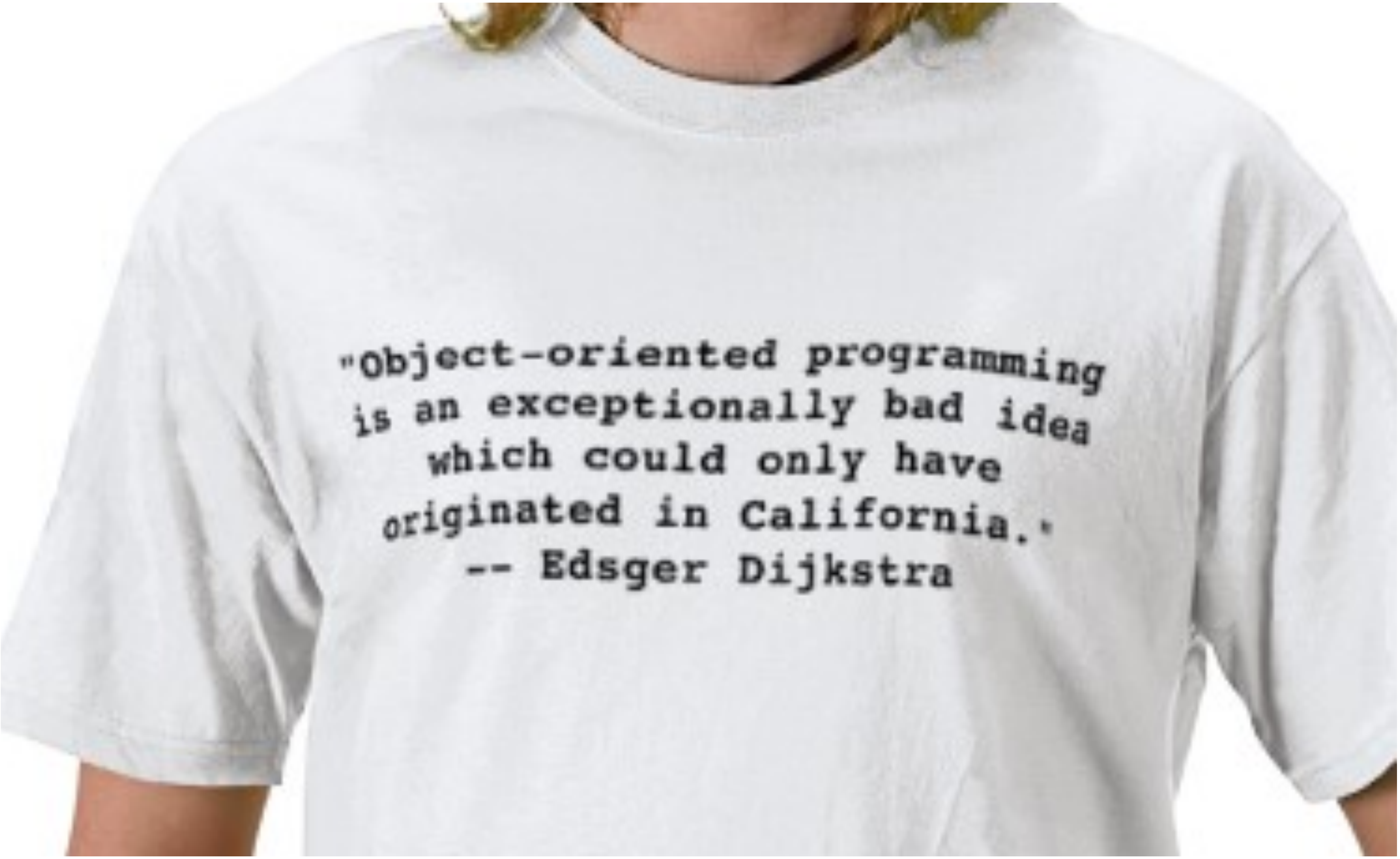
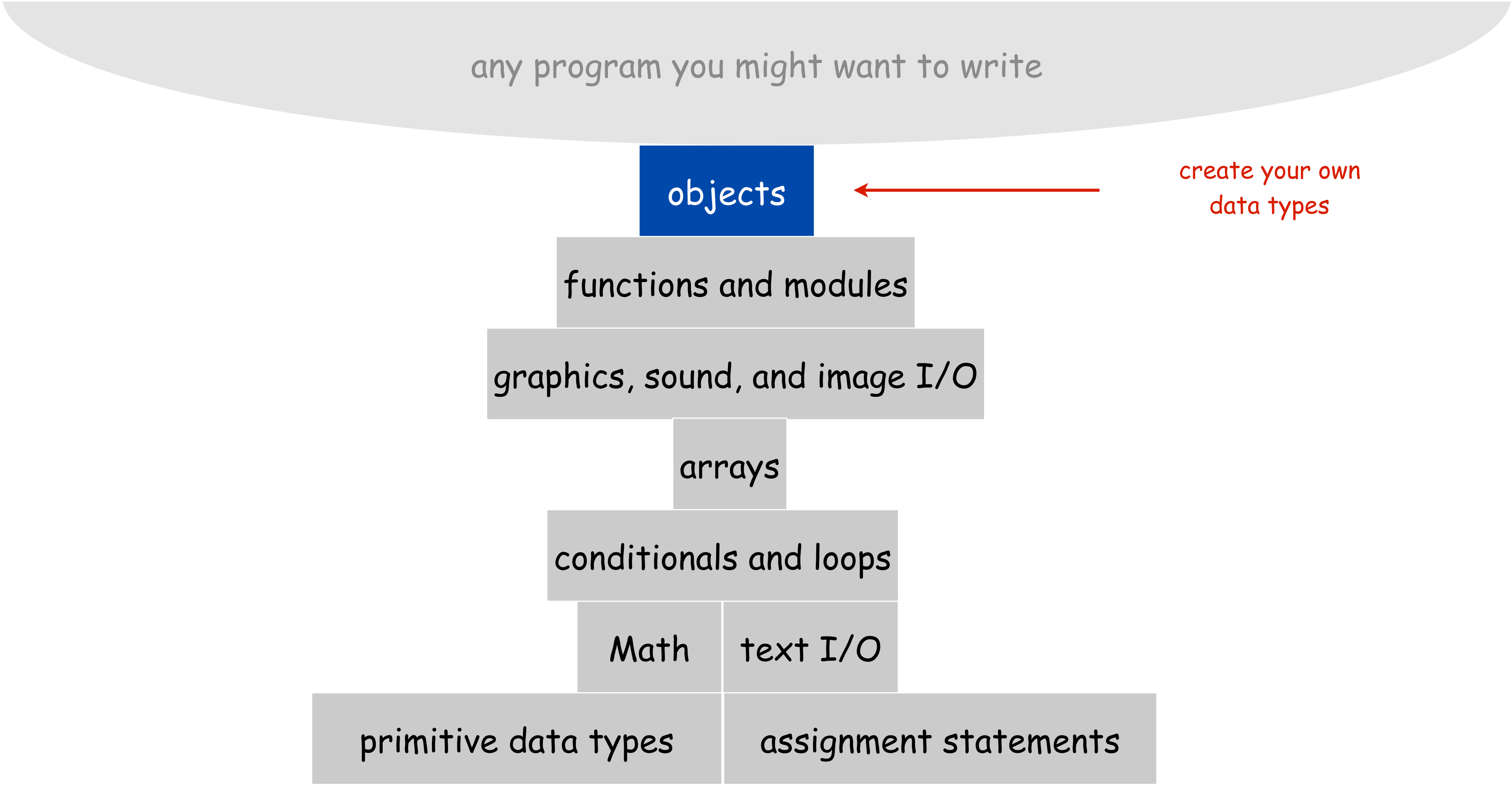

"Object-oriented programming
is an exceptionally bad idea
which could only have
originated in California."
-- Edsger Dijkstra

# 3.1 Data Types



any program you might want to write

objects

create your own data types

functions and modules

graphics, sound, and image I/O

arrays

conditionals and loops

Math

text I/O

primitive data types

assignment statements

# Abstract Data Types

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

Abstract data type. Data type whose representation is hidden from the user.

Primitive types.

- values directly map to machine representations
- operations 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.

Object-oriented programming.

- Create your own data types (sets of values and ops on them)
- Use them in your programs (manipulate objects that hold values).

| 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 |

Abstract data type (ADT). Object representation is hidden.

Impact. We can use ADTs without knowing implementation details.

