        int b = c.getBlue();
        return .299*r + .587*g + .114*b;
    }
}
```

## Color Compatibility

Q. Which font colors will be most readable with which background colors on computer monitors and cell phone screens?

A. Rule of thumb: difference in luminance should be  $\geq 128$ .



```
public static boolean compatible(Color a, Color b) {  
    return Math.abs(lum(a) - lum(b)) >= 128.0;  
}
```

# Grayscale

**Grayscale.** When all three R, G, and B values are the same, resulting color is on grayscale from 0 (black) to 255 (white).

**Convert to grayscale.** Use luminance to determine value.

```
public static Color toGray(Color c) {  
    int y = (int) Math.round(lum(c));  
    Color gray = new Color(y, y, y);  
    return gray;  
}
```

| <i>red</i> | <i>green</i> | <i>blue</i> |                          |   |
|------------|--------------|-------------|--------------------------|---|
| 9          | 90           | 166         | <i>this color</i>        |  |
| 74         | 74           | 74          | <i>grayscale version</i> |  |
| 0          | 0            | 0           | <i>black</i>             |  |

$$0.299 * 9 + 0.587 * 90 + 0.114 * 166 = 74.445$$

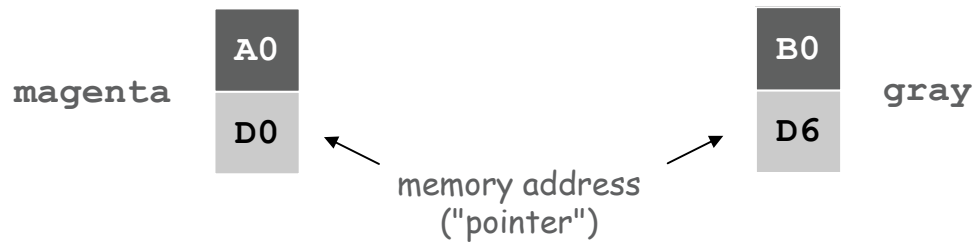
round double to nearest int

**Bottom line.** We are writing programs that manipulate **color**.

# OOP Context for Color

Possible memory representation.

| D0  | D1 | D2  | D3 | D4 | D5 | D6  | D7  | D8  |
|-----|----|-----|----|----|----|-----|-----|-----|
| 255 | 0  | 255 | 0  | 0  | 0  | 105 | 105 | 105 |



Object reference is analogous to variable name.

- We can manipulate the value that it holds.
- We can pass it to (or return it from) a method.

## References

René Magritte. "This is not a pipe."



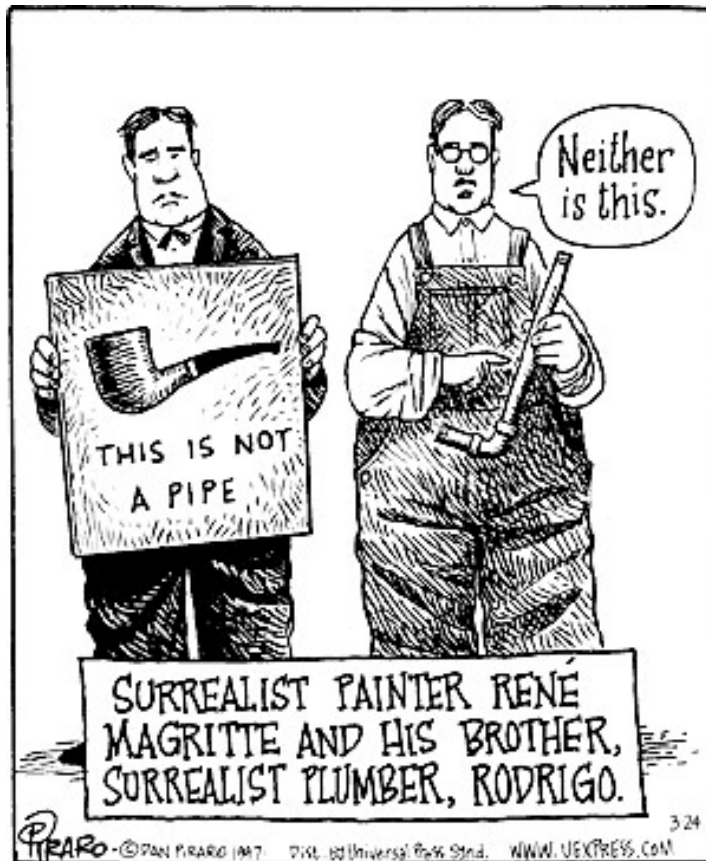
Java. This is not a color.

