

1

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.

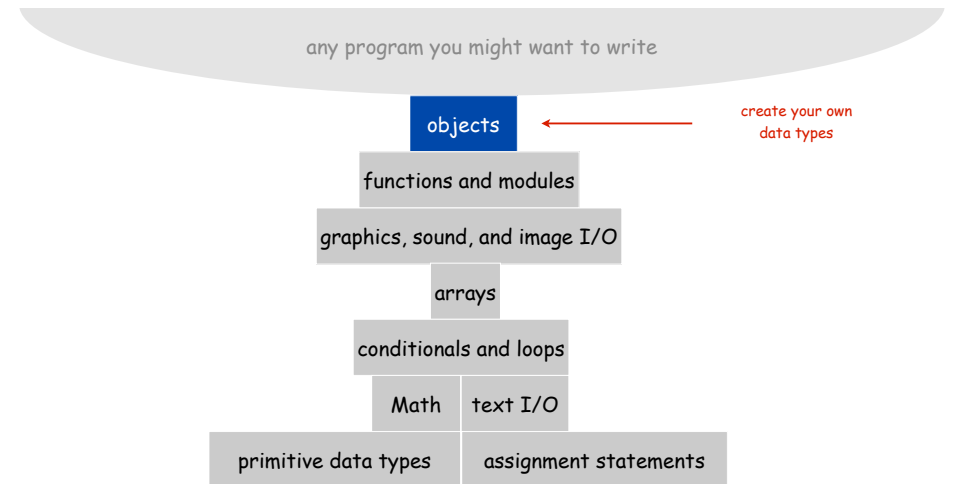
Data Type	Set of Values	Operations
boolean	true, false	not, and, or, xor
int	-2^{31} to $2^{31} - 1$	add, subtract, multiply
double	any of 2^{64} possible reals	add, subtract, multiply

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, ...

3

3.1 Data Types



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.

```

declare a variable (object name)
String s;
call a constructor to create an object
s = new String("Hello, World");
System.out.println(s.substring(0, 5));
object name      call a method that operates
                  on the object's value
dot operator
    
```

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

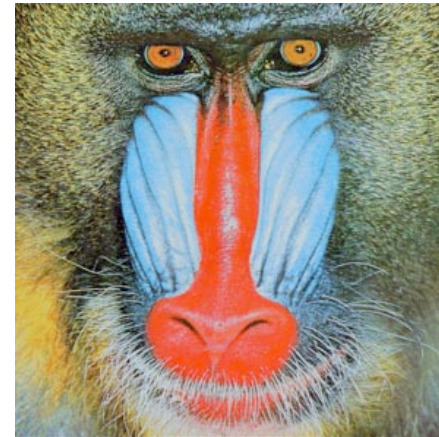
Color Data Type

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

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.

API (Application Programming Interface) specifies **set of operations**.

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

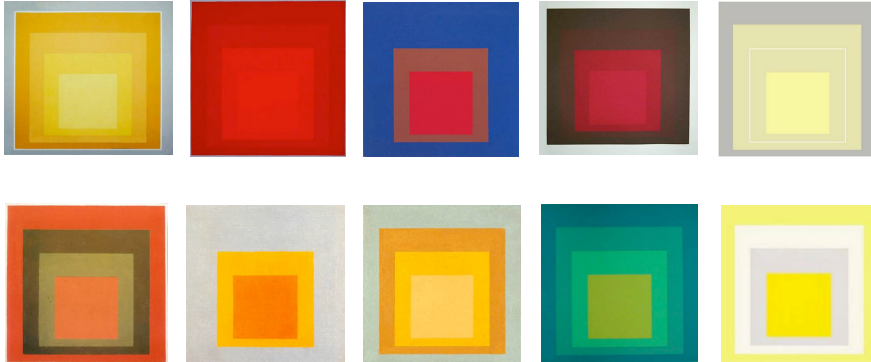
```

    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.

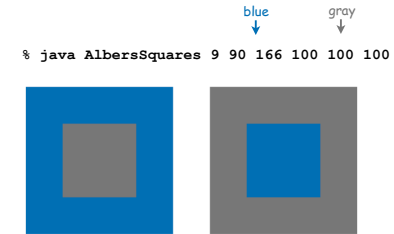


Homage to the Square by Josef Albers (1949-1975)

9

Albers Squares

Josef Albers. Revolutionized the way people think about color.



10

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);
    }
}
```

11

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;
    }
}
```

12

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;
}
```