- this lecture: how to write client programs for several useful ADTs
- next lecture: how to implement your own ADTs

# Constructors and Methods

To use a data type, you need to know how to:

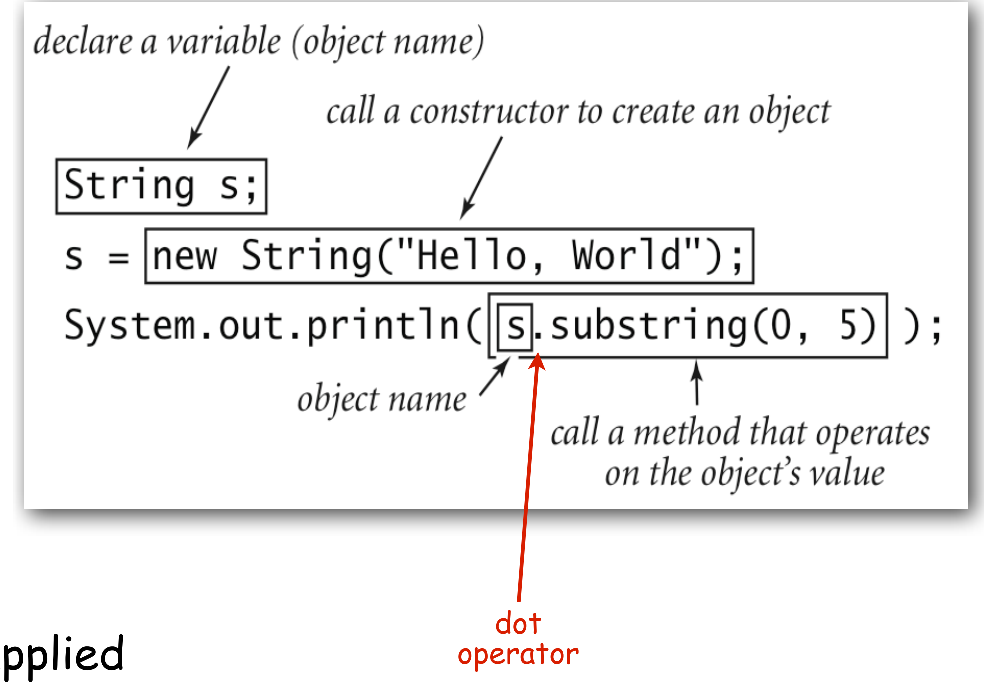
- Construct new objects.
- Apply operations to a given object.

To construct a new object:

- Use keyword **`new`** to invoke a "constructor."
- Use name of data type to specify which type of object.

To apply an operation:

- Use name of object to specify which object
- Use the dot operator to indicate an operation is to be applied
- Use a method name to specify which operation



```
String s;
s = new String("Hello, World");
System.out.println( s.substring(0, 5) );
```

declare a variable (object name)
call a constructor to create an object
object name
call a method that operates on the object's value
dot operator

# 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.

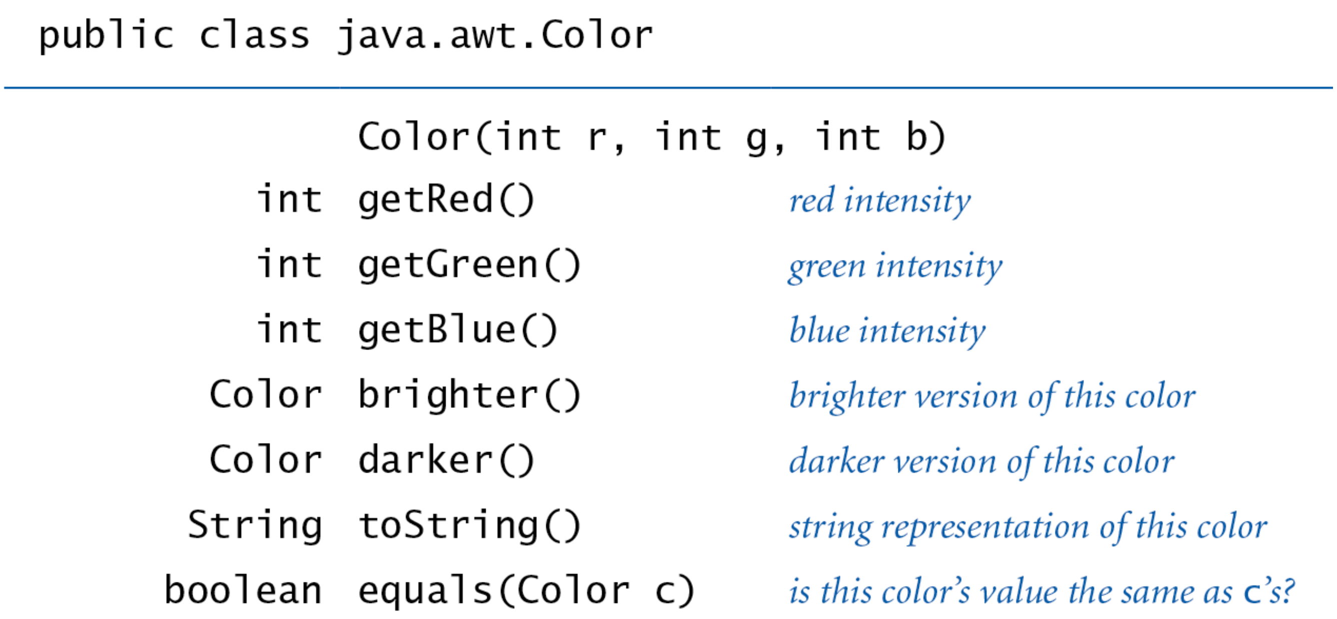
| R | G | B | Color |
|---|---|---|---|
| 255 | 0 | 0 | 🟥 |
| 0 | 255 | 0 | 🟩 |
| 0 | 0 | 255 | 🟦 |
| 255 | 255 | 255 | ⬜ |
| 0 | 0 | 0 | ⬛ |
| 255 | 0 | 255 | 🟪 |
| 105 | 105 | 105 | ▨ |

# 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) specifies set of operations.



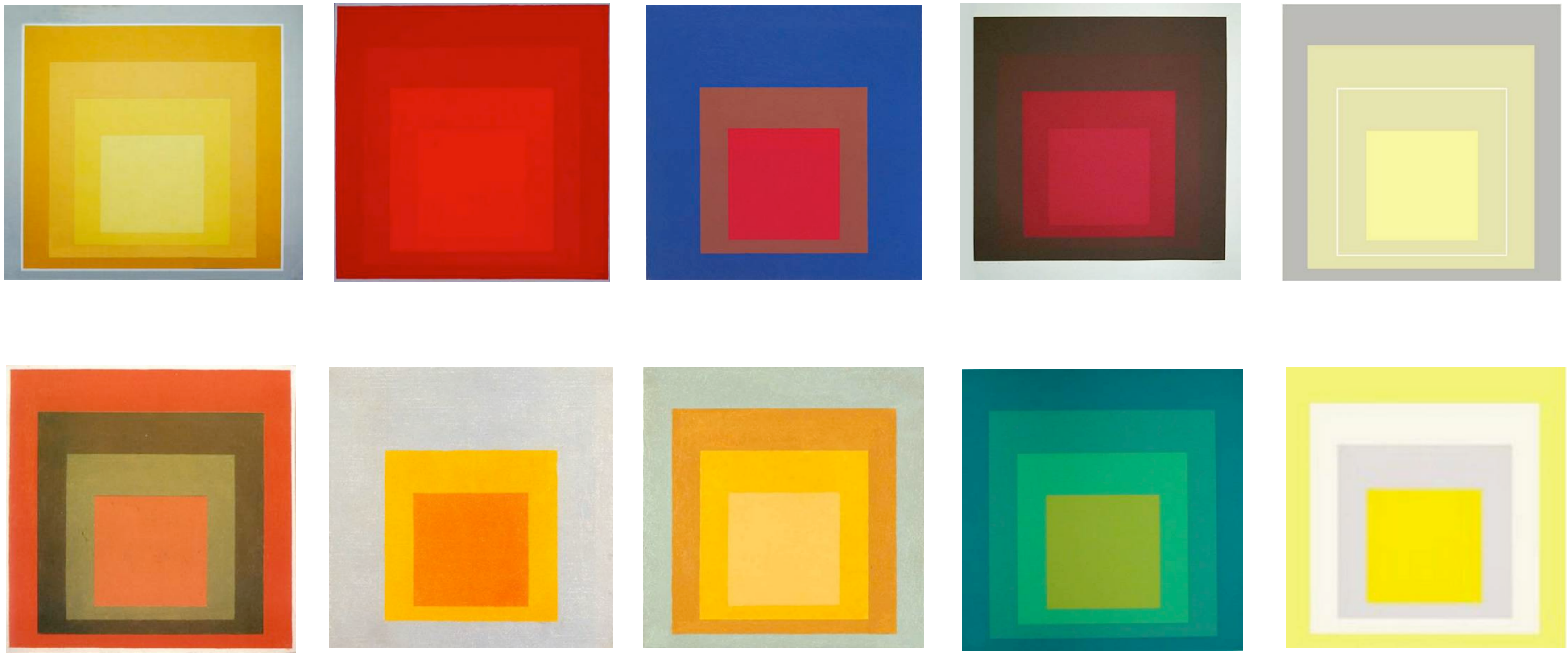
public class java.awt.Color

| | | |
|---|---|---|
| | `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* |
| `boolean` | `equals(Color c)` | *is this color's value the same as* `c`*'s?* |

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.