```
Color sienna = new Color(160, 82, 45);  
Color c = sienna.darker();
```

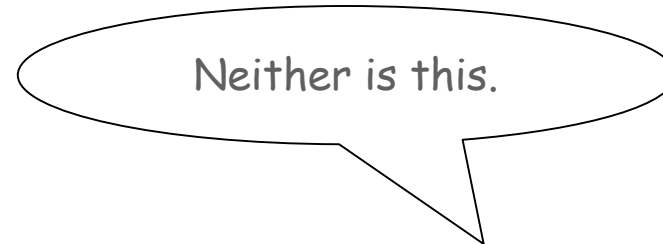
OOP. Natural vehicle for studying abstract models of the real world.



# This is Not a Pipe



Dan Piraro, <http://www.uexpress.com>



```
% java RandomSeq 10000 | java Average
```

# Picture Data Type

Raster graphics. Basis for image processing.

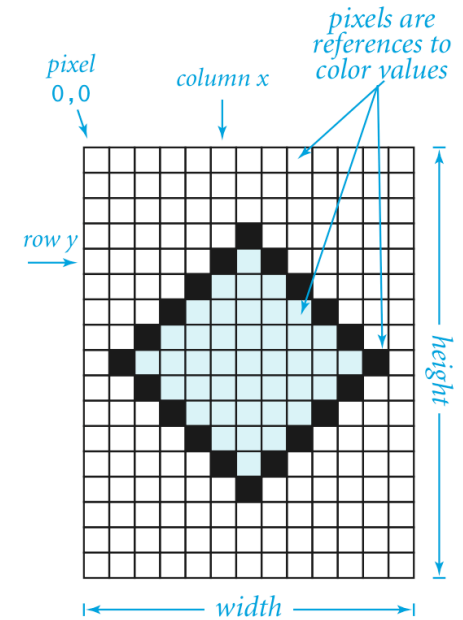
Set of values. 2D array of `Color` objects (pixels).

API.

```
public class Picture
```

---

|  |   |
|--|---|
| <code>Picture(String filename)</code>        | <i>create a picture from a file</i>       |
| <code>Picture(int w, int h)</code>           | <i>create a blank w-by-h picture</i>      |
| <code>int width()</code>                     | <i>return the width of the picture</i>    |
| <code>int height()</code>                    | <i>return the height of the picture</i>   |
| <code>Color get(int x, int y)</code>         | <i>return the color of pixel (x, y)</i>   |
| <code>void set(int x, int y, Color c)</code> | <i>set the color of pixel (x, y) to c</i> |
| <code>void show()</code>                     | <i>display the image in a window</i>      |
| <code>void save(String filename)</code>      | <i>save the image to a file</i>           |



## Image Processing: Grayscale Filter

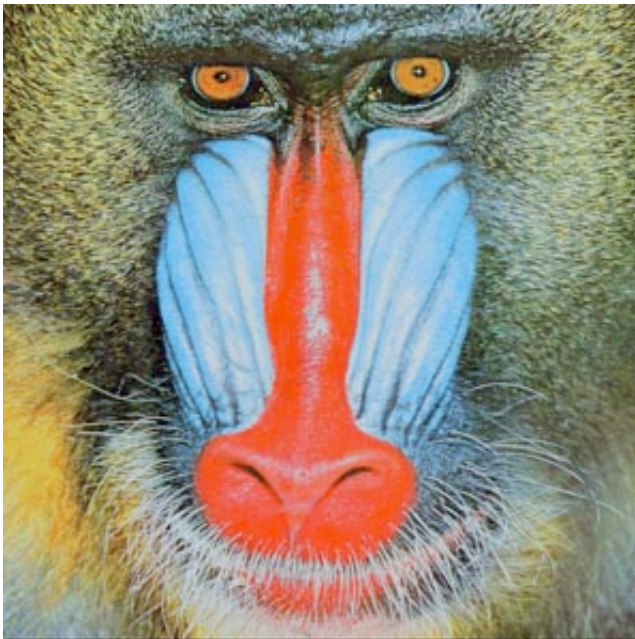
**Goal.** Convert color image to grayscale according to luminance formula.