13

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;
}
```

red	green	blue		
9	90	166	this color	
74	74	74	grayscale version	
0	0	0	black	

round double
to nearest int

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

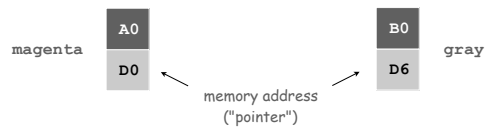
Bottom line. We are writing programs that manipulate color.

14

OOP Context for Color

Possible memory representation (in TOY).

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.

15

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();
```

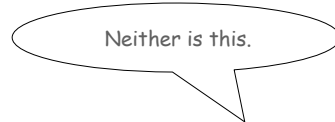
OOP. Natural vehicle for studying abstract models of the real world.

16

This is Not a Pipe



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



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

17

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();
    }
}
```

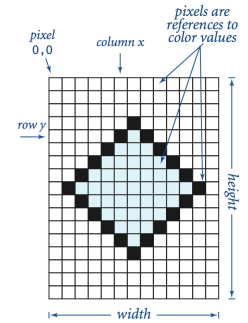
set each
pixel to
gray

19

Picture Data Type

Raster graphics. Basis for image processing.

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



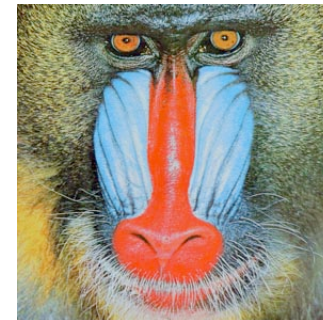
API.

```
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
}
```

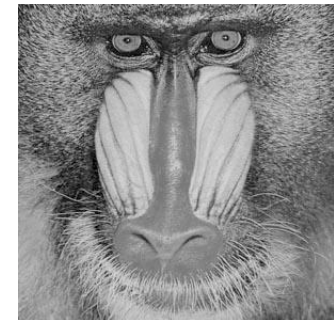
18

Image Processing: Grayscale Filter

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



mandrill.jpg



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

20

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();
```

21

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();
```

22

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();
```

23

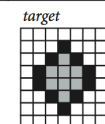
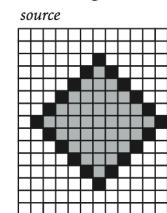
Image Processing: Scaling Filter

Goal. Shrink or enlarge an image to desired size.

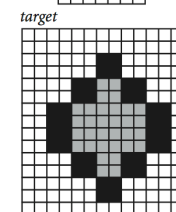
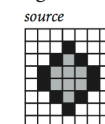
Downscaling. To shrink in half, delete half the rows and columns.

Upscaling. To enlarge to double, replace each pixel by 4 copies.

downscaling



upscaling



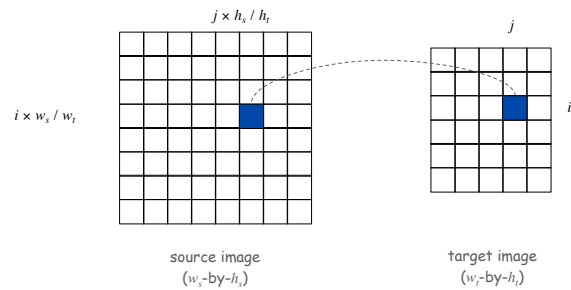
24

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 row index by w_s / w_t .
- Scale column 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.



25

Image Processing: Scaling Filter

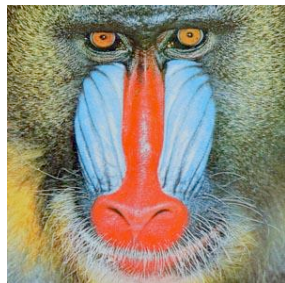
```
import java.awt.Color;

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();
    }
}
```