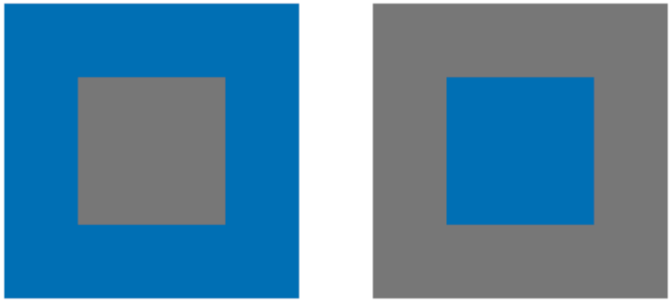
Homage to the Square by Josef Albers (1949-1975)

# Albers Squares

Josef Albers. Revolutionized the way people think about color.

blue ↓ gray ↓

```
% java AlbersSquares 9 90 166 100 100 100
```



# Example Client Program for Color ADT

```
import java.awt.Color;                                       to access Color library

public class AlbersSquares
{
   public static void main(String[] args)
   {
      int r1 = Integer.parseInt(args[0]);                    first color
      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]);                    second color
      int g2 = Integer.parseInt(args[4]);
      int b2 = Integer.parseInt(args[5]);
      Color c2 = new Color(r2, g2, b2);

      StdDraw.setPenColor(c1);                               first square
      StdDraw.filledSquare(.25, .5, .2);
      StdDraw.setPenColor(c2);
      StdDraw.filledSquare(.25, .5, .1);

      StdDraw.setPenColor(c2);                               second square
      StdDraw.filledSquare(.75, .5, .2);
      StdDraw.setPenColor(c1);
      StdDraw.filledSquare(.75, .5, .1);
   }
}
```

# Monochrome Luminance

Monochrome luminance. Effective brightness of a color.

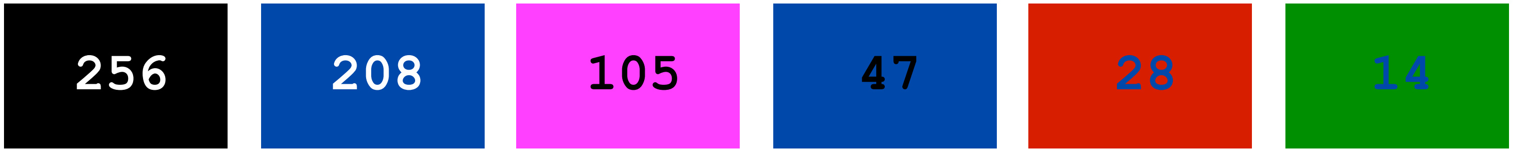
NTSC formula. $Y = 0.299r + 0.587g + 0.114b.$

```
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 ≥ 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;
}
```

round `double` to nearest `int`

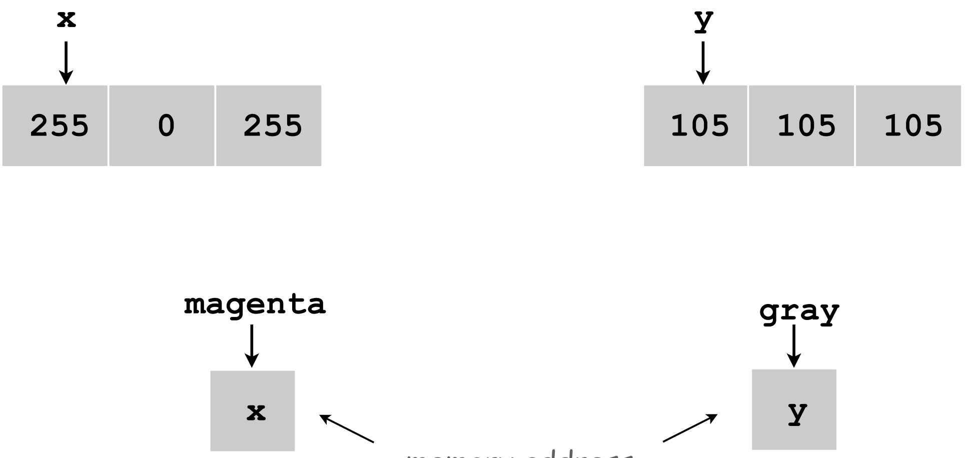
| *red* | *green* | *blue* | | |
|---|---|---|---|---|
| 9 | 90 | 166 | *this color* | 🟦 |
| 74 | 74 | 74 | *grayscale version* | ◼ |
| 0 | 0 | 0 | *black* | ⬛ |

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

Bottom line. We are writing programs that manipulate color.

# OOP Context for Color

Possible memory representation (stay tuned for details).



memory address ("pointer")

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.

Similar to references to arrays.

# References

René Magritte. "This is not a pipe."

Ceci n'est pas une 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

Neither is this.
THIS IS NOT A PIPE
SURREALIST PAINTER RENÉ MAGRITTE AND HIS BROTHER, SURREALIST PLUMBER, RODRIGO.
3.24
PIRARO - ©DAN PIRARO 1997 DIST. BY UNIVERSAL PRESS SYND. WWW.UEXPRESS.COM

Dan Piraro, `http://www.uexpress.com`

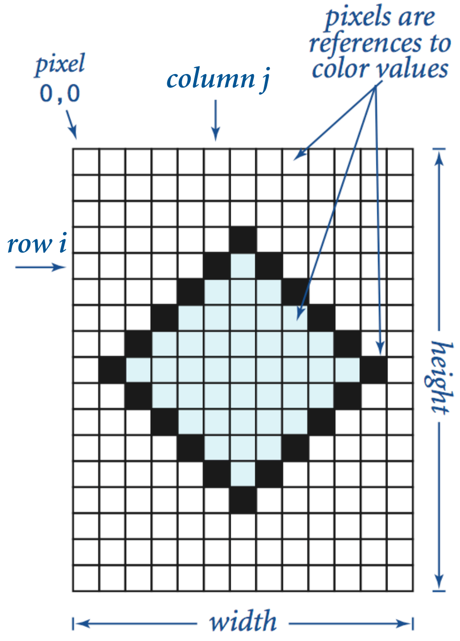
Neither is this.

```
% 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`

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

# Image Processing:  Grayscale Filter

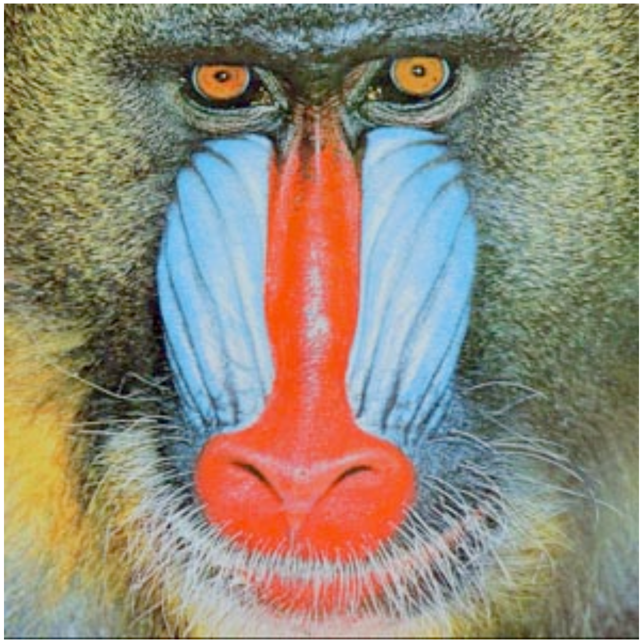
Goal. Convert color image to grayscale according to luminance formula.

```
import java.awt.Color;

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

      pic.show();
   }
}
```

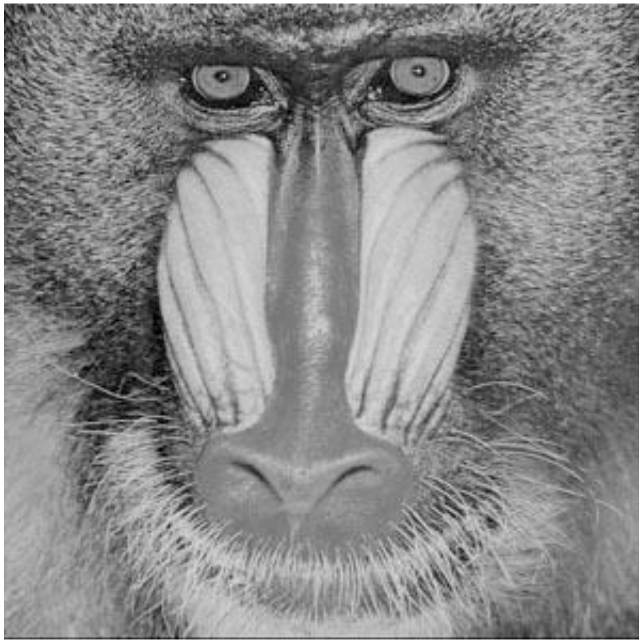
set each pixel to gray

# Image Processing: Grayscale Filter

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



`mandrill.jpg`



```
% java Grayscale mandrill.jpg
```

# TEQ on Image Processing 1

What does the following code do? (Easy question!)