```
import java.awt.Color;

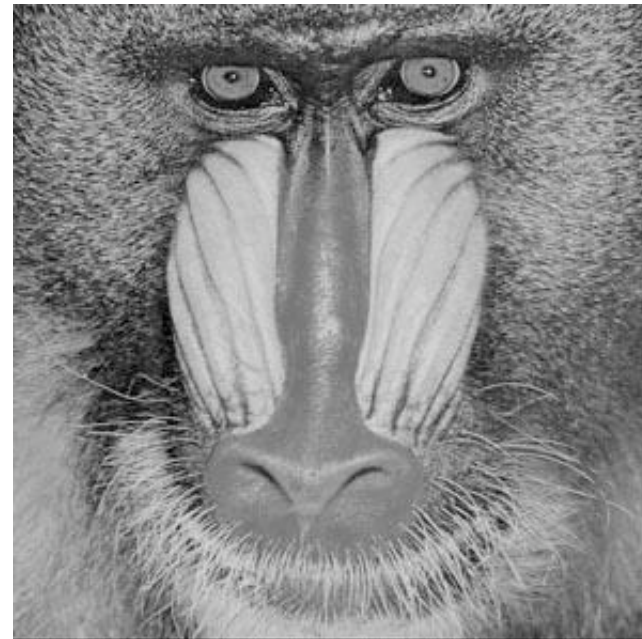
public class Grayscale {
    public static void main(String[] args) {
        Picture pic = new Picture(args[0]);
        for (int x = 0; x < pic.width(); x++) {
            for (int y = 0; y < pic.height(); y++) {
                Color color = pic.get(x, y);
                Color gray = Luminance.toGray(color);
                pic.set(x, y, gray);
            }
        }
        pic.show();
    }
}
```

## Image Processing: Grayscale Filter

**Goal.** Convert color image to grayscale according to luminance formula.



`mandrill.jpg`



`% java Grayscale mandrill.jpg`

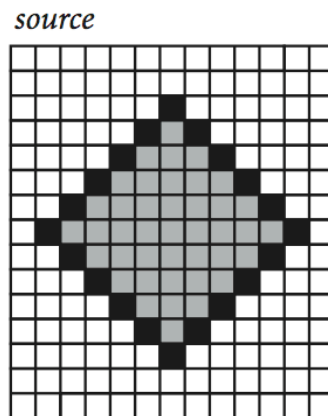
# Image Processing: Scaling Filter

**Goal.** Shrink or enlarge an image to desired size.

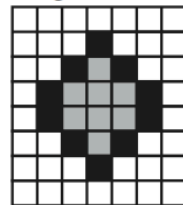