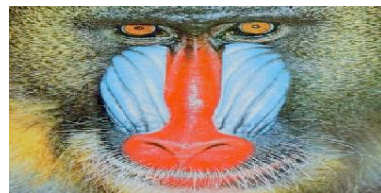
26

Image Processing: Scaling Filter

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

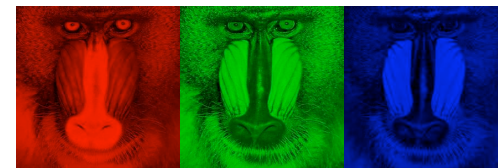


mandrill.jpg

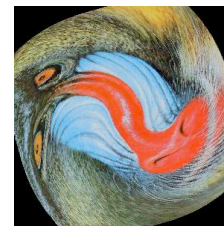


% java Scale 400 200 mandrill.jpg

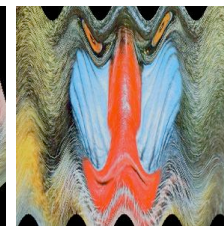
More Image Processing Effects



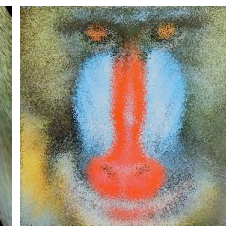
RGB color separation



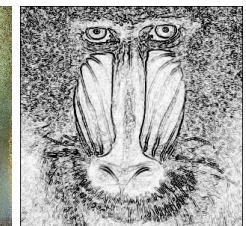
swirl filter



wave filter



glass filter



Sobel edge detection

27

28

String Processing



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 to b
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

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

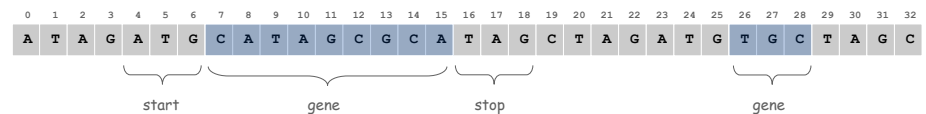
Gene Finding

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>	output	remaining portion of input string
	0		-1		ATAGATGCATAGCGCATAGCTAGATGTGCTAGC
	1	TAG	-1		TAGATGCATAGCGCATAGCTAGATGTGCTAGC
	4	ATG	4		ATGCATAGCGCATAGCTAGATGTGCTAGC
start →	9	TAG	4	CATAGCGCA	TAGCGCATAGCTAGATGTGCTAGC
	16	TAG	4		TAGCTAGATGTGCTAGC
stop →	20	TAG	-1		TAGATGTGCTAGC
	23	ATG	23		ATGTGCTAGC
	29	TAG	23	TGC	TAGC

33

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
```

34

OOP Context for Strings

Possible memory representation of a string (using TOY addresses).

- `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.

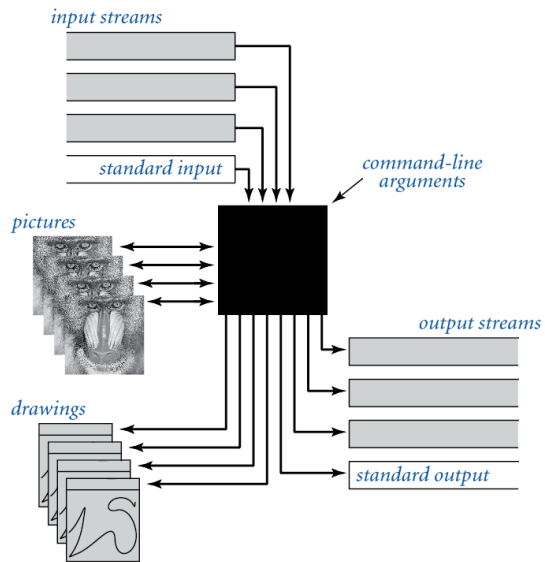
compares pointers

compares character sequences

35

In and Out

Bird's Eye View (Revisited)



Non-Standard Input

Standard input. Read from terminal window. or use OS to redirect from one file

Goal. Read from **several** different input streams.

In data type. Read text from stdin, a file, a web site, or network.

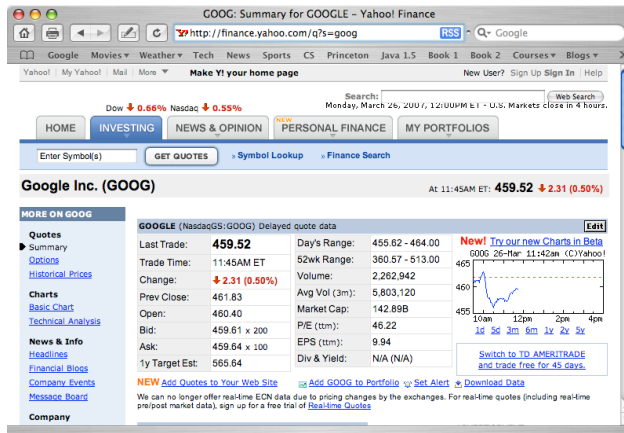
Ex: Are two text files identical?

```
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.



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 and 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 and
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 `` and `` after "Last Trade:".

41

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.

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.

43

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

42