```
Picture pic = new Picture(args[0]);
for (int i = 0; i < pic.width(); i++)
   for (int j = 0; j < pic.height(); j++)
      pic.set(i, j, pic.get(i, j)); pic.show();
```

# TEQ on Image Processing 1

What does the following code do? (Easy question!)

```
Picture pic = new Picture(args[0]);
for (int i = 0; i < pic.width(); i++)
   for (int j = 0; j < pic.height(); j++)
      pic.set(i, j, pic.get(i, j));
pic.show();
```

A. Nothing, then shows the picture.

# TEQ on Image Processing 2

What does the following code do? (Hard question.)

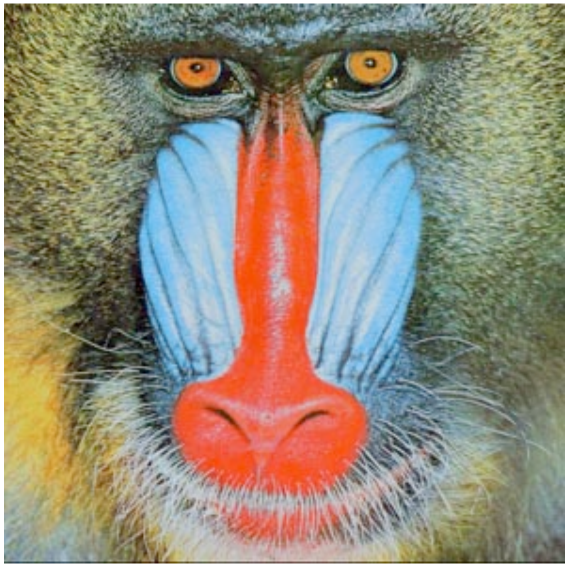
```
Picture pic = new Picture(args[0]);
for (int i = 0; i < pic.width(); i++)
   for (int j = 0; j < pic.height(); j++)
      pic.set(i, pic.height()-j-1, pic.get(i, j));
pic.show();
```

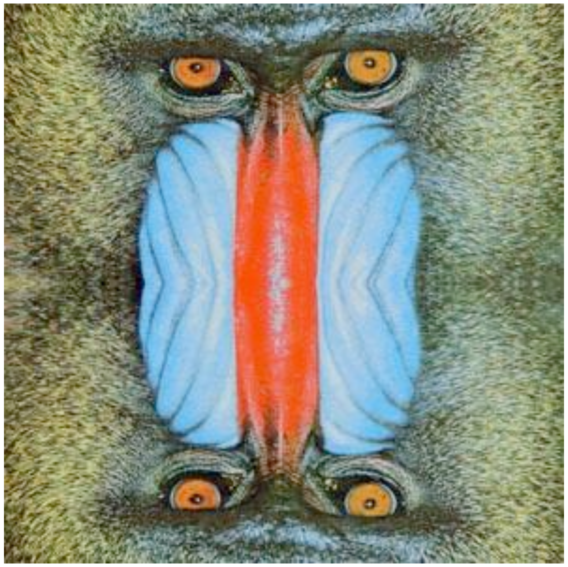
# TEQ on Image Processing 2

What does the following code do? (Hard question.)

```
Picture pic = new Picture(args[0]);
for (int i = 0; i < pic.width(); i++)
   for (int j = 0; j < pic.height(); j++)
      pic.set(i, pic.height()-j-1, pic.get(i, j));
pic.show();
```

A. Tries to turn image upside down, but fails. An instructive bug!





# TEQ on Image Processing 3

What does the following code do?

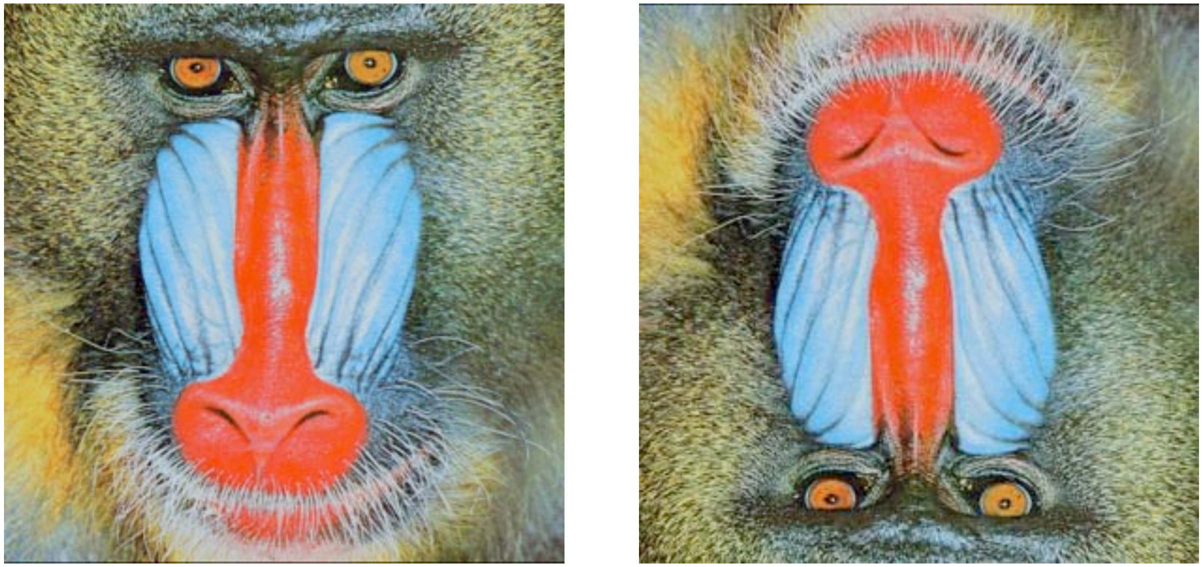
```
Picture source = new Picture(args[0]);
int width  = source.width();
int height = source.height();
Picture target = new Picture(width, height);
for (int i = 0; i < width; i++)
   for (int j = 0; j < height; j++)
      target.set(i, height-j-1, source.get(i, j));
target.show();
```

# TEQ on Image Processing 3

What does the following code do? (Hard question.)