**Downscaling.** To shrink, delete half the rows and columns.

**Upscaling.** To enlarge, replace each pixel by 4 copies.

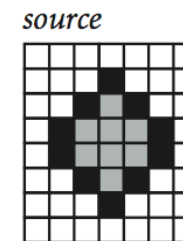
*downscaling*



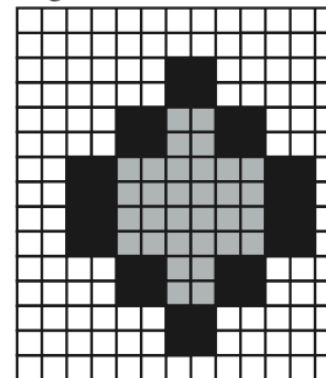
*target*



*upsampling*



*target*

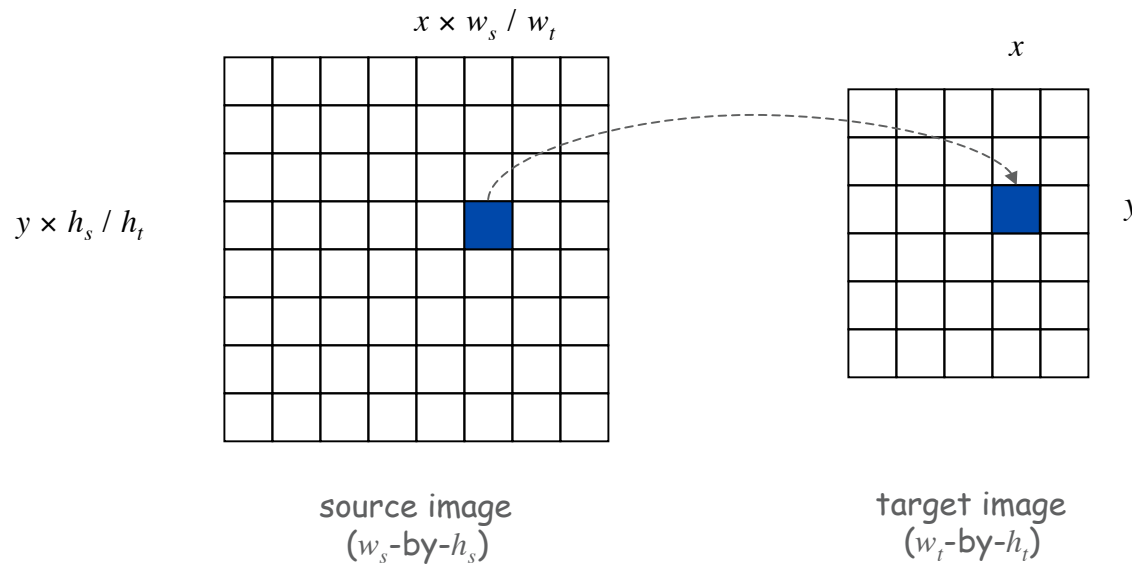


# Image Processing: Scaling Filter

**Goal.** Shrink or enlarge an image to desired size.

**Uniform strategy.** To convert from  $w_s$ -by- $h_s$  to  $w_t$ -by- $h_t$ :

- Scale column index by  $w_s / w_t$ .
- Scale row index by  $h_s / h_t$ .
- Set color of pixel  $(x, y)$  in target image to color of pixel  $(x \times w_s / w_t, y \times h_s / h_t)$  in source image.



## Image Processing: Scaling Filter

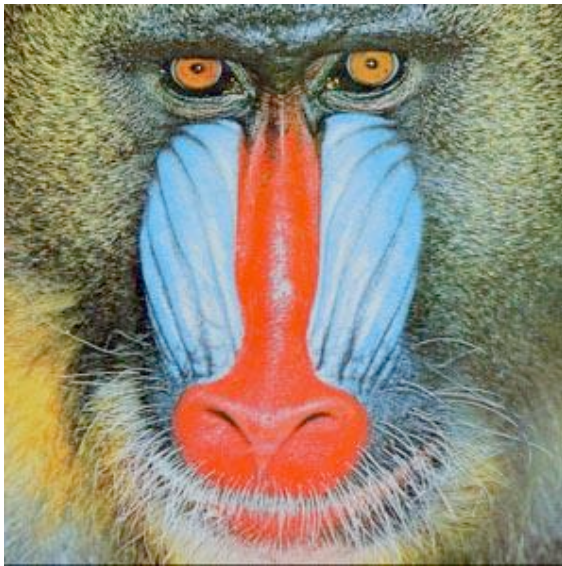
```
import java.awt.Color;

public class Scale {
    public static void main(String[] args) {
        String filename = args[0];
        int w = Integer.parseInt(args[1]);
        int h = Integer.parseInt(args[2]);
        Picture source = new Picture(filename);
        Picture target = new Picture(w, h);
        for (int tx = 0; tx < w; tx++) {
            for (int ty = 0; ty < h; ty++) {
                int sx = tx * source.width() / w;
                int sy = ty * source.height() / h;
                Color color = source.get(sx, sy);
                target.set(tx, ty, color);
            }
        }
        source.show();
        target.show();
    }
}
```



## Image Processing: Scaling Filter

Scaling filter. Creates two `Picture` objects and two windows.



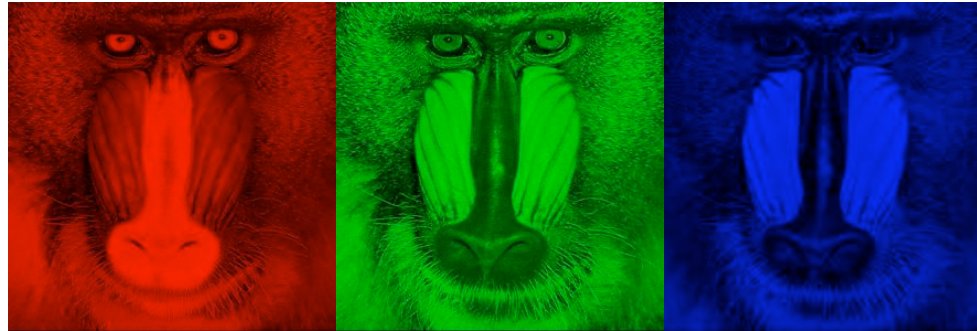
`mandrill.jpg`



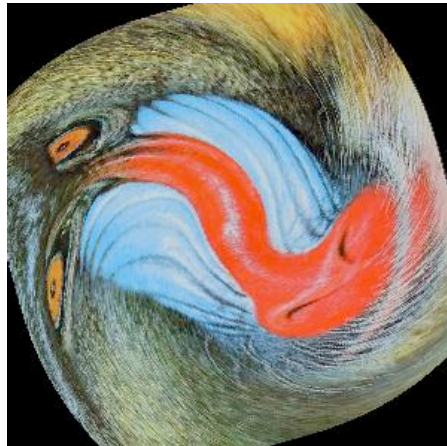
```
% java Scale mandrill.jpg 400 200
```



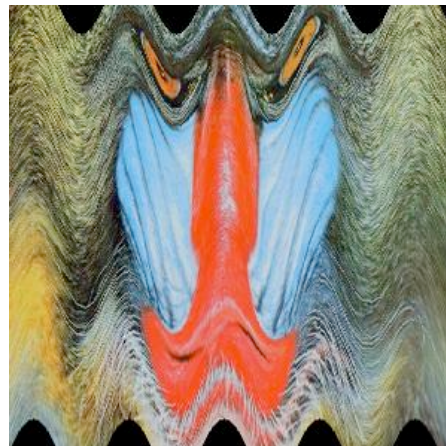
# More Image Processing Effects



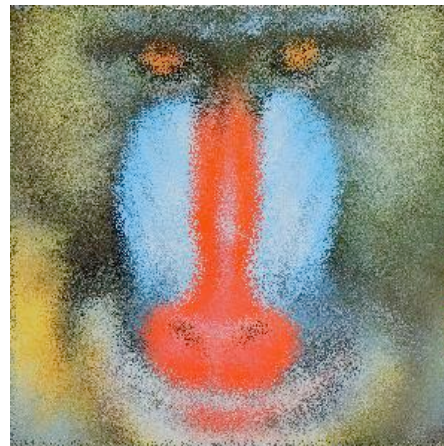
RGB color separation



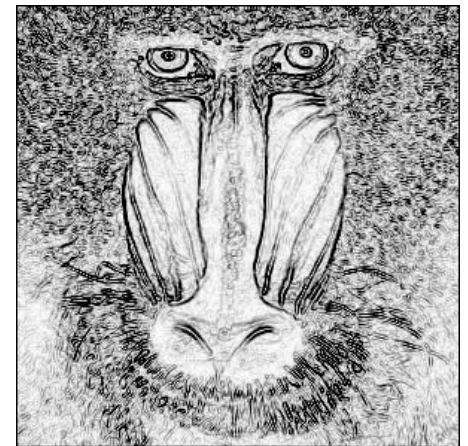
swirl filter



wave filter



glass filter



Sobel edge detection

# Text Processing

---

# String Data Type

**String data type.** Basis for text processing.  
**Set of values.** Sequence of Unicode characters.

## API.

public class String (Java string data type)

---

|                                       |   |
|---------------------------------------|---|
| String(String s)                      | <i>create a string with the same value as s</i>             |
| int length()                          | <i>string length</i>  |
| char charAt(int i)                    | <i>i<sup>th</sup> character</i>                             |
| String substring(int i, int j)        | <i>i<sup>th</sup> through (j-1)<sup>st</sup> characters</i> |
| boolean contains(String sub)          | <i>does string contain sub as a substring?</i>              |
| boolean startsWith(String pre)        | <i>does string start with pre?</i>                          |
| boolean endsWith(String post)         | <i>does string end with post?</i>                           |
| int indexOf(String p)                 | <i>index of first occurrence of p</i>                       |
| int indexOf(String p, int i)          | <i>index of first occurrence of p after i</i>               |
| String concat(String t)               | <i>this string with t appended</i>                          |
| int compareTo(String t)               | <i>string comparison</i>                                    |
| String replaceAll(String a, String b) | <i>result of changing as to bs</i>                          |
| String[] split(String delim)          | <i>strings between occurrences of delim</i>                 |
| boolean equals(String t)              | <i>is this string's value the same as t's?</i>              |
| ...                                   |   |