```
Picture source = new Picture(args[0]);
int width  = source.width();
int height = source.height();
Picture target = new Picture(width, height);
for (int i = 0; i < width; i++)
   for (int j = 0; j < height; j++)
      target.set(i, height-j-1, source.get(i, j));
target.show();
```

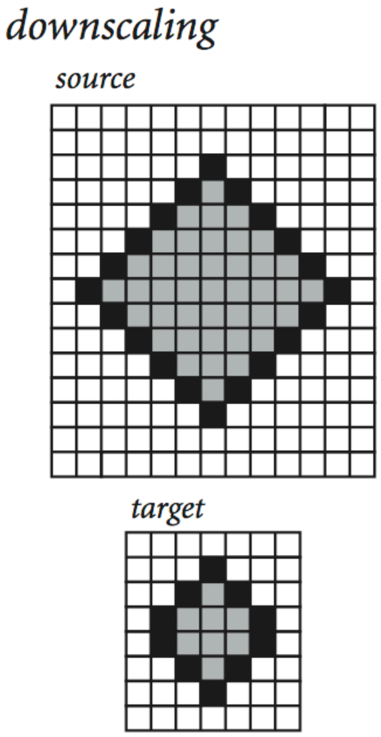
A. Makes an upside-down copy of the image.

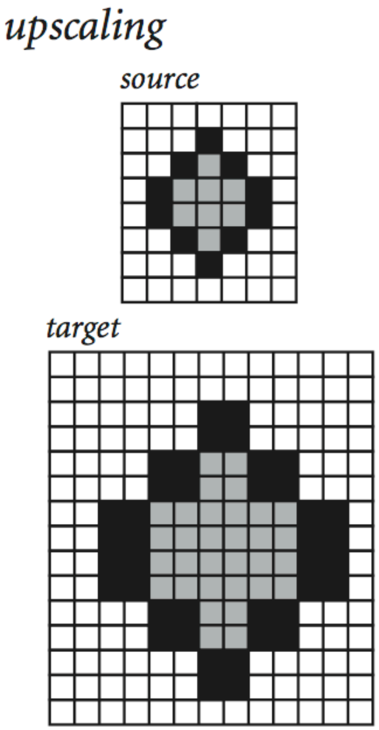


# Image Processing: Scaling Filter

Goal. Shrink or enlarge an image to desired size.

Downscaling. To shrink in half, delete half the rows and columns.
Upscaling. To enlarge to double, replace each pixel by 4 copies.



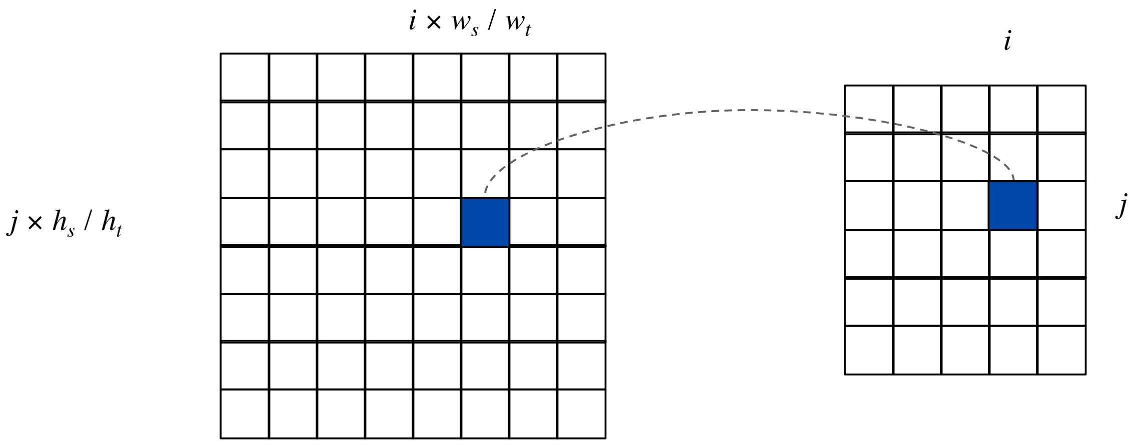


# Image Processing: Scaling Filter

**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 $(i, j)$ in target image to color of pixel $(i \times w_s / w_t \,,\; j \times h_s / h_t)$ in source image.



source image ($w_s$-by-$h_s$)

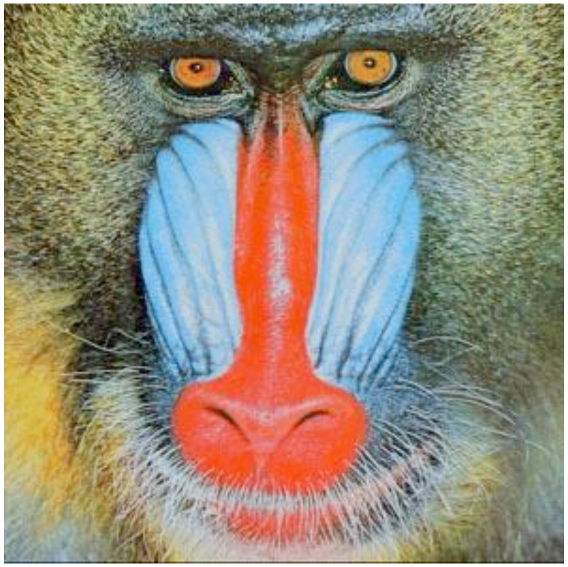
target image ($w_t$-by-$h_t$)

# Image Processing:  Scaling Filter

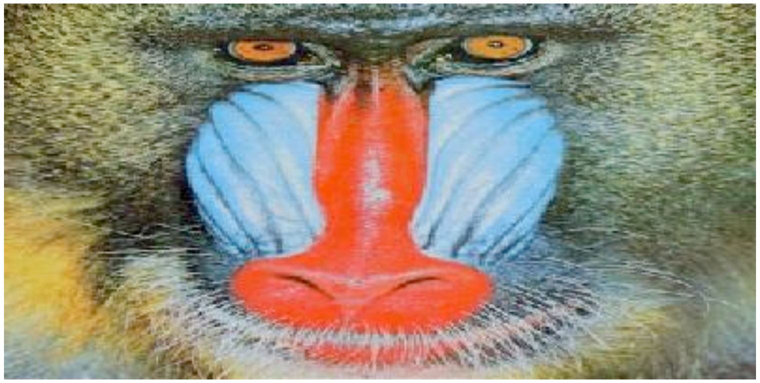
```
import java.awt.Color;

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 ti = 0; ti < w; ti++)
         for (int tj = 0; tj < h; tj++)
         {
            int si = ti * source.width()  / w;
            int sj = tj * source.height() / h;
            Color color = source.get(si, sj);
            target.set(ti, tj, color);
         }
     source.show();
     target.show();
  }
}
```

# Image Processing: Scaling Filter

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

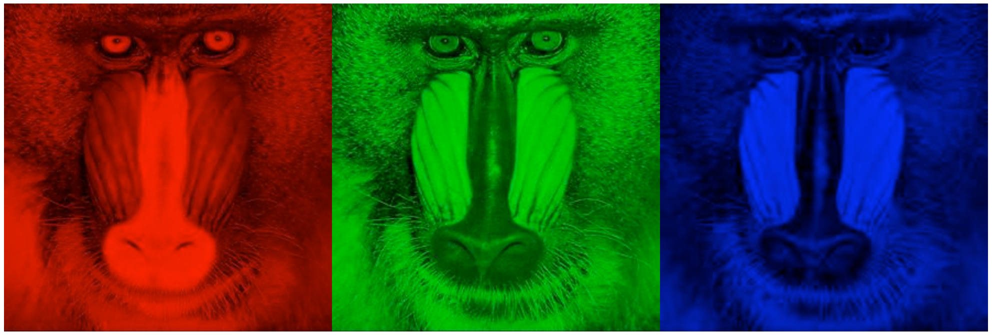


`mandrill.jpg`

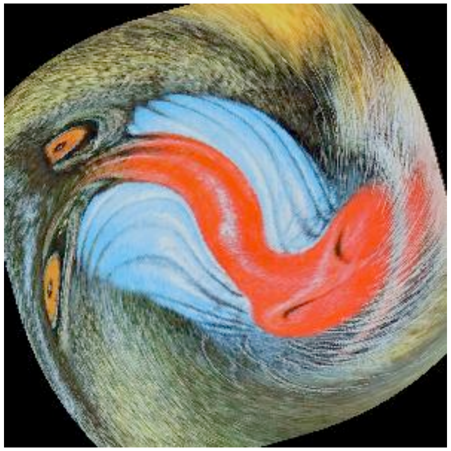