<http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html>

## Typical String Processing Code

|   |   |
|---|---|
| <p><i>is the string a palindrome?</i></p>   | <pre>public static boolean isPalindrome(String s) {     int N = s.length();     for (int i = 0; i &lt; N/2; i++)         if (s.charAt(i) != s.charAt(N-1-i))             return false;     return true; }</pre> |
| <p><i>extract file name and extension from a command-line argument</i></p>                              | <pre>String s = args[0]; int dot = s.indexOf("."); String base = s.substring(0, dot); String extension = s.substring(dot + 1, s.length());</pre>  |
| <p><i>print all lines in standard input that contain a string specified on the command line</i></p>     | <pre>String query = args[0]; while (!StdIn.isEmpty()) {     String s = StdIn.readLine();     if (s.contains(query)) StdOut.println(s); }</pre>  |
| <p><i>print all the hyperlinks (to educational institutions) in the text file on standard input</i></p> | <pre>while (!StdIn.isEmpty()) {     String s = StdIn.readString();     if (s.startsWith("http://") &amp;&amp; s.endsWith(".edu"))         StdOut.println(s); }</pre>  |

# Gene Finding

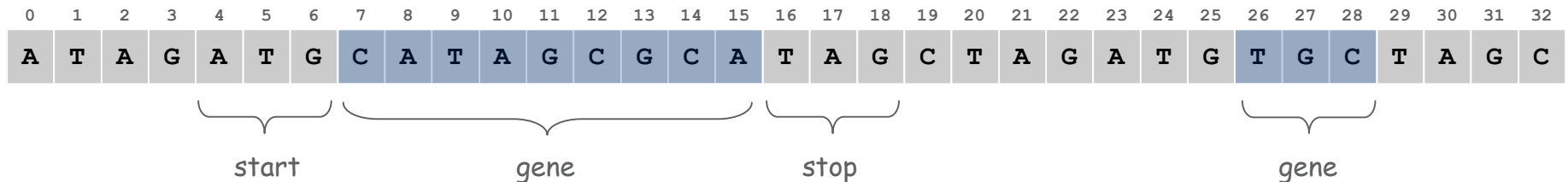
Pre-genomics era. Sequence a human genome.

Post-genomics era. Analyze the data and understand structure.

Genomics. Represent genome as a string over { A, C, T, G } alphabet.

Gene. A substring of genome that represents a functional unit.

- Preceded by ATG. [start codon]
- Multiple of 3 nucleotides. [codons other than start/stop]
- Succeeded by TAG, TAA, or TGA. [stop codons]



## Gene Finding: Algorithm

**Algorithm.** Scan left-to-right through genome.

- If start codon, then set `beg` to index `i`.
- If stop codon and substring is a multiple of 3
  - output gene
  - reset `beg` to -1

|         | <code>i</code> | codon | <code>beg</code> | <i>output</i>                                      | <i>remaining portion of input string</i> |
|---------|----------------|-------|------------------|--|--|
|         | 0              |       | -1               |  | ATAGATGCATAGCGCATAGCTAGATGTGCTAGC        |
|         | 1              | TAG   | -1               |  | TAGATGCATAGCGCATAGCTAGATGTGCTAGC         |
|         | 4              | ATG   | 4                |  | ATGCATAGCGCATAGCTAGATGTGCTAGC            |
| start → | 9              | TAG   | 4                | $\underbrace{\hspace{2em}}_{\text{multiple of 3}}$ | TAGCGCATAGCTAGATGTGCTAGC                 |
|         | 16             | TAG   | 4                | CATAGCGCA  | TAGCTAGATGTGCTAGC                        |
| stop →  | 20             | TAG   | -1               |  | TAGATGTGCTAGC                            |
|         | 23             | ATG   | 23               |  | ATGTGCTAGC                               |
|         | 29             | TAG   | 23               | TGC  | TAGC                                     |

## Gene Finding: Implementation