```
% java Scale 400 200 mandrill.jpg
```

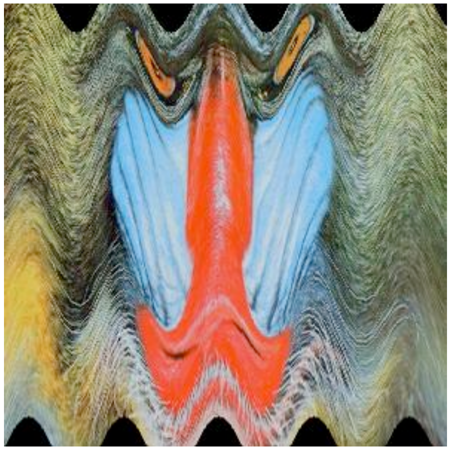
# More Image Processing Effects



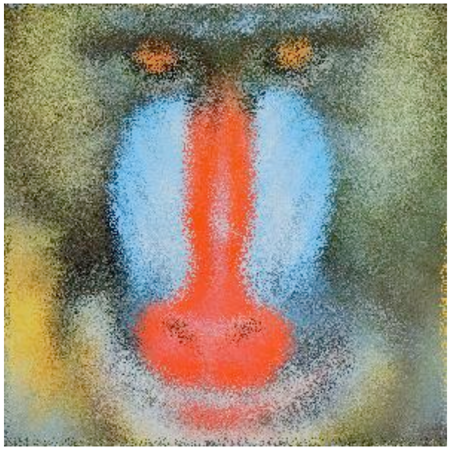
RGB color separation



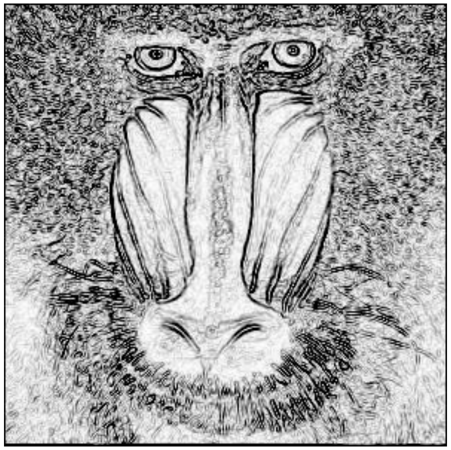
swirl filter



wave filter



glass filter



Sobel edge detection

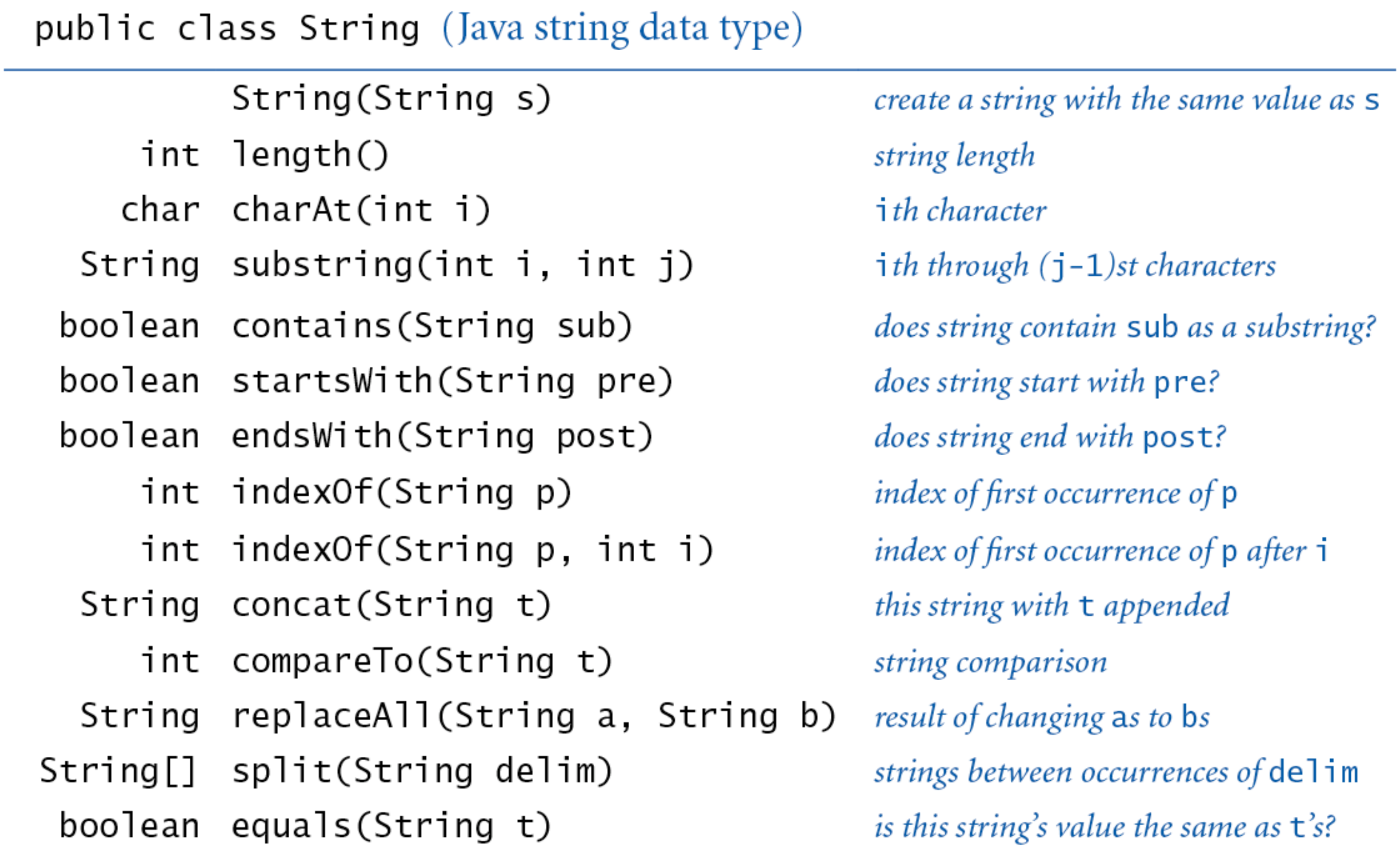
# String Processing

# String Data Type

String data type. Basis for text processing.

Set of values. Sequence of Unicode characters.

API.

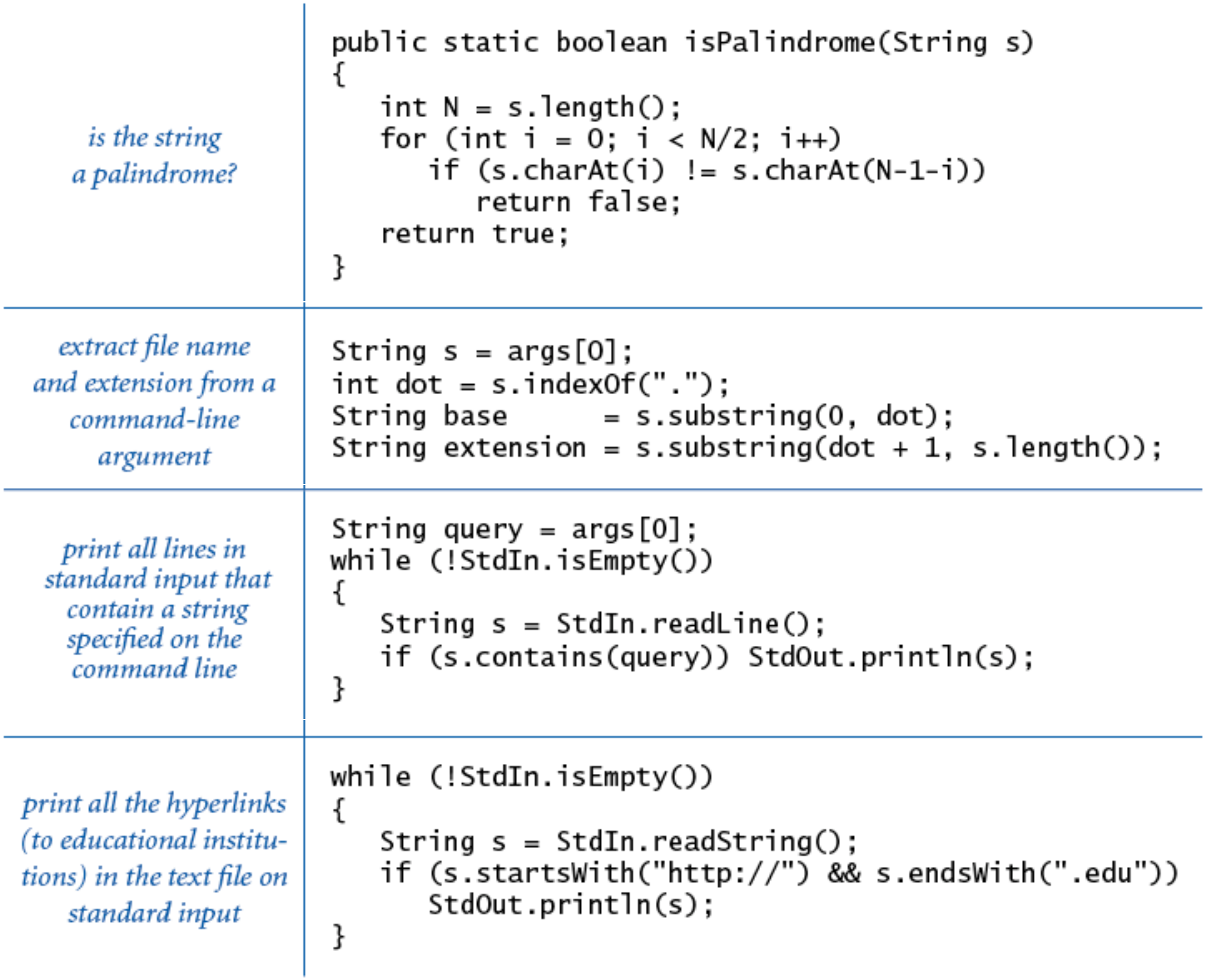


`public class String` (Java string data type)

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

http://java.sun.com/javase/6/docs/api/java/lang/String.html

# Typical String Processing Code



*is the string a palindrome?*

```
public static boolean isPalindrome(String s)
{
   int N = s.length();
   for (int i = 0; i < N/2; i++)
      if (s.charAt(i) != s.charAt(N-1-i))
         return false;
   return true;
}
```

*extract file name and extension from a command-line argument*

```
String s = args[0];
int dot = s.indexOf(".");
String base      = s.substring(0, dot);
String extension = s.substring(dot + 1, s.length());
```

*print all lines in standard input that contain a string specified on the command line*

```
String query = args[0];
while (!StdIn.isEmpty())
{
   String s = StdIn.readLine();
   if (s.contains(query)) StdOut.println(s);
}
```

*print all the hyperlinks (to educational institutions) in the text file on standard input*