```
public class GeneFind {
    public static void main(String[] args) {
        String start = args[0];
        String stop = args[1];
        String genome = StdIn.readAll();

        int beg = -1;
        for (int i = 0; i < genome.length() - 2; i++) {
            String codon = genome.substring(i, i+3);
            if (codon.equals(start)) beg = i;
            if (codon.equals(stop) && beg != -1) {
                String gene = genome.substring(beg+3, i);
                if (gene.length() % 3 == 0) {
                    StdOut.println(gene);
                    beg = -1;
                }
            }
        }
    }
}
```

```
% more genomeTiny.txt
```

```
ATAGATGCATAGCGCATAGCTAGATGTGCTAGC
```

```
% java GeneFind ATG TAG < genomeTiny.txt
```

```
CATAGCGCA
```

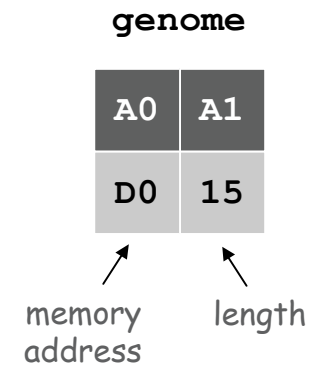
```
TGC
```

# OOP Context for Strings

Possible memory representation of a string.

- `genome = "aacaagtttacaagc";`

| D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| a  | a  | c  | a  | a  | g  | t  | t  | t  | a  | c  | a  | a  | g  | c  |



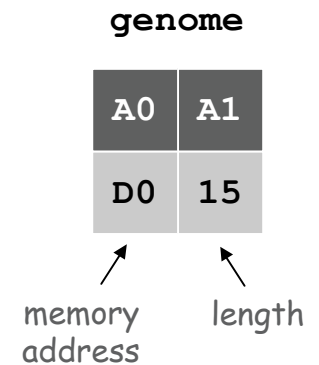


# OOP Context for Strings

## Possible memory representation of a string.

- `genome = "aacaagtttacaagc";`

| D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| a  | a  | c  | a  | a  | g  | t  | t  | t  | a  | c  | a  | a  | g  | c  |



- `s = genome.substring(1, 5);`
- `t = genome.substring(9, 13);`

| s  |    | t  |    |
|----|----|----|----|
| B0 | B1 | B2 | B3 |
| D1 | 4  | D9 | 4  |

*s and t are different strings that share the same value "aaca"*

- `(s == t)` is false, but `(s.equals(t))` is true.

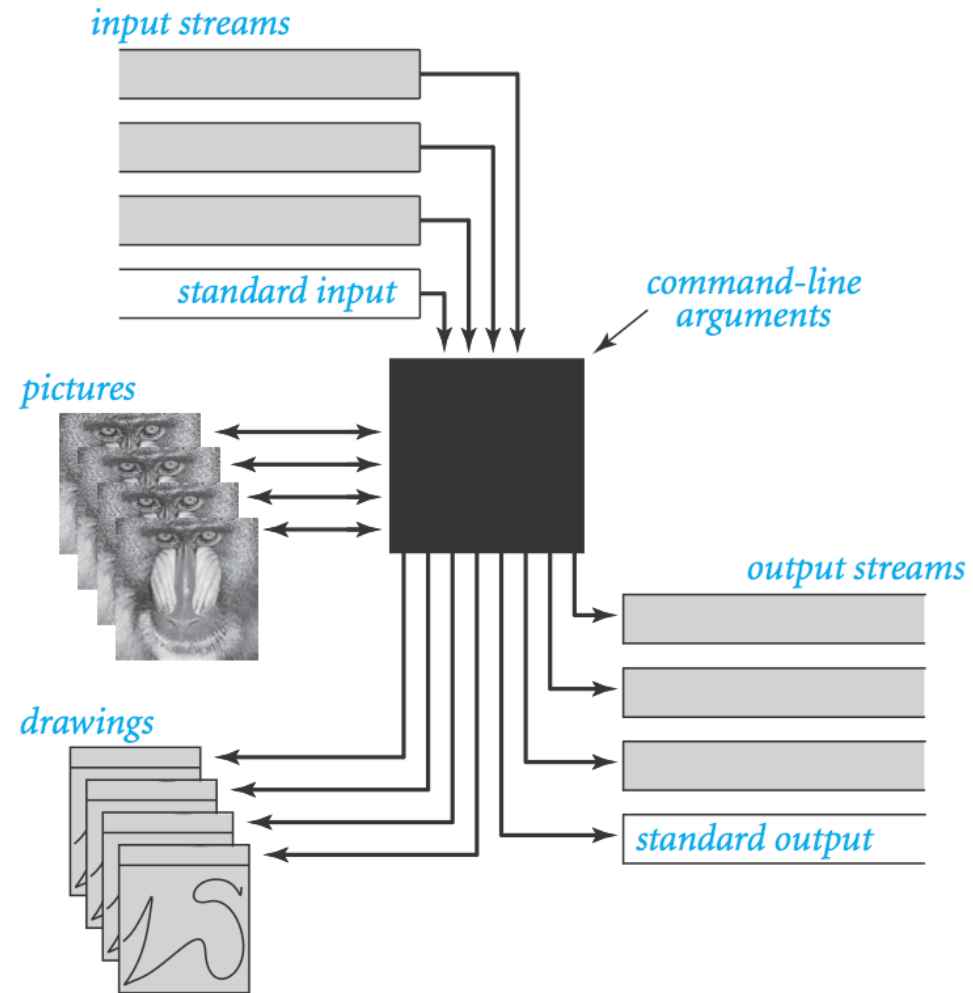
*compares pointers*