```
while (!StdIn.isEmpty())
{
   String s = StdIn.readString();
   if (s.startsWith("http://") && s.endsWith(".edu"))
      StdOut.println(s);
}
```

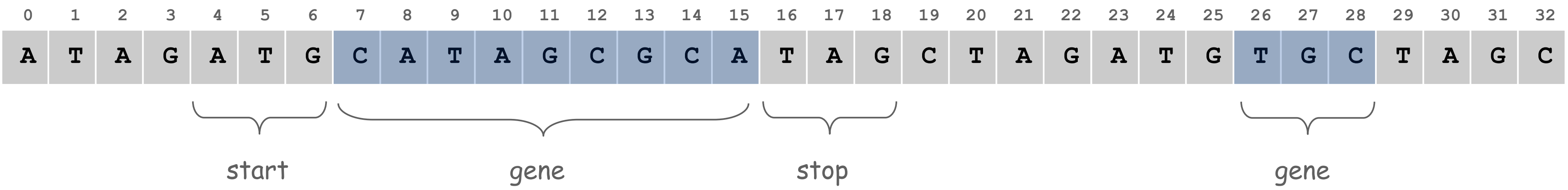
# Gene Finding

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]

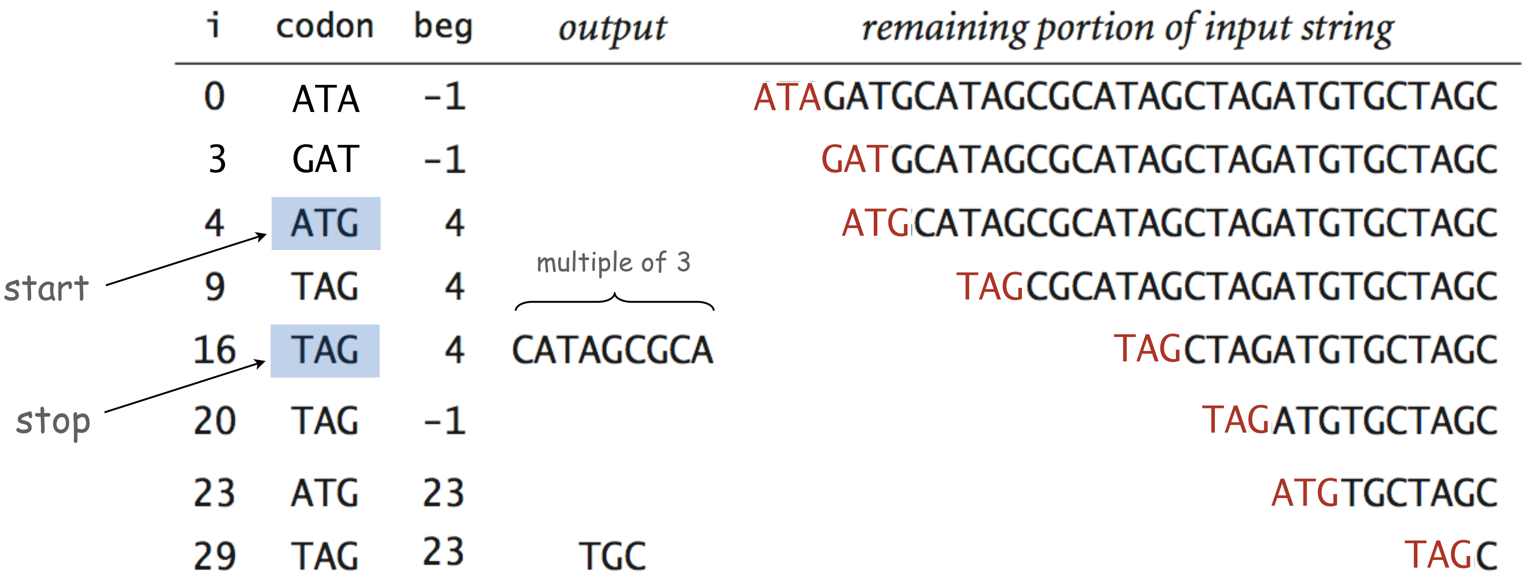


| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | T | A | G | A | T | G | C | A | T | A | G | C | G | C | A | T | A | G | C | T | A | G | A | T | G | T | G | C | T | A | G | C |

# Gene Finding: Algorithm

Algorithm. Scan left-to-right through genome.

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


start
stop
multiple of 3

| i | codon | beg | *output* | *remaining portion of input string* |
|---|---|---|---|---|
| 0 | ATA | -1 | | ATAGATGCATAGCGCATAGCTAGATGTGCTAGC |
| 3 | GAT | -1 | | GATGCATAGCGCATAGCTAGATGTGCTAGC |
| 4 | ATG | 4 | | ATGCATAGCGCATAGCTAGATGTGCTAGC |
| 9 | TAG | 4 | | TAGCGCATAGCTAGATGTGCTAGC |
| 16 | TAG | 4 | CATAGCGCA | TAGCTAGATGTGCTAGC |
| 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 && beg+3 < i)
         {
            String gene = genome.substring(beg+3, i);
            if (gene.length() % 3 == 0)
            {
               StdOut.println(gene);
               beg = -1;
            }
         }
      }
   }
}
```

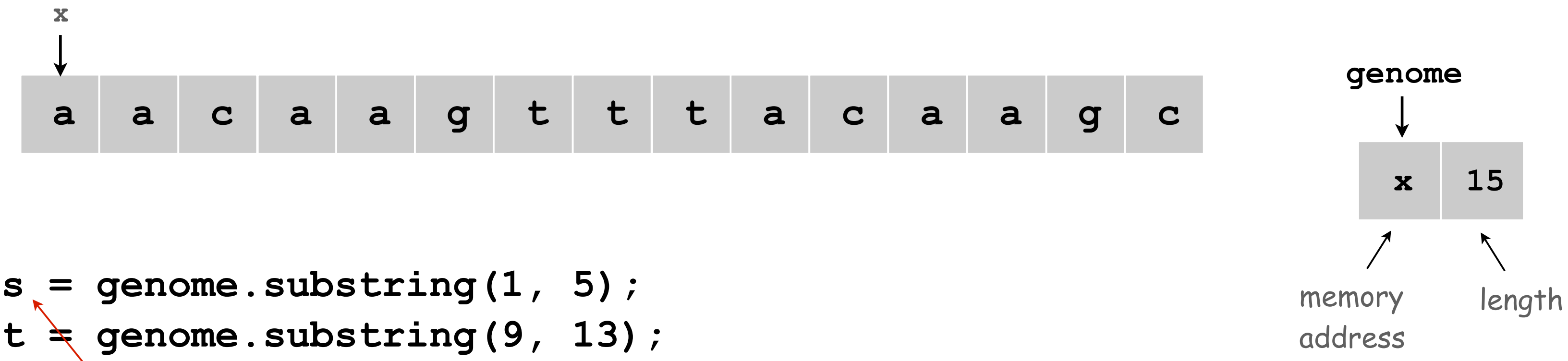
Fixes bug in Prog 3.1.8
TEQ 1: What's the bug?
TEQ 2: Give input that
makes Prog 3.1.8 crash

```
% more genomeTiny.txt
ATAGATGCATAGCGCATAGCTAGATGTGCTAGC


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

# OOP Context for Strings

## Possible memory representation of a string

- `genome = "aacaagtttacaagc";`



x

| a | a | c | a | a | g | t | t | t | a | c | a | a | g | c |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

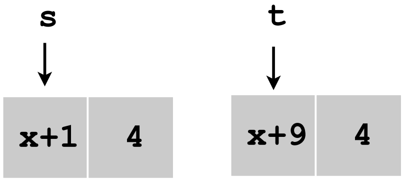
genome

| x | 15 |
|---|---|

memory address

length

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



s

| x+1 | 4 |
|---|---|

t

| x+9 | 4 |
|---|---|

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

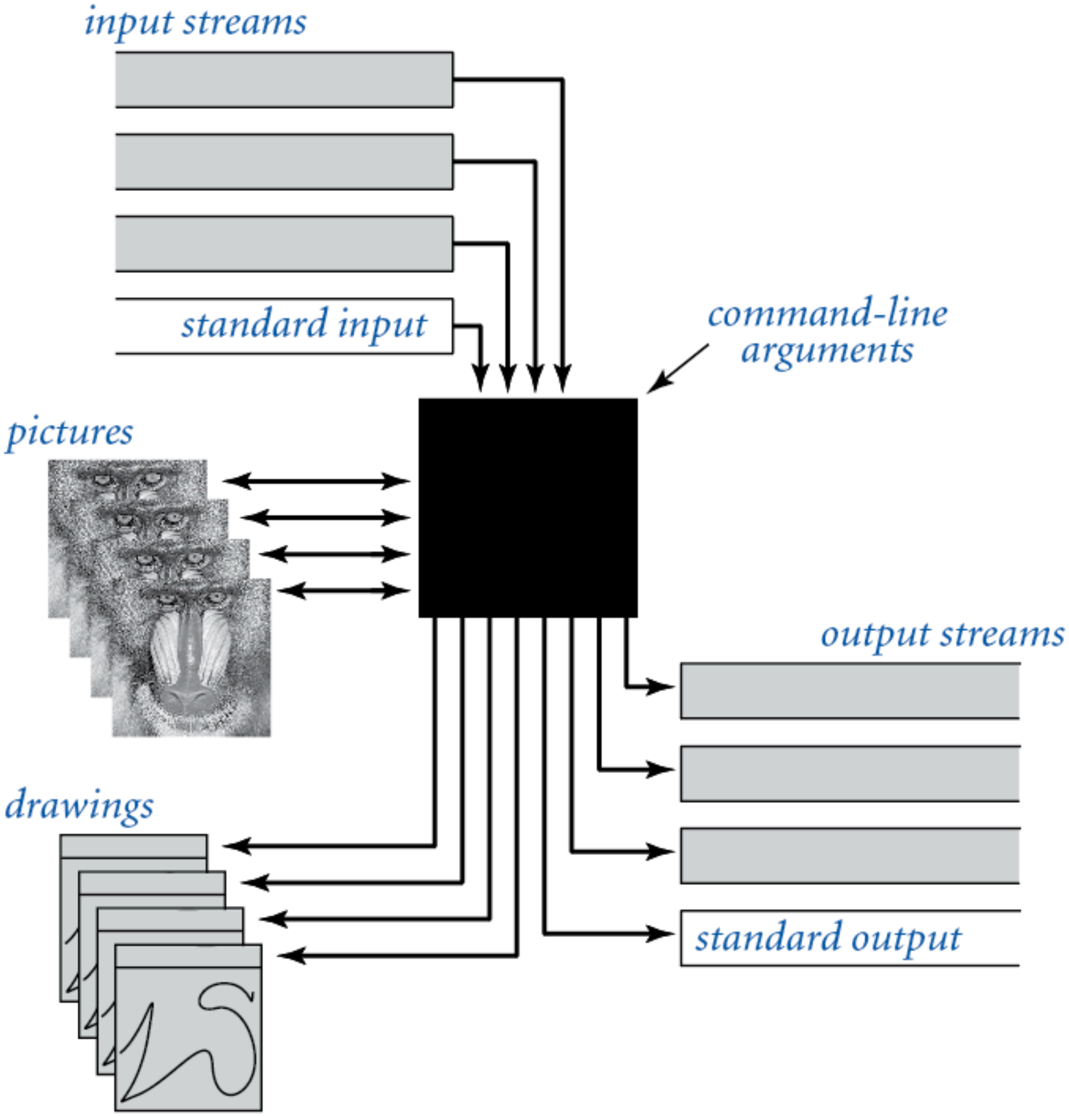
- `(s == t)` is `false`, but `(s.equals(t))` is `true`.

compares addresses

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.

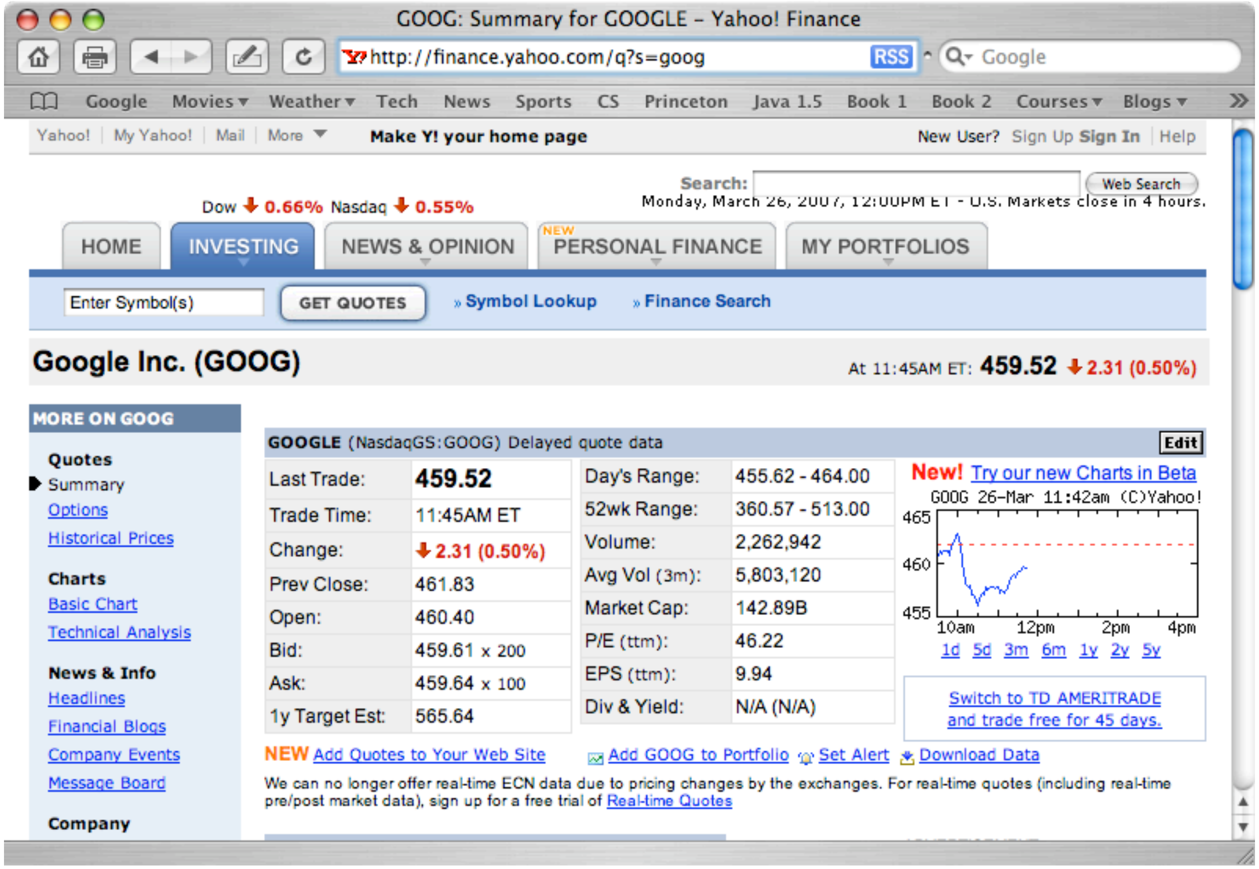
Ex: Are two text files identical?

```
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.

Step 1. Find web source.



GOOG: Summary for GOOGLE – Yahoo! Finance

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

RSS
Google
Google Movies Weather Tech News Sports CS Princeton Java 1.5 Book 1 Book 2 Courses Blogs
Yahoo! My Yahoo! Mail More
Make Y! your home page
New User? Sign Up Sign In Help
Search: Web Search
HOME INVESTING NEWS & OPINION NEW PERSONAL FINANCE MY PORTFOLIOS
Enter Symbol(s) GET QUOTES
» Symbol Lookup » Finance Search

Dow 0.66% Nasdaq 0.55%

Monday, March 26, 2007, 12:00PM ET - U.S. Markets close in 4 hours.

Google Inc. (GOOG)

At 11:45AM ET: 459.52 2.31 (0.50%)

MORE ON GOOG

Quotes
- Summary
- Options
- Historical Prices

Charts
- Basic Chart
- Technical Analysis

News & Info
- Headlines
- Financial Blogs
- Company Events
- Message Board

Company

GOOGLE (NasdaqGS:GOOG) Delayed quote data Edit

| | | | |
|---|---|---|---|
| Last Trade: | 459.52 | Day's Range: | 455.62 - 464.00 |
| Trade Time: | 11:45AM ET | 52wk Range: | 360.57 - 513.00 |
| Change: | 2.31 (0.50%) | Volume: | 2,262,942 |
| Prev Close: | 461.83 | Avg Vol (3m): | 5,803,120 |
| Open: | 460.40 | Market Cap: | 142.89B |
| Bid: | 459.61 x 200 | P/E (ttm): | 46.22 |
| Ask: | 459.64 x 100 | EPS (ttm): | 9.94 |
| 1y Target Est: | 565.64 | Div & Yield: | N/A (N/A) |

New! Try our new Charts in Beta

Switch to TD AMERITRADE and trade free for 45 days.

NEW Add Quotes to Your Web Site Add GOOG to Portfolio Set Alert Download Data

We can no longer offer real-time ECN data due to pricing changes by the exchanges. For real-time quotes (including real-time pre/post market data), sign up for a free trial of Real-time Quotes

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

NYSE symbol

# Screen Scraping

Goal. Find current stock price of Google.

Step 2. Find string representation (HTML code) of web source.

```
…
<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>
…
```

price is string
between <b> and </b>
after "Last Trade"

# Screen Scraping

Goal. Find current stock price of Google.

Step 3. Write code to extract stock price from HTML code.

```
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);
   }
}
```

price is string
between <b> and </b>
after "Last Trade"

```
% java StockQuote goog
459.52
```

- `s.indexOf(t, i)`: index of first occurrence of `t` in `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:"`.

# 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. 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.

Today, you learned to write programs that manipulate colors, pictures, strings, and I/O streams.

Next time.

You will learn to define your own abstractions and to write programs that manipulate them.