*compares character sequences*

# In and Out

---

# Bird's Eye View (Revisited)



# Non-Standard Input

← or use OS to redirect from one file

**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?

```
public class Diff {
    public static void main(String[] args) {
        In in0 = new In(args[0]);
        In in1 = new In(args[1]);
        String s = in0.readAll();
        String t = in1.readAll();
        StdOut.println(s.equals(t));
    }
}
```

# Screen Scraping

Goal. Find current stock price of Google.

```
...
<tr>
<td class="yfnc_tablehead1" width="48%">
Last Trade:
</td>
<td class="yfnc_tabledata1">
<big>
<b>459.52</b>
</big>
</td>
</tr>
<tr>
<td class="yfnc_tablehead1" width="48%">
Trade Time:
</td>
<td class="yfnc_tabledata1">
11:45AM ET
</td>
</tr>
...
```

<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`.
- Read raw html from `http://finance.yahoo.com/q?s=goog`.
- Find first string delimited by `<b>` and `</b>` after Last Trade.

```
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
459.52
```

# Day Trader

## Add bells and whistles.

- 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.



*The New Yorker, September 6, 1999*

## OOP Summary

**Object.** Holds a data type value; variable name refers to object.

**In Java, programs manipulate references to objects.**

- Exception: primitive types, e.g., `boolean`, `int`, `double`.
- Reference types: `String`, `Picture`, `Color`, arrays, everything else.
- OOP purist: language should not have separate primitive types.

**Bottom line.** We wrote programs that manipulate colors, pictures, and strings.

**Next time.** We'll write programs that manipulate **our** own abstractions.